

# Optimal path planning in cluttered environment using RRT\*-AB

Iram Noreen<sup>1</sup> · Amna Khan<sup>1</sup> · Hyejeong Ryu<sup>2</sup> ·  
Nakju Lett Doh<sup>3</sup> · Zulfiqar Habib<sup>1</sup> 

Received: 18 July 2016 / Accepted: 23 July 2017 / Published online: 3 August 2017  
© Springer-Verlag GmbH Germany 2017

**Abstract** Rapidly exploring Random Tree Star (RRT\*) has gained popularity due to its support for complex and high-dimensional problems. Its numerous applications in path planning have made it an active area of research. Although it ensures probabilistic completeness and asymptotic optimality, its slow convergence rate and large dense sampling space are proven problems. In this paper, an off-line planning algorithm based on RRT\* named RRT\*-adjustable bounds (RRT\*-AB) is proposed to resolve these issues. The proposed approach rapidly targets the goal region with improved computational efficiency. Desired objectives are achieved through three novel strategies, i.e., connectivity region, goal-biased bounded sampling, and path optimization. Goal-biased bounded sampling is performed within boundary of connectivity region to find the initial path. Connectivity region is flexible enough to grow for complex environment. Once path is found, it is optimized gradually using node rejection and concentrated bounded sampling.

Final path is further improved using global pruning to erode extra nodes. Robustness and efficiency of proposed algorithm is tested through experiments in different structured and unstructured environments cluttered with obstacles including narrow and complex maze cases. The proposed approach converges to shorter path with reduced time and memory requirements than conventional RRT\* methods.

**Keywords** Robot path planning · Optimal path · Intelligent sampling · RRT\* · RRT\*-AB · Cluttered environment

## 1 Introduction

Motion planning refers to collision-free path generation from an initial state to a specified goal state. It has widespread applications such as autonomous cars [1], Unmanned Aerial Vehicles (UAVs) [2], planet exploration rovers for space missions, surveillance operations [3] and computer-aided surgery [4]. A number of path planning algorithms including deterministic approaches [5,6], artificial potential fields [6–8], grid-based methods [9,10], neural networks [11,12] and evolutionary approaches [13–16] have been used to solve motion planning problem in static and dynamic environment. An extensive comparative study describing advantages and shortcomings of these path planning approaches exists in the literature [7,8,17–19].

Sampling-based planning (SBP) methods are the most popular and influential advancement in path planning [6,20]. SBPs are probabilistic complete, i.e., a solution will be provided, if one exists, given infinite run time [6,21]. They are capable of dealing with high-dimensional complex problems. Major edge of SBPs over other state-of-the-art algorithms is that they build a road map of feasible trajectories without explicit information of obstacles in configuration space

✉ Zulfiqar Habib  
drzhabib@ciitlahore.edu.pk

Iram Noreen  
iramnoreen@gmail.com

Amna Khan  
amna.cs@gmail.com

Hyejeong Ryu  
hjryu@kangwon.ac.kr

Nakju Lett Doh  
nakju@korea.ac.kr

<sup>1</sup> Department of Computer Sciences, COMSATS Institute of Information Technology, Lahore, Pakistan

<sup>2</sup> Division of Advanced Mechanical Engineering, Kangwon National University, Chuncheon, Korea

<sup>3</sup> School of Electrical Engineering, Korea University, Seoul, Korea

[6,22,23]. Further, their easy implementation and low computational cost in high dimensions enable them to deal with real-time complex problems. Rapidly exploring Random Tree (RRT) by Lavalle [24] and Probabilistic Road Map (PRM) by Kav-raki et al. [25] are most popular SBP algorithms [6,26]. However, dependence of PRM on obstacles geometry makes it suitable only for static environment. RRT is more feasible for dynamic cluttered environment than PRM and has inherent support for non-holonomic constraints as well [26]. Due to these advantages, RRT and its further extensions [25] have been extensively explored in recent decade.

Karaman and Frazzoli [20] proposed an asymptotic optimal extension of RRT, named RRT\*. RRT\* finds initial path quickly than RRT and refines it in successive iterations. As the number of iterations approach infinity, it generates an optimal or near-optimal path [20,27]. This asymptotic optimal property makes RRT\* algorithm very expedient for real-time applications. However, RRT\* produces sub-optimal path and has issues related to slow convergence. Major constraints of RRT\* addressed for improvement in this paper are:

1. rejection of beneficial samples in vicinity of goal region during initial iterations because they are not directly connectable to existing nodes in the tree during early iterations;
2. exploration and sampling in whole configuration space, adding far away un-necessary nodes not contributing in final path;
3. significantly high memory requirements due to the large number of nodes in tree; and
4. slow convergence towards optimal path.

In this paper, a new off-line algorithm named as RRT\*-adjustable bounds (RRT\*-AB) is proposed to overcome the aforementioned issues. RRT\*-AB performs informed exploration in search space of known environment. It generates highly converged tree populated with useful nodes, as a result reducing memory requirements. First, connectivity region and goal-biased bounded sampling make fast convergence towards optimal solution. These two features enhance the valuable samples in the vicinity of goal region very quickly, as a result leading to attain optimal or near-optimal path efficiently in less time. Further, path is optimized at fast rate using concentrated sampling, node rejection technique and path pruning. Concentrated sampling selects nodes only in limited vicinity of found path, and node rejection technique rejects inadequate nodes to insert in tree. In the end, path is further optimized by applying global pruning. The proposed algorithm RRT\*-AB has been tested for its robustness in different scenarios of structured and unstructured cluttered environments including cases of narrow and complex

maze passages. Simulation results show that our proposed approach is efficient in memory requirements, execution time and path length.

Rest of the paper is organized as follows. The related work is discussed in Sect. 2. In Sect. 3, problem definition is described. The proposed approach RRT\*-AB is presented in Sect. 4. Results are discussed in Sect. 5. Section 6 concludes the paper and highlights future research directions.

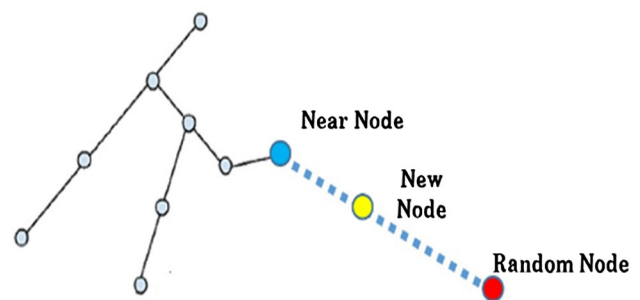
## 2 Related work

Karaman and Frazzoli [20] proposed RRT\* with proven asymptotic optimal property. RRT\* executes either for predefined fixed number of iterations or fixed period of time. RRT\* constructs tree in free space say  $Z_{\text{free}}$  by randomly exploring configuration space. Tree starts from an initial state say  $z_{\text{init}}$  to find path towards a goal state say  $z_{\text{goal}}$ . Tree grows gradually with the increase in iterations. In each iteration a random state say  $z_{\text{rand}}$  is selected from free space [19,28]. If random state lies in obstacle space, it is rejected, called as *TRAPPED* condition. If newly added node itself is goal location, then path is found, referred as *REACHED* condition. A nearest node, i.e.  $z_{\text{nearest}}$  in tree is searched according to a metric  $\sigma$ . If  $z_{\text{rand}}$  is reachable from nearest node  $z_{\text{nearest}}$  according to predefined step size, then it is added in tree as new node called  $z_{\text{new}}$  by connecting with nearest node, which becomes its parent. Otherwise, planner returns a new node  $z_{\text{new}}$  by using a steering function and adds it in tree by connecting it with nearest node as parent. A cost value is assigned to newly added node according to a defined cost function. It is referred as *EXPAND* condition [19,28]. This node expansion process of RRT\* is illustrated in Fig. 1.

