# Rapidly-exploring Random Graphs: Motion Planning of Multiple Mobile Robots

RAHUL KALA

*School of Cybernetics, School of Systems Engineering, University of Reading,*

*Whiteknights, Reading, UK, RG6 6AY, rkala001@gmail.com*

**Abstract**

Rapidly Exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM) are sampling based techniques being extensively used for robot path planning. In this paper the tree structure of the RRT is generalized to a graph structure which enables a greater exploration. Exploration takes place simultaneously from multiple points in the map, all explorations fusing at multiple points producing well-connected graph architecture. Initially, in a typical RRT manner, the search algorithm attempts to reach the goal by expansions, and thereafter furtherer areas are explored. With some additional computation cost, as compared to RRT with a single robot, the results can be significantly improved. The so formed graph is similar to roadmap produced by PRM. However as compared to PRM, the proposed algorithm has a more judicious search strategy and is adaptable to the number of nodes as a parameter. Experimental results are shown with multiple robots planned using prioritization scheme. Results show the betterment of the proposed algorithm as compared to RRT and PRM techniques.

*Keywords: Rapidly exploring random trees, probabilistic roadmaps, robot path planning, multi-robot systems.*

## 1. INTRODUCTION

Planning the path of multiple robots deals with construction of a feasible trajectory of all the robots such that all the robots reach their goals without any collision with each other or a static obstacle. Dynamic environments necessitate the planning algorithm to be computationally less expensive so as to enable the robots to quickly react to the changes in the environment. Planning may be centralized or decentralized [1]. Centralized planning techniques construct a complex configuration space consisting of all the robots and attempt to generate an optimal plan. Decentralized approaches plan each robot separately in their own configuration space. A coordination technique may be used to avoid collisions between the robots. Prioritization [2-3] is a commonly used coordination technique.

Due to the nature of the problem, sampling based techniques are increasingly being used where some samples are taken to represent the entire configuration space. Sampling may hence lead to loss of completeness and optimality with the advantage of computational time. The two frequently used sampling based techniques are Rapidly-exploring Random Trees (RRT) [4-5] and Probabilistic Road Maps (PRM) [6-7].

In RRT [4-5] the search begins with the source as the root of a search tree. At every iteration the algorithm generates a random point in the configuration space, searches for the closest point in the tree and extends this point to the random sample by a magnitude of *stepsize*. The algorithm stops when the exploration results in reaching the goal. Algorithm may be made biased to explore towards the goal. RRT-Connect algorithm checks if travelling straight in the line of expanded point reaches the goal, in which case the algorithm terminates. Bi-directional variant of RRT expands two trees, one each from the source and goal, and terminates when the two trees meet. RRTs are computationally efficient, but sub-optimal. Sub-optimality here may be global or local. Global optimality indicates the strategy to avoid obstacles (whether to go from left of an obstacle, or right of an obstacle, or in-between obstacles) while local optimality indicates distances to maintain from obstacles. RRT paths may be passed through local optimization which is computationally expensive. In a multi-robot scenario, each robot may have its own RRT instance which makes the search process time consuming.

PRM [6-7] is another widely used technique. In this technique a number of random points are sampled out of the configuration space and a local planning algorithm is used to determine which states are connected to which other states, producing a graph structure called roadmap. In Lazy PRM [8] collision checking is performed using a coarser to finer strategy which is more efficient. The construction of the roadmap is a computationally expensive step. Samples are taken from the entire configuration space, which may consist of a large number of areas which are visibly unnecessary for any robot to go from its source to its goal. This is especially true when a small number of robots navigate in sections of dynamic maps. Unlike RRT, PRM is less likely to miss being near the global optima.

The key contributions of the work are (a) A graph variant of RRT is proposed which is a completely new domain of thought. (b) The proposed algorithm can have multi-directional multi-strategized exploration with multiple initiation points. (c) In terms of path length, the proposed algorithm is better in terms of both global and local optimality as compared to RRT and its variants. (d) While the proposed algorithm is computationally more expensive as compared to RRT (and variants) with every robot’s path computed independently by an independent processor, the total processing required is smaller in case of the proposed algorithm. (c) The proposed algorithm is better than PRM and similar roadmap approaches in terms of adaptability to the number of nodes as a parameter, and tradeoff between path lengths to the number of nodes.