During each iteration, near neighbours are searched for new node within the area of a radius defined by

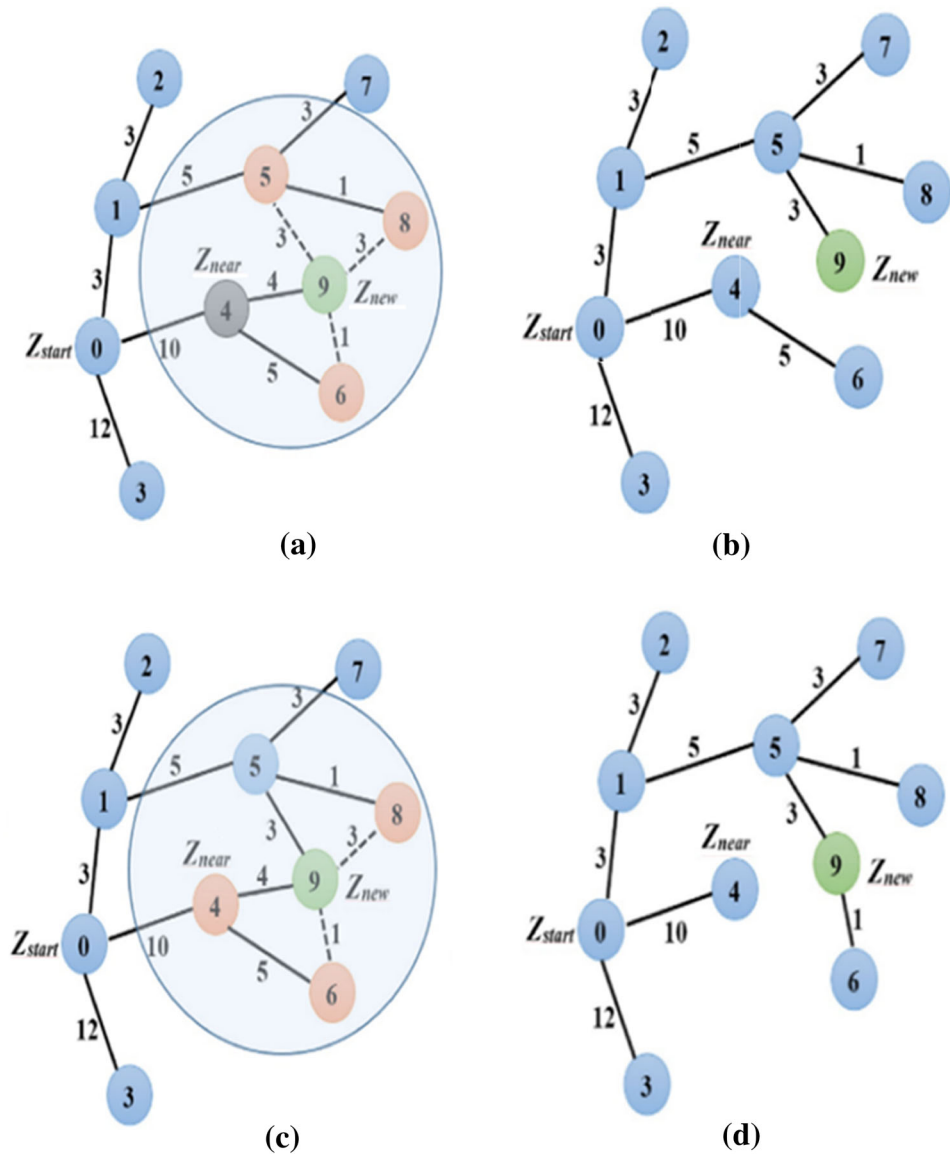
$$k = \gamma (\log(n)/n)^{(1/d)}, \quad (1)$$

where  $d$  is the dimension of configuration space,  $n$  is the number of tree nodes, and  $\gamma$  is the planning constant based on environment [19,28]. Then, a node rewiring operation



**Fig. 1** Node expansion process in RRT\* [19]

**Fig. 2** Near vertices and rewiring operations in RRT\*; **a** finding near vertices, **b** selection of best parent, **c** checking cost, **d** rewired according to minimum cost [28]



rebuilds the tree for less cost within this area using identified near neighbours [29]. This process is shown in Fig. 2. Once an initial path is found, execution continues for the rest of iterations to improve the path [19,28].

RRT\* improved path quality as compared to initial RRT approach; however, it required large number of iterations to optimize initial path resulting slow convergence [19,28]. In 2013, Nasir et al. [30] proposed a variant of RRT\* called RRT\*-Smart to resolve the issues of slow convergence in RRT\*. RRT\*-Smart converges to optimal or near-optimal solution at fast pace than RRT\* using intelligent sampling and path optimization techniques. Once, an initial path is found RRT\*-Smart optimizes path by alternatively applying

**Algorithm 1:**  $T \leftarrow RRT^*(z_{init}, z_{goal})$

```

1  $T \leftarrow InitializeTree()$ ;
2  $T \leftarrow InsertNode(\phi, z_{init}, T)$ ;
3 for  $i \leftarrow 0$  to  $N$  do
4    $z_{rand} \leftarrow Sample(i)$ ;
5    $z_{nearest} \leftarrow Nearest(T, z_{rand})$ ;
6    $(z_{new}, U_{new}) \leftarrow Steer(z_{nearest}, z_{rand})$ ;
7   if  $Obstaclefree(z_{new})$  then
8      $z_{near} \leftarrow Near(T, z_{new}, |V|)$ ;
9      $z_{min} \leftarrow Chooseparent(z_{near}, z_{nearest}, z_{new})$ ;
10     $T \leftarrow InsertNode(z_{min}, z_{new}, T)$ ;
11     $T \leftarrow Rewire(T, z_{near}, z_{min}, z_{new})$ ;
12 return  $T$ 

```

**Algorithm 2:**  $T \leftarrow \text{RRT}^*\text{-Smart}(z_{\text{init}}, z_{\text{goal}})$ 


---

```

1  $T \leftarrow \text{InitializeTree}()$ ;
2  $T \leftarrow \text{InsertNode}(\phi, z_{\text{init}}, T)$ ;
3 for  $i \leftarrow 0$  to  $N$  do
4   if  $i \leftarrow n + b, n + 2b, n + 3b \dots$  then
5      $z_{\text{rand}} \leftarrow \text{Sample}(i, z_{\text{beacons}})$ ;
6   else
7      $z_{\text{rand}} \leftarrow \text{Sample}(i)$ ;
8    $z_{\text{nearest}} \leftarrow \text{Nearest}(T, z_{\text{rand}})$ ;
9    $(z_{\text{new}}, U_{\text{new}}) \leftarrow \text{Steer}(z_{\text{nearest}}, z_{\text{rand}})$ ;
10  if  $\text{Obstaclefree}(z_{\text{new}})$  then
11     $z_{\text{near}} \leftarrow \text{Near}(T, z_{\text{new}}, |V|)$ ;
12     $z_{\text{min}} \leftarrow \text{Chooseparent}(z_{\text{near}}, z_{\text{nearest}}, z_{\text{new}})$ ;
13     $T \leftarrow \text{InsertNode}(z_{\text{min}}, z_{\text{new}}, T)$ ;
14     $T \leftarrow \text{Rewire}(T, z_{\text{near}}, z_{\text{min}}, z_{\text{new}})$ ;
15  if  $\text{InitialPathFound}$  then
16     $n \leftarrow i$ ;
17     $(T, \text{directcost}) \leftarrow \text{PathOptimization}(T, z_{\text{init}}, z_{\text{goal}})$ ;
18    if  $\text{directcostnew} < \text{directcostold}$  then
19       $z_{\text{beacons}} \leftarrow \text{PathOptimization}(T, z_{\text{init}}, z_{\text{goal}})$ ;
20 return  $T$ 

```

---

random sampling and intelligent sampling at regular intervals [19,28]. These intervals are managed by biasing ratio, defined by

$$\text{Biasing Ratio} = (n/Z_{\text{free}}) * B, \quad (2)$$

where  $n$  is the number of tree nodes,  $Z_{\text{free}}$  is free space, and  $B$  is a programmer-dependent constant which sets a trade-off between convergence rate and exploration of space. Further, intelligent sampling is performed only around biasing beacon points identified in initial path. RRT\*-Smart converges more efficiently than RRT\*; however, it requires adjustment of biasing ratio which makes it highly sensitive to type of environment map. Moreover, process to identify new biasing points each time when path improves also causes computational overhead. Additionally, inefficient Biasing ratio increases random sampling over intelligent sampling. Thus, RRT\*-Smart still explores the whole configuration space and requires thousands of iterations to converge to optimal path. This phenomenon generates very dense tree, increasing memory requirements with large number of such nodes which do not contribute in final path [19,28]. Approaches of RRT\* and RRT\*-Smart are shown in Algorithms 1 and 2, respectively. A comparative feature analysis of RRT, RRT\* and RRT\*-Smart is provided in [28].

Gammell et al. [31] proposed another variant of RRT\* called Informed RRT\*. Informed RRT\* creates an elliptical area between start and goal positions using an ellipsoidal informed subset, once an initial sub-optimal path is found. Sampling area of ellipse also decreases with the convergence

in path. However, Informed RRT\* was designed specifically to address extremely narrow passages. Therefore, its elliptical sub area could not be reduced effectively in cluttered environment. As a result, sampling in larger ellipse could not converge quickly to optimal solution in reasonable computation time.

### 3 Problem definition

Let the given state space be denoted by a set  $Z \subset \mathbb{R}^n$ ,  $n \in \mathbb{N}$ , and  $n = 2$ , where  $n$  represents the dimension of given space. Further, configuration space occupied with obstacles is denoted by  $Z_{\text{obs}} \subset Z$  and obstacle-free region is denoted by  $Z_{\text{free}} = Z/Z_{\text{obs}}$ .  $z_{\text{goal}} \in Z_{\text{free}}$  is the goal and  $z_{\text{init}} \in Z_{\text{free}}$  is the starting point. This paper only considers off-line planning in Euclidean space and positive Euclidean distance between any two states, e.g.,  $z_1 \in Z_{\text{free}}$  and  $z_2 \in Z_{\text{free}}$  is denoted by  $d(z_1, z_2)$ . Let the path connecting states  $z_1$  and  $z_2$  be denoted by a metric function  $\sigma : [0, s]$ , such that  $\sigma(0) = z_1$  and  $\sigma(s) = z_2$ , where  $s$  is the positive scalar length of the path.  $\sigma_f \in Z_{\text{free}}$  is an end-to-end feasible path and set of all collision-free paths in  $Z_{\text{free}}$  is denoted as  $\Sigma_f$ , i.e.,  $\sigma_f \in \Sigma_f$ . Algorithm runs to find a feasible path  $\sigma_f \in \Sigma_f$  from  $\sigma_f(0) = z_{\text{init}}$  to  $\sigma_f(s) = z_{\text{goal}}$  following the system constraints. In order to find the solution, algorithm builds a tree  $T = (V, E)$ , where  $V$  are the vertices sampled from  $Z_{\text{free}}$ , and  $E$  are the edges to connect these vertices. The following motion planning problems will be considered in the proposed algorithm:

**Problem 1 (Feasible path solution):** Find a path  $\sigma_f : [0, s]$ , if one exists in  $Z_{\text{free}} \subset Z$  such that  $\sigma_f(0) = z_{\text{init}} \in Z_{\text{free}}$  and  $\sigma_f(s) = z_{\text{goal}} \in Z_{\text{free}}$  and report failure if no such path exists.

**Problem 2 (Optimal path solution):** Find an optimal path  $\sigma_f^* : [0, s]$  which connects  $z_{\text{init}}$  and  $z_{\text{goal}}$  in  $Z_{\text{free}} \subset Z$  such that the cost of the path  $\sigma_f^*$  is minimum, i.e.,  $c(\sigma_f^*) = \{ \min_{\sigma_f \in \Sigma_f} c(\sigma_f) : \sigma_f \in \Sigma_f \}$ .

**Problem 3 (Convergence to optimal solution):** Find an optimal path  $\sigma_f^* : [0, s]$  in  $Z_{\text{free}} \subset Z$  in the least possible time  $t \in \mathbb{R}$ .

### 4 Proposed approach

This section describes proposed approach RRT\*-AB. The proposed approach aims to address limitations described earlier. The proposed approach tends to grow more promising tree by exploring only favourable regions to find optimal path in less time and memory requirements. The innovation of proposed approach lies in features named as connectivity

region, bounded sampling, and path optimization. The proposed approach is given in Algorithm 3. Major operations of the proposed approach are explained as following:

---

**Algorithm 3:**  $T \leftarrow RRT^*-AB(z_{init}, z_{goal})$

---

```

1  $T \leftarrow InitializeTree()$ ;
2  $T \leftarrow InsertNode(\phi, z_{init}, T)$ ;
3  $C_{Region} \leftarrow ConnectivityRegion(z_{init}, z_{goal}, T)$ ;
4 for  $i \leftarrow 0$  to  $N$  do
5    $z_{rand} \leftarrow BoundedSample(i, C_{Region})$ ;
6    $z_{nearest} \leftarrow Nearest(T, z_{rand})$ ;
7    $(z_{new}, U_{new}) \leftarrow Steer(z_{nearest}, z_{rand})$ ;
8   if  $CollisionCheck(z_{new})$  then
9      $z_{near} \leftarrow Near(T, z_{new}, |V|)$ ;
10     $z_{min} \leftarrow ChooseParent(z_{near}, z_{nearest}, z_{new})$ ;
11     $T \leftarrow InsertNode(z_{min}, z_{new}, T)$ ;
12     $T \leftarrow Rewire(T, z_{near}, z_{min}, z_{new})$ ;
13    if  $PathFound(T)$  then
14       $C_{Region} \leftarrow ConnectivityRegion(z_{init}, z_{goal}, T)$ ;
15    else if  $CompleteScan(C_{Region})$  then
16       $C_{Region} \leftarrow ConnectivityRegion(z_{init}, z_{goal}, T)$ ;
17  $T \leftarrow PrunePath(T)$ ;
18 return  $T$ 

```

---

*ConnectivityRegion:* This function identifies a connectivity region denoted by  $C_{Region}$  using  $D_{scale}$  covering the search space between  $z_{init}$  and  $z_{goal}$ .

*BoundedSample:* It randomly samples a state  $z_{rand} \in C_{Region} \subset Z_{free}$ .

*Nearest:* The function  $Nearest(T, z_{rand})$  returns the nearest node from  $T = (V, E)$  to  $z_{rand}$  according to cost function.

*Steer:* The function  $Steer(z_{nearest}, z_{rand})$  provides a control input  $U_{new}$  to drive the system from  $z(0) = z_{rand}$  to  $z(1) = z_{nearest}$  along the path  $z : [0, 1] \rightarrow Z$  giving  $z_{new}$  at a distance  $\Delta q$  from  $z_{nearest}$  towards  $z_{rand}$ , where  $\Delta q$  is the incremental distance.

*CollisionCheck:* This function determines whether a path  $z(t) \in Z_{free}$  such that  $z : [0, 1]$  for all  $t = 0$  to  $t = 1$ .

*Near:* The function  $Near(T, z_{new}, |V|)$  returns the nearby neighbouring nodes defined by Eq. 1.

*ChooseParent:* This function selects the best parent  $z_{min}$  from the nearby nodes.

*InsertNode:* This function adds a node  $z_{new}$  to  $V$  in the tree  $T = (V, E)$  and connects node  $z_{min}$  as its parent. A cost is assigned to  $z_{new}$  which is equal to the cost of its parent plus the Euclidean cost returned by the distance function between  $z_{new}$  and its parent  $z_{min}$ .

*Rewire:* The function  $Rewire(T, z_{near}, z_{min}, z_{new})$  checks if the cost to the nodes in  $z_{near}$  is less through  $z_{new}$  as compared to their older costs. If it is less for a particular node, its parent is changed to  $z_{new}$ .