## 2. RELATED WORKS

Raveh et al. [9] noted the problem of optimality of the RRT and proposed running multiple instances of the algorithm to produce multiple trees or paths to the goal. The authors then used a dynamic programming based algorithm for merging the best sub-paths of the various instances to produce the best overall path. In another work Yao and Gupta [10] used roadmap consisting of sensors which can communicate with each other. Planning was local and distributed along the network graph where each robot queried the nearest sensor for deciding the next step. A generalized version of the RRT and PRM algorithm is displayed in the work of Chakravorty and Kumar [11]. A connection between any two samples or points is done on the basis of Monte Carlo simulations. Carpin and Pagello [12] studied the aspects of decentralized motion planning of multiple robots, with respect to computational time and optimality. The authors used priority based coordination with space time graph approach for planning.

Jaillet et al. [13] integrated costs with the configurations space and used a transition test to allow or disallow the expansion of a node to produce a new node. Urmson and Simmons [14] talk about heuristics in the selection of nodes. This allows expansion along better areas of the map, making generated path near-optimal. Kalisiak and van de Panne [15] proposed a variant of RRT called RRT-Blossom. In this method a flood fill like mechanism was adopted for planning. The expansions were biased towards goal, resulting in increasing explorations towards goal from source. Strandberg [16] floated the idea of local trees which could be initiated from various parts of the map. These trees captured local information about the map by continuous expansions, starting from where they were initiated. Ferguson and Stenz [17] proposed another variant called Anytime RRTs using similar heuristics for selection and expansion of nodes.

Sampling based approaches face narrow corridor problem wherein the algorithm is unable to compute path of the robot from source to goal when the path passes through a narrow corridor. A solution called retraction based RRT was presented by Zhang and Manocha [18] in which a sample generated within obstacle was promoted to its nearest neighbor in free space. Clark [19] used single query based PRM algorithm for planning. The solution consisted of software, communication, planning, and control modules. Connections could be build or get broken as the environment changed.

Kala [20] solved the problem of planning of multiple robots using co-evolutionary genetic programming. The solution consisted of multiple instances of slave genetic programming for each robot, while master genetic algorithm computed the optimal combination of plan. Robots had common memory for sharing optimal sub-plans. Kala et al. [21] used an iterative solution to the problem of path planning of robot using A* algorithm. The algorithm proceeded from coarser to finer level which was executed by a tree-based map representation. At every iteration cells along computed paths were broken down into sub-cells representing a finer map. Gayle et al. [22, 23] introduced the concept of reactive deforming map where the map could adapt to changing situations. The movement of robot was done based on the internal forces that represent forces of the roadmap; and the external forces or the forces of the dynamic obstacles which become dominant if they come into the vicinity of the robot suddenly.

## 3. ALGORITHM

The problem is to find a collision-free trajectory $\tau_i(t)$ for a number of robots, each starting from a given source ($\tau_i(0) = S_i$) to a given goal ($\tau_i(T_i) = G_i$), where the journey completes in time $T_i$. This is a multi-robot single query problem. The robots must not collide with any static obstacle at any time i.e. $\tau_i(t) \in \zeta^{free}_i \;\; \forall\, i,\; 0 \le t \le T_i$. Here $\zeta^{free}_i$ denotes the free configuration space of the robot $i$. The robots must also not collide with each other in their way i.e. $\tau_i(t) \;\in \zeta^{free}_i - C_j(\tau_j(t)) \;\; \forall\, i, j \neq i,\; 0 \le t \le T_i$. It is assumed the robots disappear as soon as they reach their goal. The function $C_j(\tau_j(t))$ returns set of points in the configuration space $\zeta$ which are occupied by robot $j$ when placed at position $\tau_j(t)$. The travelling speed of the robot $v_i$ is constant.

The objective is to minimize the cumulative travel time $\sum_i T_i$. Additionally the robots may prefer to maintain some minimal separation *smin* from obstacles and other robots, which means preferably $\tau_i(t) \pm smin \in \zeta^{free}_i - C_j(\tau_j(t)) \;\; \forall\, i, j \neq i,\; 0 \le t \le T_i$. The robots may have non-holonomic constraints as per their own modeling, which needs to be accounted for.

The overall solution is summarized by figure 1. The algorithm first iteratively produces a graph by constantly adding nodes and edges. This graph is then used for the planning of the mobile robots. The various aspects of the algorithm are summarized in the subsequent sections.

### 3.1. Rapidly Exploring Random Graphs

The proposed algorithm maintains a graph structure called Rapidly-exploring Random Graphs (RRG). Each node of the graph may be connected to one or more nodes. The chief properties of the graph are:

(i) The complete graph may be interconnected, or may consist of disconnected sub-graphs.

(ii) Minimum distance between any two nodes is $r$ which avoids too many nodes being produced close to each other resulting in high computational costs with minimal increase in quality. $r$ is roughly of the order of *stepsize*. Here *stepsize* is the distance by which a node is expanded to produce a new node.

(iii) Each node has a *location* and a *color*. Location tells the position in the configuration space, while the *color* speaks about the origin of the node that is the initial node by continuous expansion of whose children the node is produced. Every child node takes the color of the *parent*, whose expansion produces the child.

(iv) Each node is connected to every node at a distance of less than 2*stepsize*, if the connection between then is collision free.

(v) Every node may be connected to multiple nodes of the same color signifying redundant connectivity (unlike RRT) between nodes. Hence expansion of a child from parent may also result in the child being connected to the parent's parent. Besides, a node may be connected to some nodes of different color with corresponding edges known as *bridges* between the colored sub-graphs.

(vi) It is possible to construct a *cover* around all nodes, which marks the area out of the entire configuration space which has been explored. No new node may easily be produced inside this cover. Hence when the cover includes entire free configuration space, the algorithm must stop. For most practical applications though it may be made to stop a lot early.

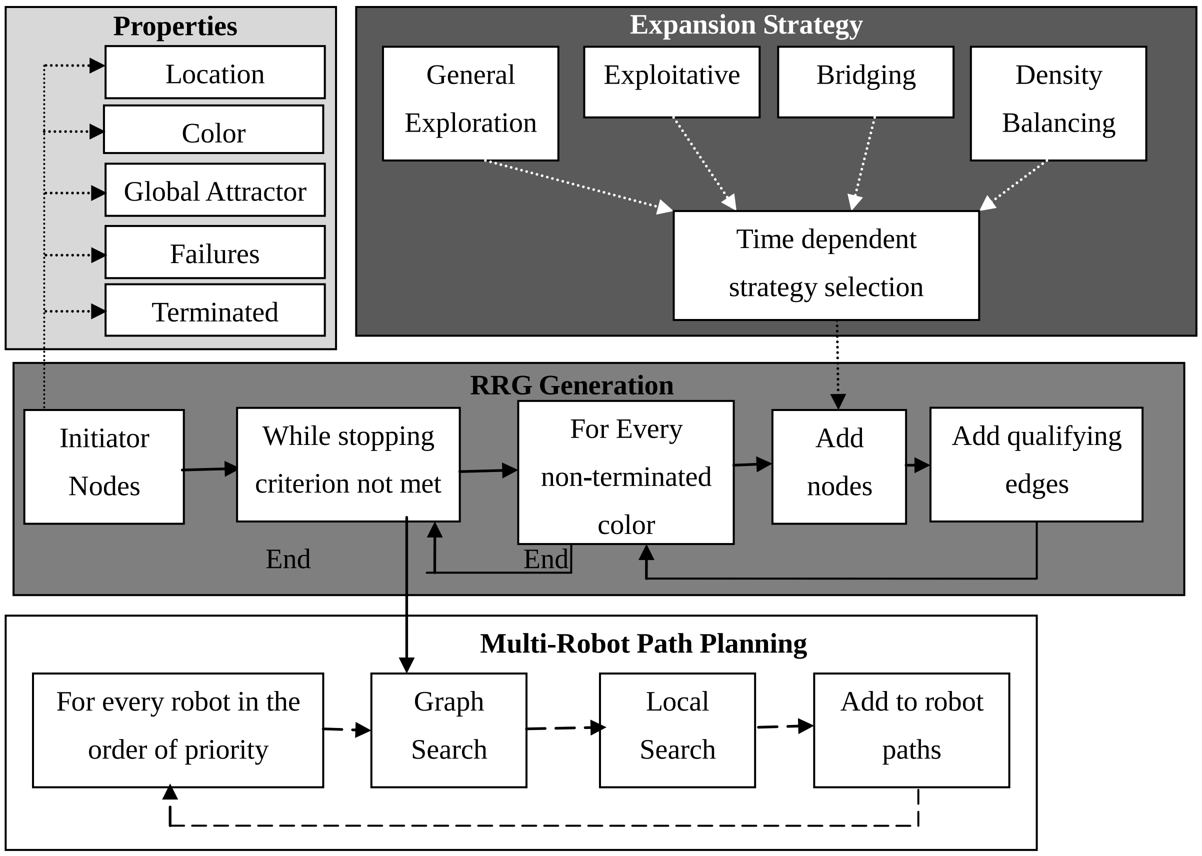


Figure 1 : Algorithm Framework.