*CompleteScan:* It returns true if one whole scan within  $C_{Region}$  is completed. If after single complete scan path is not found, then algorithm has to grow  $C_{Region}$ .

*PrunePath:* It returns nodes from path that are connectable directly without a collision with obstacles.

### 4.1 Connectivity region and intelligent bounded sampling

Proposed algorithm initializes tree  $T$  with  $z_{init}$  as root (see steps 1–2 in Algorithm 3). Then it populates its tree using *intelligent bounded sampling* which randomly selects nodes in limited region named as  $C_{Region}$ .

$C_{Region}$  limits the search space for sampling of new random nodes.  $C_{Region}$  is based on simple step of defining the search space between  $z_{init}$  and  $z_{goal}$  using expansion distance scale called  $D_{scale}$  using

$$D_{scale} = E/m, \tag{3}$$

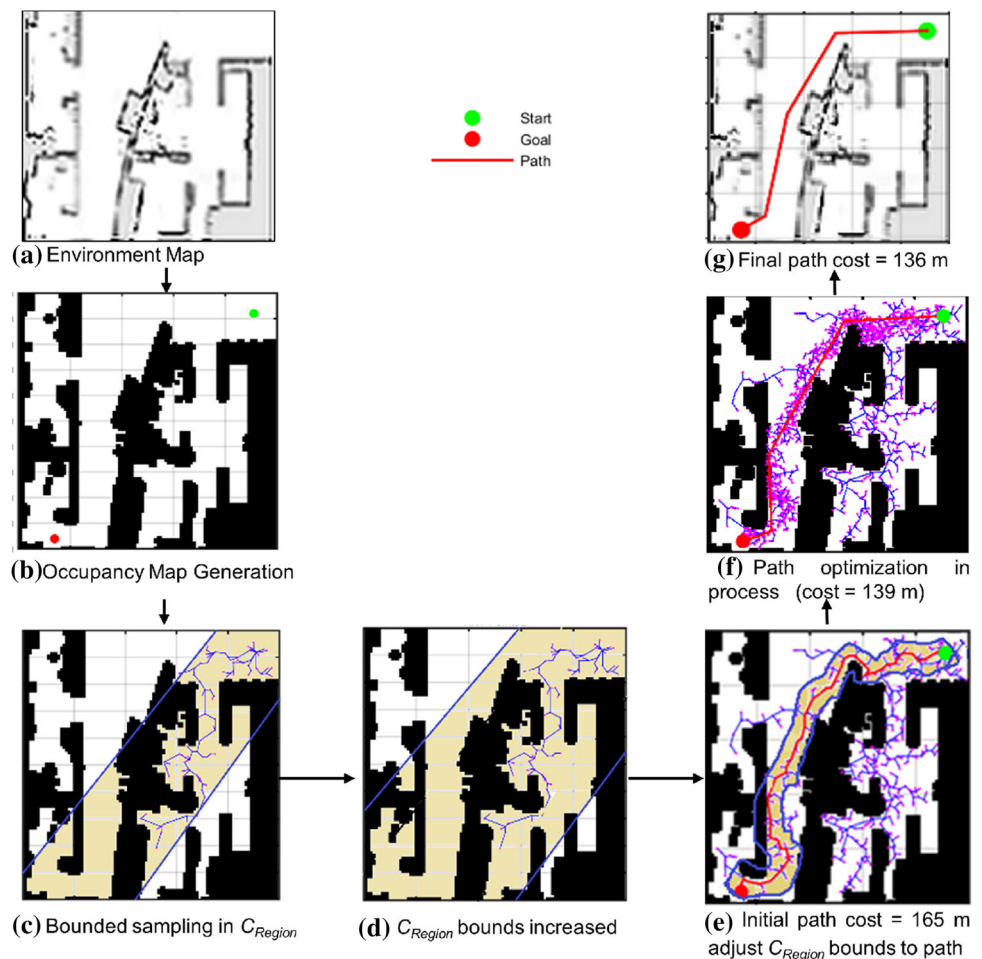
where  $E$  is the size of environment map and  $m$  is the expansion factor (see step 3 in Algorithm 3), as shown in Fig. 3c, d. By default  $m$  is 4, which gives the scale to generate  $C_{Region}$  half the size of the environment map.  $m$  could be increased to get a narrow  $C_{Region}$ . Intelligent bounded sampling randomly selects nodes within bounds of  $C_{Region}$  (see step 5 in Algorithm 3) using goal-biased heuristic. Steps of near neighbour search, best parent selection, rewiring and node insertion are performed same as in RRT\* (see steps 6–12 in Algorithm 3) during this procedure.

A path is found when intelligent sampling finds goal node or a node in goal region (close neighbourhood of goal) (see steps 13–14 in Algorithm 3). When bounded sampling completes a single scan of  $C_{Region}$  (see steps 15 and 16 in Algorithm 3) and path is not found in the first scan, then  $D_{scale}$  is increased gradually to increase  $C_{Region}$  until a path is found (see step 16 in Algorithm 3), as shown in Fig. 3c, d.  $D_{scale}$  can grow  $C_{Region}$  up to the whole environment map in extreme complex scenario such as a maze. Exploring the obstacle-free space in  $C_{Region}$  (see steps 6–12 in Algorithm 3) makes sampling highly biased towards goal as shown in Fig. 3c. Once an initial path is searched within  $C_{Region}$ , a cost function returns the cost of path in terms of Euclidean distance defined by

$$\Delta d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{4}$$



**Fig. 3** Process of proposed algorithm RRT\*-adjustable bounds



## 4.2 Path optimization

Once an initial path is identified, the path is further optimized using three strategies: a) path concentrated sampling, b) node rejection technique, and c) path pruning.

**Concentrated sampling** Firstly, connectivity region is adjusted in close vicinity of initial path (see step 14 in Algorithm 3). This new reduced and bounded area is very narrow  $C_{Region}$  in the vicinity of path way-points. It is generated by redefining the scale of  $C_{Region}$  around initial path as shown in Fig. 3f (see step 14 in Algorithm 3). As a result of this phenomenon frequent rewiring operations occur in the vicinity of initial path, which benefit to decrease path cost very rapidly. Each time when new path is found,  $C_{Region}$  is redefined or readjusted again along new path.

**Node rejection** Second important factor of path optimization is node rejection technique. Node rejection works on the simple principle of avoiding high-cost nodes. According to it, a newly sampled node is rejected to insert in tree, if sum of its cost and its distance from goal is greater than current path cost. Such a node could not be useful to optimize the path.

Thus, tree cost is maintained by expanding tree with useful nodes only. Path gradually improves with the combined effect of concentrated sampling and node rejection technique until the end of defined iterations.

**Path pruning** In the final phase of path optimization, a global pruning process [30] is performed to further shorten the path (see step 17 in Algorithm 3).

Figure 3 shows the top level sequence of major steps in proposed approach. Figure 3a shows an indoor environment map. Figure 3b shows occupancy map generated along with start and goal position shown. Then connectivity region  $C_{Region}$  is defined by Eq. 3. When path is not searched during first scan of  $C_{Region}$  so it is enhanced in double as shown in Fig. 3d. Figure 3e shows initial path found with a path cost of 165 m and redefinition of connectivity region along initial path. Figure 3f shows the intermediate result of path optimization process. Sampling concentrated in the nearby area around path is performed along with node rejection technique, which resulted in near-optimal path with a convergence to cost of 136 m until completion of iterations, as shown in Fig. 3g.

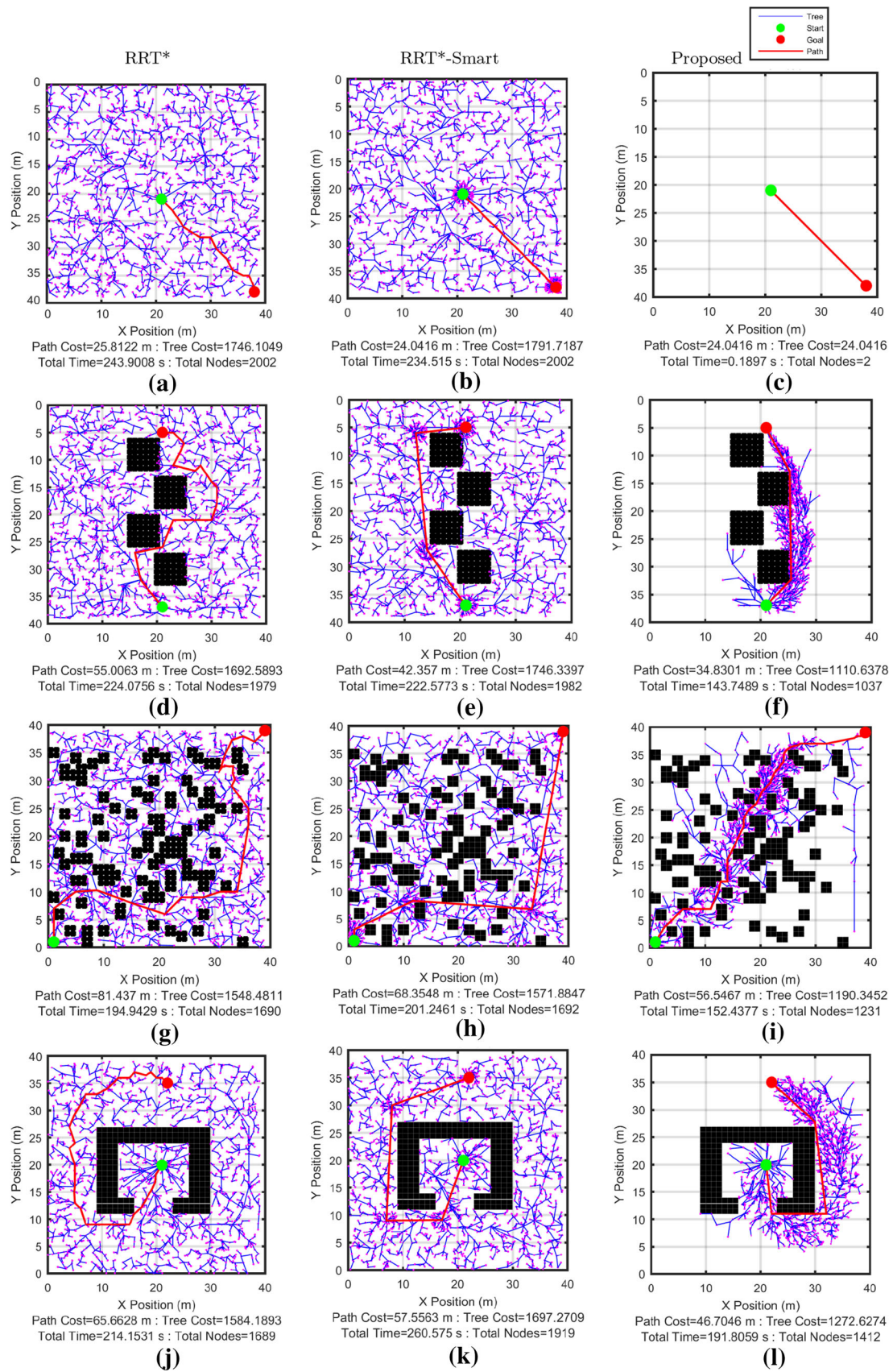
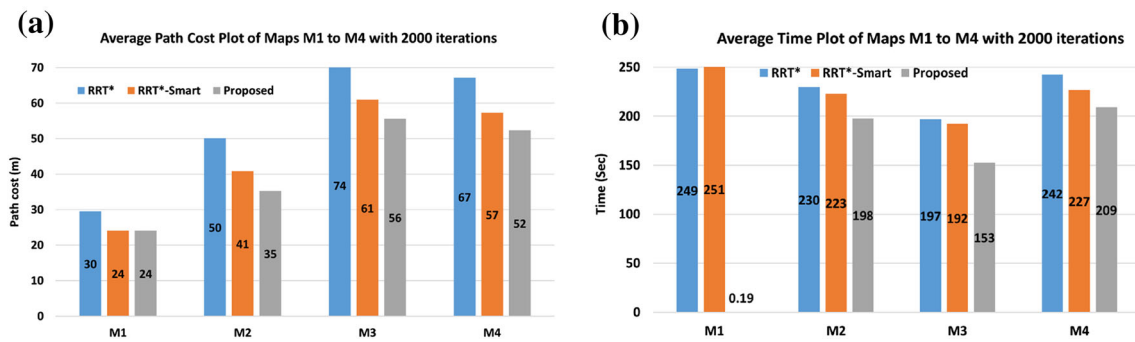


Fig. 4 Results comparison of structured maps M1 to M4



**Fig. 5** Plots for maps M1 to M4; **a** average execution time, **b** average path cost

## 5 Experimental results and discussion

This section demonstrates experiments of proposed approach in different environments, each with different scenario of obstacles. The numerical analysis plots of RRT\* [20], RRT\*-Smart [30] and RRT\*-AB are also presented. The algorithms are implemented using 64-bit version of MATLAB 15 and tested on a PC with an Intel i3-4010U@1.70 GHz CPU and 4GB internal RAM. The operating system is 64-bit Windows 8 Enterprise. Experiments are performed with different environment maps referred as M1, M2, M3, M4, M5, M6, and M7 to verify the robustness of proposed approach. Statistics of experiments along with simulation results are recorded to provide graphical and numerical comparative analysis with previous approaches. Considering the stochastic nature of the methods, each experiment set was conducted twenty times for each case of environment map and for each approach using 4000 iterations for maps M5 and M6 and 2000 for all other maps, respectively. Initial and goal points are chosen on extreme distance from one end to another end in all the maps.

### 5.1 Structured cluttered environment

Map M1 is the scenario of obstacle-free environment shown in Fig. 4a–c. Map M2 and M3 are scenario of environment cluttered with different obstacles density, as shown in Fig. 4d–i, respectively. M4 represents an in-bound obstacle shape shown in Fig. 4j–l. In case of map M1, it is obvious from Fig. 4a–c that previous approaches explore whole search space and generate highly dense tree with large number of nodes, which are not worthy to contribute in final path. On the contrary, proposed approach finds out the collision-free route between  $z_{init}$  and  $z_{goal}$  without moving towards deep exploration of configuration search space. Similarly, for the cases of M2, M3, and M4 both previous approaches have expanded tree in whole space with high tree cost and large number of nodes as compared to the proposed approach and has also generated larger paths in more time than proposed approach.

Results in Fig. 4 clearly exhibit that the proposed approach explores targeted environment region intelligently and adds most of the meaningful nodes in tree to explore the path. Further, it has generated shorter paths with less tree cost and has also converged to minimum path cost in less time than both other approaches. It is also demonstrated from comparison of average path cost and average execution time plots of twenty test runs of all approaches for each map in Fig. 5.

### 5.2 Narrow and complex environment

M5 is a scenario of complex narrow passage, shown in Fig. 6a–c, whereas M6 is a case of maze passage shown in Fig. 6d–f. Their results show the flexibility of  $C_{Region}$  to adjust its bounds even for highly complex cluttered environment. Average execution time plot and path cost plot of narrow passage case M5 in Fig. 7 demonstrate that proposed approach generated shorter path in less time than RRT\* and RRT\*-Smart. Average path cost plot and execution time plot of maze case M6 in Fig. 7 demonstrate that RRT\* has less execution time than RRT\*-Smart and proposed approach. However, RRT\* has generated longer path than both RRT\*-Smart and proposed approach. The main factor of more time and better path by RRT\*-Smart and proposed approach is that they both adapt intelligent sampling and path optimization techniques as compared to RRT\*. For maze case, the proposed approach had to expand its  $C_{Region}$  for second scan. However, it still converged to better path than both previous approaches.