Initially a set of *initiator nodes* are added to the graph, which are the only vertices in the graph. All initiator nodes are disconnected from each other. All points are given a distinct color. Search operation in random direction starting from the initiator point may be a costly exploration. Hence optionally a *global attractor* may be specified for every initiator point. The exploration, starting from specified point, proceeds so as to move towards its global attractor. The default choice is to take the set of sources and goals of all robots as the set of initiator points. Redundant points or points less than a distance of *r* are taken as one. The corresponding goals for sources and vice versa can be taken as the global attractors. Additionally if from any apriori analysis of the configuration space some interesting insights are already drawn, these may be added as initiator nodes. Say it is known that passing through a corridor is likely to be good for some robot; a point inside corridor may be specified as the initiator node. The search strategy is hence multi-directional in nature. The various terms are summarized in figure 2.

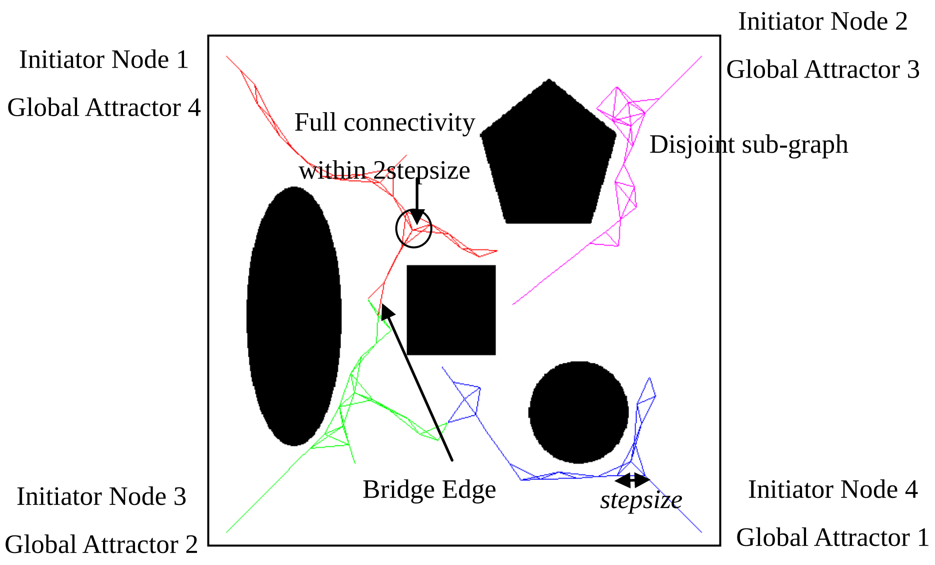


Figure 2 : A sample RRG.

### 3.2. Expansion Strategy

The graph (RRG) is iteratively produced by constantly adding nodes, and finding the nodes with which the newly produced node must be connected to via edges. We iterate through all the colors in the graph. A color here denotes a sub-graph which may or may not be connected to other sub-graphs depending upon the presence of bridge edges. For every color *c* a sample is generated ($S \in \zeta$), closest node is found in the colored sub-graph ($V$: $min_V(\| V - S \|$, $color(V)=c$, $V \in RRG$). The node $V$ is expanded towards $S$ by a magnitude of *stepsize* to produce a new node ($N$) which takes the same color as $V$ ($color(N) = c$). The node $N$ is added to the *RRG* only if no other node already lies in *RRG* close enough to $N$ and $N$ is collision free. After the addition of node, new edges need to be added. Edges are all undirected. $N$ is connected to any node $p$ by an edge $e$ if $\| N - p \| < 2stepsize$. Collision checking is additionally performed. The edge $e$ is called as a bridge edge if $color(N) \neq color(p)$.

The important issue here is selection of sample $S$ which forms the expansion strategy. Four ways are formulated which are.

(i) *General exploration:* The sample is randomly chosen out of the entire configuration space $\zeta$.

(ii) *Exploitative:* The global attractor corresponding to the color is chosen as the sample.

(iii) *Bridging:* A random sample *S* is drawn out of *RRG* such that $color(S) \neq c$. This is applied in pursuit of bridge edges and merging of sub-graphs

(iv) *Density balancing:* Attempt is to expand the graph towards areas which have not been appreciably covered yet. For this a coarser level hashmap is created with every block ($B \subset \zeta$) of points on the configuration space $\zeta$ corresponding to a cell of hashmap. The number of free states in *B* ($free(B)=count(b)$, $b \in B \cap \zeta^{free}$) are stored in the hashmap along with the number of nodes in *RRG* that lie within the block *B* ($noNodes(B)=count(r)$, $r \in RRG \cap B$). A cell of the hashmap is selected by tournament selection between two randomly chosen cells with weight of selection of cell *B* ($w(B)$) given by equation (1).

$$w(B) = free(B) - noNodes(B).stepsize^2 \quad (1)$$

The sample *S* is taken to be the mean of the block corresponding to winning cell *W*, that is $S=\overline{W}$ .