### 5.3 Unstructured indoor environment

Experiments on unstructured indoor environment are also conducted. For this purpose, map M7 is adapted from Intel research lab datasets [32]. It is visible from comparison of paths in Fig. 8a–c and average execution time and path cost plots in Fig. 9a that proposed approach converges to optimal solution quickly in less time. Average value plots in Fig. 9a show proposed approach time and memory efficient than previous approaches. Since the process of convergence



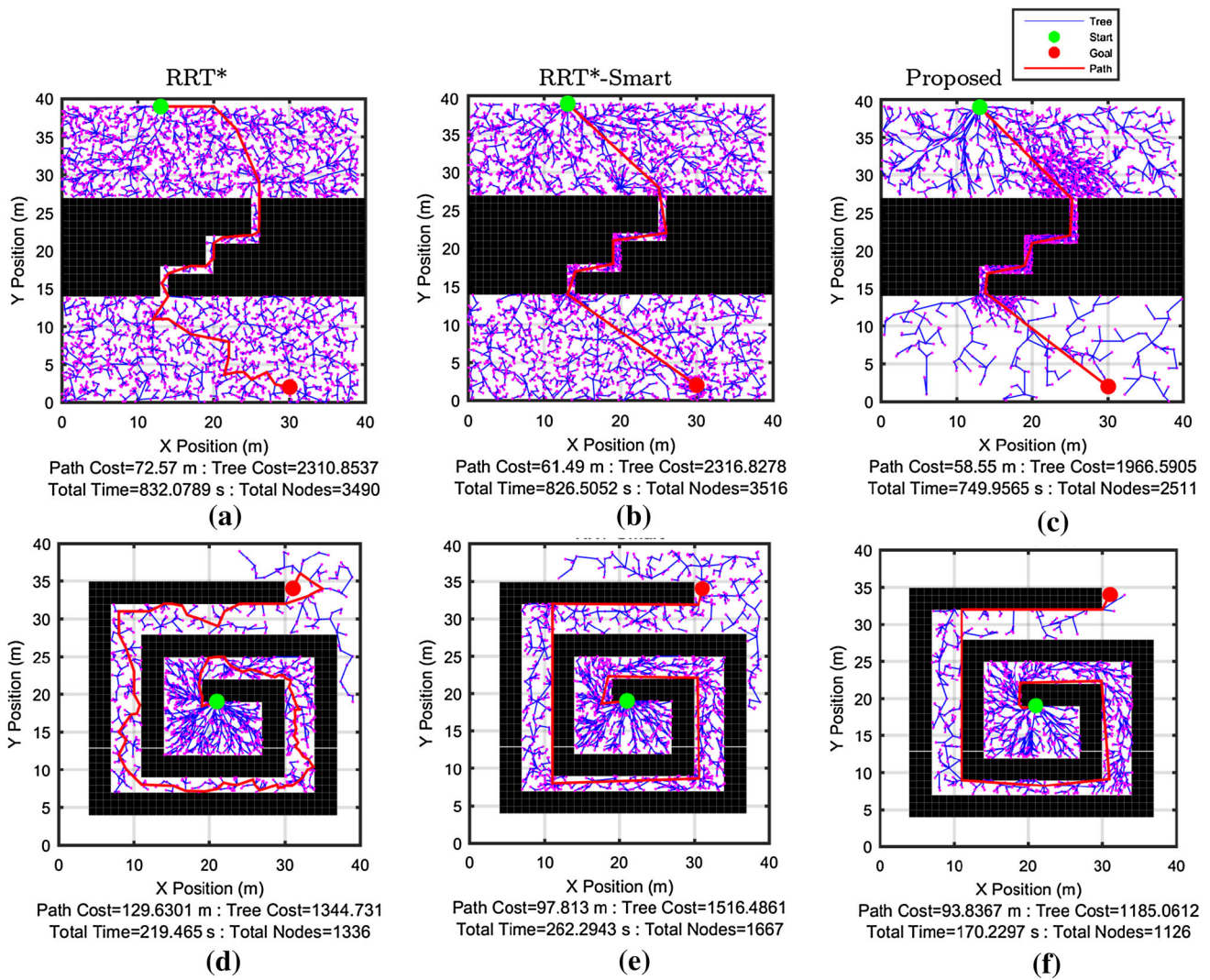


Fig. 6 Results comparison of cases M5 and M6

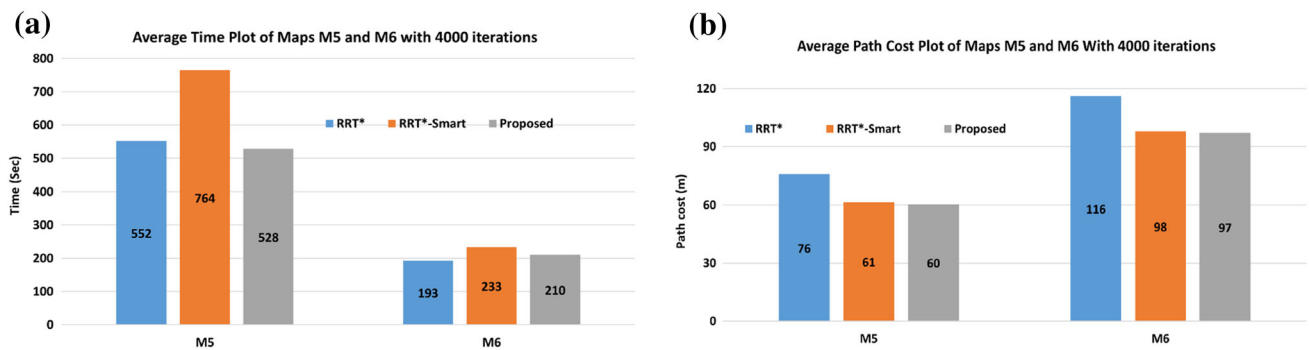


Fig. 7 Plots for maps M5 and M6; **a** average execution time, **b** average path cost

to the optimal solution starts after finding the initial feasible solution, therefore, convergence rate is calculated after computation of initial path. Figure 9b shows convergence rate comparison of RRT\*, RRT\*-Smart and the proposed approach RRT\*-AB for 20 different trials of unstructured

cluttered environment map M7 using 2000 iterations. Let the initial feasible path, denoted by  $\sigma_{init}$ , is computed in  $t_{init}$  time with  $c_{init}$  path cost while the optimal path solution, denoted as  $\sigma_{(*)}$ , is computed in  $t_*$  time with  $c_*$  path cost. Then the convergence rate is defined as

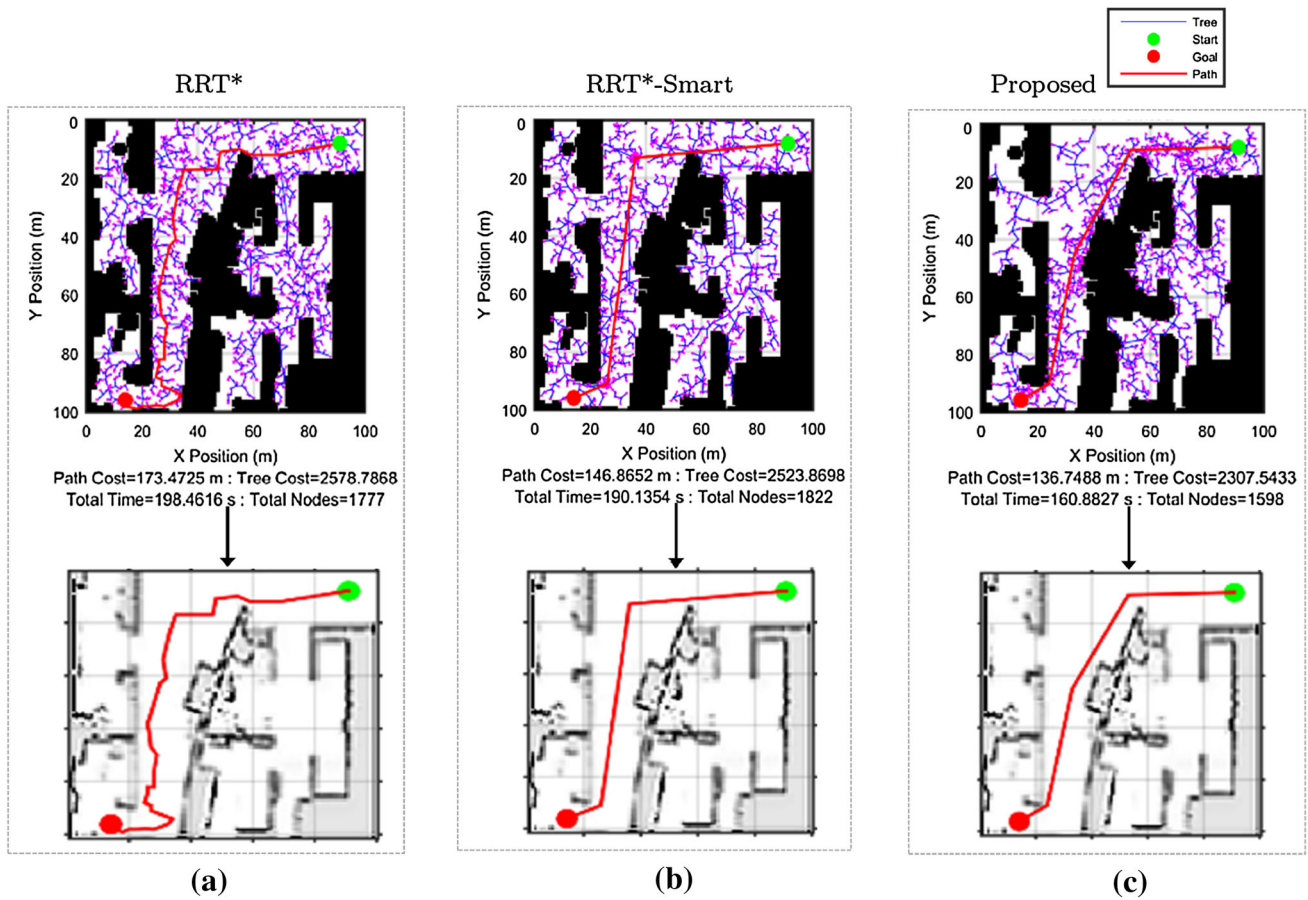


Fig. 8 Results comparison of unstructured map M7

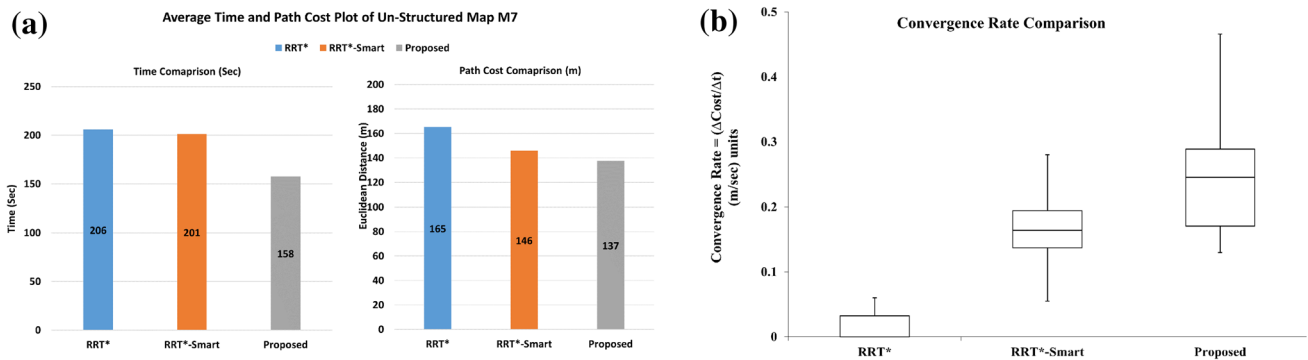


Fig. 9 Plots of unstructured map M7; a average execution time (left) and average path cost (right), b convergence rate

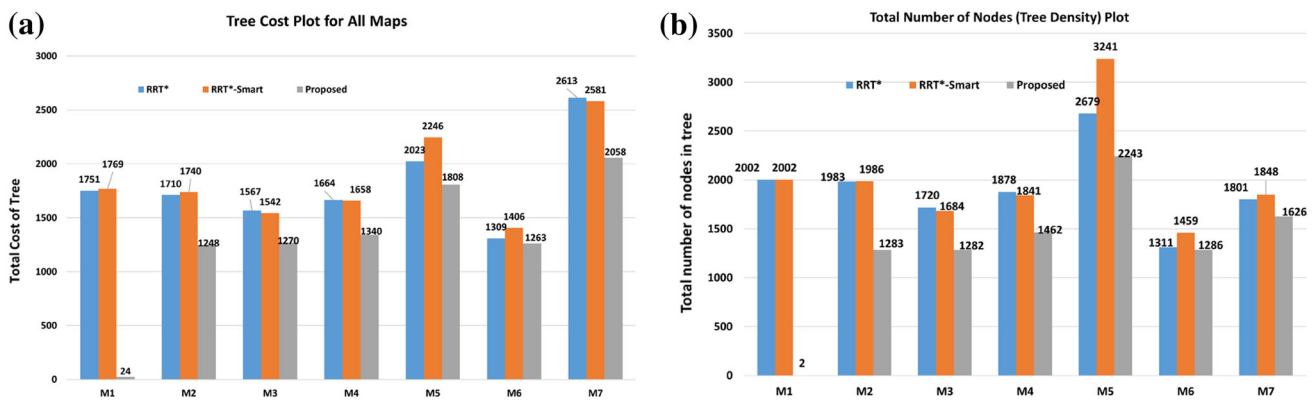
$$\frac{(c_{init} - c_*)}{(t_* - t_{init})} \tag{5}$$

It is clear from the plot that convergence rate of proposed approach RRT\*-AB is highest, followed by RRT\*-Smart and RRT\*. It shows the fact that once an initial path is found, the proposed approach improves it rapidly to converge it to optimal solution than both other approaches.

To provide the comparison of memory requirements and tree density, average total number of nodes in tree and total

tree cost for all maps by each approach are shown in Fig. 10a, b. It is evident from Fig. 10 that proposed approach finds path using minimum tree cost and it generates less dense, but more promising tree than other approaches resulting in less memory requirements.

Proposed approach RRT\*-AB performs better than RRT\* and RRT\*-Smart [30]. RRT\*-Smart converges quickly than RRT\*; however, it is dependent upon biasing ratio constant. Biasing constant defines the ratio of direct sampling to intelligent sampling using a computation process (see step 4 in



**Fig. 10** Plots indicating memory consumption by all maps; **a** average total number of nodes in tree, **b** average tree cost

Algorithm 2). Secondly its intelligent sampling is limited to beacon points. Therefore, it does not completely switches to intelligent sampling and uses direct random sampling also till the end of iterations. Hence, it partially limits the search space, as a result increases tree density with un-suitable nodes during optimization process. Further, its pruning is executed each time when a new path is found, which also delays execution time. On contrary, proposed approach uses intelligent sampling throughout the process, reduces search space effectively and uses a three-step optimization process. Two steps of optimization, i.e., concentrated sampling and node rejection in readjusted  $C_{Region}$  are performed till end of the iterations and pruning is used once in the end.

Informed RRT\* [31] also limits the search space efficiently but in an hyperellipsoid area and after getting initial path. It is specifically designed to deal with extremely narrow passages. Therefore, it fails to perform well in cluttered environment because its hyperellipsoid could not be reduced effectively to converge the path [33]. On contrary, formation of limiting search space in the proposed approach is two stage. Initially, proposed approach limits search space using goal-biased heuristics, which could be increased in case of complex environment. Once an initial path is found, then it adjusts a very narrow limited region according to the way-points of found path, as shown in Fig. 3c–e. As a result, proposed approach generates fast and meaningful rewiring operations to improve the path. Moreover, proposed approach can be used in narrow passages, complex maze, and cluttered environments as demonstrated by results. Similarly, Informed RRT\* does not use any path optimization technique, whereas three-step optimization process in proposed approach not only accelerates the convergence but also removes redundant nodes in the path.