The four methods of expansion happen with probabilities $p_{explore}(t)$, $p_{exploit}(t)$, $p_{bridge}(t)$, *and* $p_{density}(t)$, such that

$$p_{explore}(t) + p_{exploit}(t) + p_{bridge}(t) + p_{density}(t) = 1 \quad (2)$$

These parameters are made to change with time (*t*). In the initial little iterations the intention may be more exploitative on reaching towards the corresponding global attractor with some exploration. This ensures that a solution is obtained quickly much like traditional RRT. Later attempt is to explore new areas and attempt to connect even to sub-graphs which lie far away, if no direct or indirect connection exists. This adds a deterministic adaptation to the system. In the present form the change of probabilities with time is as given by equation (3).

$$p_x(t) = p_x(0) – t/T.(p_x(T)- p_x(0)) \quad (3)$$

where $p_x(0)$ is the initial value, $p_x(T)$ is the final value, *T* is maximum time for which change in probability holds, *t* is time subjected to a maximum of *T* after which no change is made to these probabilities. At any time equation (2) must hold.

## 3.3. Exploration and Exploitation

Apart from general exploration and exploitation, an important concept designed is *bridging* in which the different colored sub-graphs try to connect to the neighboring sub-graphs. Imagine a general scenario with multiple robots. If the global attractor of an initiator point is near the location of another initiator point, the two colored sub-graphs are bound to meet and bridge. This is equivalent to the bidirectional search using RRT. Similar is the case when a global attractor corresponding to some robot lies close to the only path possible for some other robot. However for the other cases, it is likely that the sub-graphs no not bridge. The motivation is to add a little computation to RRT (for the case of a single robot) to significantly improve the path cost, which is modeled by bridging behavior. It requires little additional computation, but once merged opens a pool of options consisting of the sub-graph to which bridging has taken place along with all sub-graphs to which the bridged sub-graph was already bridged to. If individual sub-graphs are visualized as explorations of each robot being planned, bridging acts as a form of experience sharing between them.

When left to be executed indefinitely, RRG has the potential to eventually cover the entire configuration space available. However in the later iterations, random samples may be generated near the already explored areas, or at the goal which may have already been found or explored. If a large number of nodes are already present in the area, a new node may not find a place with the algorithm returning a failure to add node. Hence there are more failures than exploration to a rare state in the configuration space, which wastes computational time. An effort is hence made to avoid additional computation in expansion along explored areas, but to rather use it to go to new unexplored areas. Hence exploration should guide a graph for expanding towards most unvisited places in the configuration space, which are also nearest to a particular color as compared to any other color. The *density balancing* method carries a practical implementation of same concept.

As a result of these steps nodes are widespread. Unlike PRM, the algorithm is initially exploitative, and then explorative.

## 3.4. Post Processing and Multi-Robot Planning

The stopping criterion of the algorithm may be based on execution time, number of iterations, maximum number of nodes, or maximum number of failures in adding a node. As the algorithm proceeds certain colors may reach their threshold of expansion, wherein their covers either merge with the covers of other colors or limits of the configurations space. Failures to add nodes for every color are monitored, and if a set threshold is crossed termination for expansion of a particular color is set. Graph search is used to compute the trajectory of a robot. As per the scenario the best graph search algorithm is A* algorithm with Euclidean distance from goal heuristic function. Paths are post-processed using splines with small local optimization. This gives the final trajectory $\tau_i(t)$. For multi-robot systems planning is done in a priority based manner. The algorithm is given by

**CreateRRG(initiatorPoints)**

```
nodes ← initiatorPoints<Points, Colors, Global Attractors>
edges ← null
failures_c ← 0 for all colors c
terminate_c ← false for all colors c
while stopping criterion
        for every color c
                if terminate_c = false
                        success ← Expand(c)
                        if ┐ success failures_c  ← failures_c + 1
                if $failures_c$ > threshold $terminate_c$ = true
            end if
        end for
    end while
```