The improved results of proposed approach are based on the three features. First, goal-biased intelligent sampling is performed in limited region of environment map, which enables to find path quickly. Second, when path is found,

connectivity region is readjusted according to the path. A concentrated sampling is performed in new connectivity region, i.e., only in close vicinity of the nodes forming way-points of path, which causes meaningful rewiring operations to improve path cost. Whereas in other approaches, a lot of nodes are rewired in tree far away from the found path, giving no benefit to improve path cost. Since, rewiring in tree also adds time overhead, so it is highly desired that only beneficial rewiring should take place once initial path is found. Third, node rejection technique restricts tree to increase unnecessarily by inserting only promising nodes. Moreover, it also effects execution time for near neighbour search and rewiring operations, as in RRT\* these operations take much time if tree has large number of nodes. Further, improvement in time efficiency can be achieved by adapting near neighbour search techniques such as *kd*-tree or quad tree [34].

## 6 Conclusion and future directions

In this paper, a novel path finding algorithm called RRT\*-adjustable bounds (RRT\*-AB) is proposed. Experimental results show that proposed algorithm (1) generates promising tree by intelligently exploring the search space using novel schemes, i.e., connectivity region, goal-biased bounded sampling, concentrated sampling and node rejection; (2) converges to optimal solution quickly; (3) generates shorter paths; (4) consumes less memory; (5) also deals with narrow and complex environment formations along with cluttered environment. Numerical analysis of simulation results is presented to support the theoretical analysis. Moreover, convergence rate is also identified for simulations to compare the improved performance with other RRT\* approaches. Future directions are the advancement of this approach to generate feasible path in un-known 3D environment for car-like mobile robots.



**Acknowledgements** This study is jointly supported by the research Grant of Higher Education Commission (HEC) of Pakistan (No. 20-2359/NRPU/R&D/HEC/12-6779) and by a Project of Korean Government (10073166).

## References

- Lan X, Di Cairano S (2015) Continuous curvature path planning for autonomous vehicle maneuvers using RRT\*. In: Paper presented at the European control conference (ECC)
- Zhang X, Chen J, Xin B (2014) Path planning for unmanned aerial vehicles in surveillance tasks under wind fields. *J Cent South Univ* 21(8):3079–3091. doi:10.1007/s11771-014-2279-7
- Lau G, Liu HHT (2013) Real-time path planning algorithm for autonomous border patrol: design, simulation, and experimentation. *J Intell Robot Syst* 75(3–4):517–539. doi:10.1007/s10846-013-9841-7
- Ahmidi N, Hager GD, Ishii L, Gallia GL, Ishii M (2012) Robotic path planning for surgeon skill evaluation in minimally-invasive sinus surgery. In: Paper presented at international conference on medical image computing and computer-assisted intervention
- LaValle SM (2006) Planning algorithms. Cambridge University Press, Cambridge
- Elbanhawi M, Simic M (2014) Sampling-based robot motion planning: a review survey. *IEEE Access* 2:56–77
- Hwang YK (1992) Gross motion planning—a survey. *ACM Comput Surv* 24(3):219–291
- Goerzen C, Kong Z, Mettler B (2009) A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *J Intell Robot Syst* 57(1–4):65–100. doi:10.1007/s10846-009-9383-1
- Hart PE (1968) A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans Syst Sci Cybern* 4(2):100–107
- Daniel K, Nash A, Koenig S (2010) Theta any-angle path planning on grids. *J Artif Intell Res* 39:533–579
- Wang X, Hou Z, Lv F, Tan M, Wang Y (2014) Mobile robots modular navigation controller using spiking neural networks. *Neurocomputing* 134:230–238. doi:10.1016/j.neucom.2013.07.055
- Zhu DQ, Li WC, Yan MZ, Yang SX (2014) The path planning of AUV based on D-S information fusion map building and bio-inspired neural network in unknown dynamic environment. *Int J Adv Robot Syst*. doi:10.5772/56346
- Liang JH, Lee CH (2015) Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm. *Adv Eng Softw* 79:47–56. doi:10.1016/j.advengsoft.2014.09.006
- Arora T, Gigras Y, Arora V (2014) Robotic path planning using genetic algorithm in dynamic environment. *Int J Comput Appl* 89(11):8–12
- Kala R (2012) Multi-robot path planning using co-evolutionary genetic programming. *Expert Syst Appl* 39(3):3817–3831. doi:10.1016/j.eswa.2011.09.090
- Zhu WR, Duan HB (2014) Chaotic predator-prey biogeography-based optimization approach for UCAV path planning. *Aerosp Sci Technol* 32(1):153–161. doi:10.1016/j.ast.2013.11.003
- Nosrati M, Karimi R, Hasanvand HA (2012) Investigation of the \* (Star) search algorithms characteristics methods and approaches. *World Appl Program* 2(4):251–256
- Aissa O, Xu H, Zhao G (2009) Survey and the relative issues on the path planning of mobile robot in rough terrain (Online)
- Noreen I, Khan A, Habib Z (2016) Optimal path planning using RRT\* based approaches: a survey and future directions. *Int J Adv Comput Sci Appl (IJACSA)* 7:97–107
- Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *Int J Robot Res* 30(7):846–894. doi:10.1177/0278364911406761
- Karaman S, Walter M, Perez A, Frazzoli E, Teller S (2011) Anytime motion planning using the RRT. In: Paper presented at the IEEE international conference on robotics and automation (ICRA)
- Ju T, Liu S, Yang J, Sun D (2014) Rapidly exploring random tree algorithm-based path planning for robot-aided optical manipulation of biological cells. *IEEE Trans Autom Sci Eng* 11(3):649–657
- Yang K (2011) Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments. *Int J Control Autom Syst* 9(4):750–758. doi:10.1007/s12555-011-0417-7
- LaValle SM (1998) Rapidly-exploring random trees: a new tool for path planning
- Kavraki LE, Svestka P, Latombe J-C, Overmars M (1996) Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans Robot Autom* 12(4):566–580
- Karaman S (2013) Sampling based optimal motion planning for non-holonomic dynamical systems. In: Paper presented at the IEEE international conference on robotics and automation (ICRA)
- Qureshi AH, Ayaz Y (2015) Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. *Robot Auton Syst* 68:1–11. doi:10.1016/j.robot.2015.02.007
- Noreen I, Khan A, Habib Z (2016) A comparison of RRT, RRT\* and RRT\*-smart path planning algorithms. *Int J Comput Sci Netw Secur* 16:20–27
- Loeve JW (2012) Finding time-optimal trajectories for the resonating arm using the RRT\* algorithm. Delft University of Technology, Delft
- Nasir J, Islam F, Malik U, Ayaz Y, Hasan O, Khan M, Saeed M (2013) RRT\*-SMART: a rapid convergence implementation of RRT\*. *Int J Adv Robot Syst* 10:1–12. doi:10.5772/56718
- Gammell JD, Srinivasa SS, Barfoot T D (2014) Informed RRT\*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In: Paper presented at the IEEE RSJ international conference on intelligent robots and systems (IROS), Chicago
- Robotics Datasets University of Freiberg. <http://www2.informatik.uni-freiburg.de/~stachnis/datasets.html>. Accessed 2016
- Choudhury S, Gammell JD, Barfoot TD, Srinivasa SS, Scherer S (2016) Regionally accelerated batch informed trees (rabit\*): a framework to integrate local information into optimal path planning. In: Paper presented at the IEEE international conference on robotics and automation (ICRA), Stockholm, Sweden
- Svenstrup M, Bak T, Andersen HJ (2011) Minimising computational complexity of the RRT algorithm—a practical approach. In: Paper presented at the IEEE international conference on robotics and automation, Shanghai, China