**Expand(Color c)**

```
    Generate sample S
    $V \leftarrow \min_V \| V - S \|$, colour(V) = c, $V \in$ RRG
    $N \leftarrow V$ + stepsize.$(S - V)/\| S - V\|$
    if $N \in \zeta^{free}_i$
        color(N) $\leftarrow$ c
        Nodes $\leftarrow$ Nodes $\cup$ N
        edges $\leftarrow$ edges $\cup$ (p, N) $\cup$ (N, p) $\forall$ p $\in$ RRG, p$\neq$N, $\|$ p $-$ N $\|$ < 2stepsize, p $\rightarrow$ N is
        collision free
        return true
    end
    return false
```

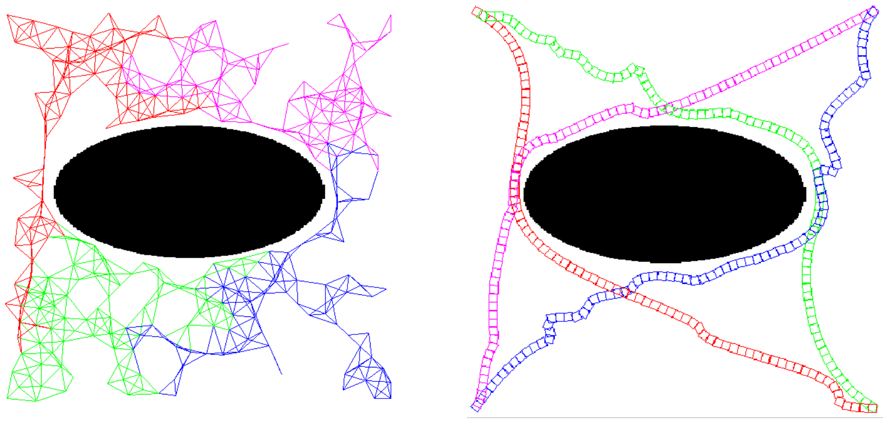
**GetTrajectory()**

```
    $\tau \leftarrow$ null
    for every robot i in decreasing order of priority
        Path $\leftarrow$ GraphSearch(RRG, $Source_i$, $Goal_i$, $\tau$)
        $\tau_i \leftarrow$ localSearch(Path, $\tau$)
        $\tau \leftarrow \tau \cup \tau_i$
    end for
```

## 4. Experimental Results

The approach was tested by means of simulations. A number of scenarios were generated from easy to more challenging ones. The simulations were performed on a system with Intel i3 processor (2.2 GHz) with 3 GB RAM. Two scenarios are discussed here. The first scenario consisted of a simple single obstacle in the middle of the paths of robots from source to goal. Four robots were generated at the corners which were supposed to travel to the other corner. The graph started the generation process from the four corners and continued till the four sub-graphs met and later the process continued exploring the entire robotic map. The generation of RRG is shown in figure 3(a). A video showing the expansion (until it was no longer possible to add any node) may be found at [24]. The path traced by the robots is shown in figure 3(b) (please refer [24] for more insights into the results).

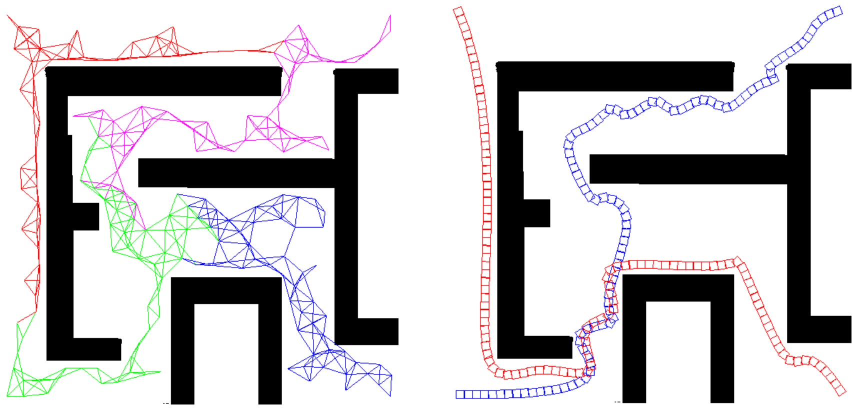
It is natural prolonged expansions would not be done in real time operations and generation would be terminated much before. The initial paths were more biased towards the diagonal and sub-optimal. The optimal paths are much deviated from the map diagonal. Hence enabling RRG generation for larger time is profitable.



(a) RRG generated after few execution steps (b) Path followed by the participating robots

Figure 3 : Simulation results for first scenario.

The second scenario consists of more number of obstacles forming a kind of maze through which two robots need to navigate across the opposite corners. Again four sub-graphs are generated from the four corners. There are two routes possible for every robot, out of which they need to collectively select the optimal one. These two routes have small differences in path length. The generated graph is shown in figure 4(a). The path traversed by robots is shown in figure 4(b). It may be seen that the robots could select the optimal path from source to goal, and travelled by the same.

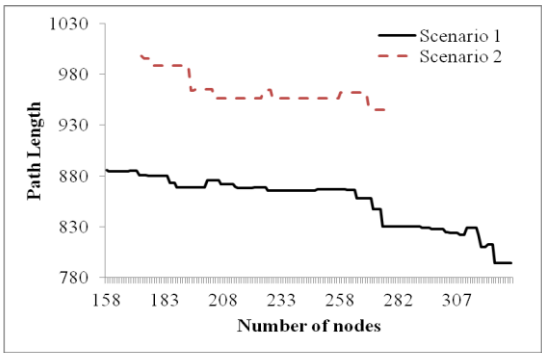


(a) RRG generated after few execution steps. (b) Path followed by the participating robots.

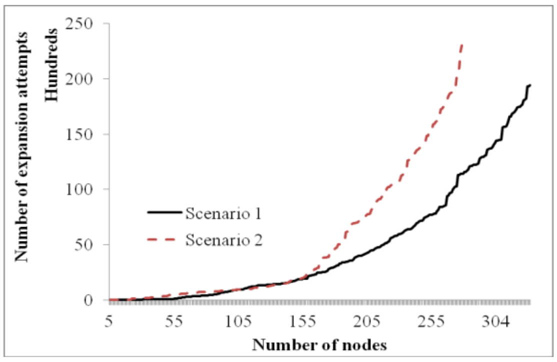
Figure 4 : Simulation results for second scenario.

The maximum number of nodes that can be generated in the RRG is limited (proportional to the area of the map without obstacles) beyond which it may not be able to add nodes. The average path length is plotted against different number of nodes in the RRG. No post processing was done. The results are shown in figure 5(a). Higher number of nodes implies a larger chance of having nodes close to the optimal path. The small increase in path length at certain places is due to the presence of multiple robots. Addition of some node may lead a robot to navigate by the added node, which might in turn be coming on the optimal path of another robot. The other robot now needs to take a longer route, thereby having sufficient separation from the earlier robot.

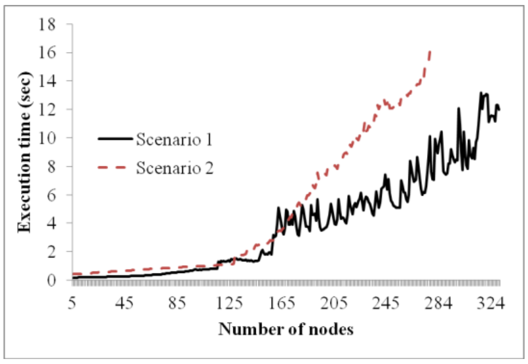
Number of expansions of the RRG is not the same as the number of nodes in it since many expansions result in infeasible nodes which are not added. As the number of nodes in RRG increases, it becomes increasingly difficult to expand feasible nodes as the overall map is already covered by the RRG. Figure 5(b) shows the number of expansions of RRG for various numbers of nodes, and figure 5(c) shows the execution time. The execution time includes the time spent in generation phase and the graph search phase. Normally 5% of deviation from the optimal path is acceptable if it comes with a big computational gain, which can be easily seen from figure 5. By about 2000-3000 (scenario 1) or about 4000-6000 expansion attempts (scenario 2), it is possible to solve for all the robots in about 3-4 seconds (scenario 1) or 5-6 seconds (scenario 2) getting paths within the optimality threshold. Obviously expansion of more number of nodes would result in more computation time which may also result in better paths, and vice versa. Whether further time can be invested or not depends upon the problem and the computational availability.



(a) Path length v/s number of nodes.



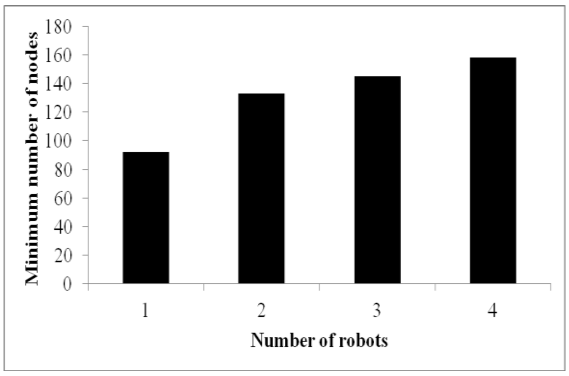
(b) Number of expansion attempts v/s number of nodes.



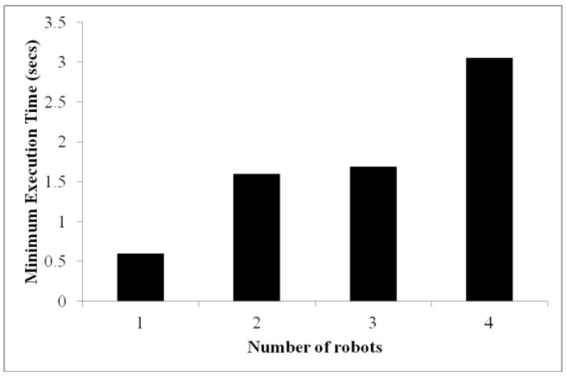
(c) Execution time v/s number of nodes.

Figure 5 : Analysis of number of nodes.

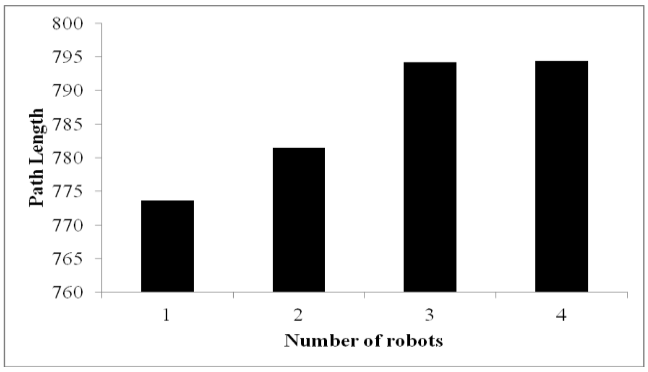
It is further important to study the effect of changing the number of robots. We take the first scenario and repeat the experiment for different number of robots. The minimum number of nodes required for generation of a feasible plan, the minimum execution time for generation of a feasible plan, and average path length is shown in figure 6. It may be clearly seen that the increase in minimum nodes and execution time is reasonably less as the robots are increased. It should be noted that for the stage of construction of the graph, the computational time increases with the number of robots till it reaches a worst case of PRM with moderate number of robots, after which it remains constant. In such a case graph generated by RRG is similar to the PRM roadmap. Hence RRG construction complexity (worst case performance) is invariant to the number of robots. Both PRM and RRG can be used with a more reactive and less deliberative roadmap based planning methods (e.g. [22]) rather than the present naive approach in case the number of robots is very high. In other words, the number of robots is not a limit for the algorithm in general. The path length slightly increases with number of robots as optimal plans of individual robots are prone to collisions with each other.



(a) Minimum number of nodes v/s number of robots.



(b) Minimum execution time v/s number of robots.



(c) Path length v/s number of robots.

Figure 6: Analysis of number of robots.

## 5. Comparisons with RRT and PRM

The same two scenarios as discussed in section 4 are taken for comparisons. In terms of path length RRG clearly outperformed RRTs by an extent of 8.2%. RRT travelled diagonally straight until it found the obstacle and then took a turn to reach the goal. This shows RRT is more locally sub-optimal as compared to RRG. In terms of total number of nodes as well RRG performed better. For RRT every robot generated its own tree which meant a lot of duplication across robots. RRT took 233.544% larger number of nodes for generation of a feasible plan, as compared to RRG.

For the second scenario as well, RRG clearly outperformed RRT. It generated 15.46% shorter path. RRT could not make out the shorter of the two possibilities for the robots as visible in the map. Hence at some run the longer alternative was followed, while at some other the shorter alternative was followed. This made the path length large. RRG could at all instances analyze the better path, and made the robot move by the same. This shows RRT is more globally sub-optimal as compared to RRG. The RRG took 136.199% lesser nodes for generation of a feasible plan. For two robots the RRT had to replicate the process of generation of tree twice which accounts for the difference.

Comparison of PRM with RRG is difficult to do as both RRG and PRM have different interpretation of the parameter of number of nodes. Having a larger number of nodes result in complete exploration in RRG which produces a graph which closely resembles a PRM roadmap. While comparing RRG with PRM using path length as a metric, a marginal betterment of RRG to PRM was observed. This may be attributed to the more directed nature of expansion of RRG. The major limitation of PRM was in the number of nodes as an algorithmic parameter. While specifying large number of nodes led to large execution times, the small number of nodes had no guarantee of generation of a feasible plan. It was not possible to determine this parameter by any way during the execution, which is not the case with RRG. This parameter can be made adaptive in RRG.

In experiments maps where the source and goal are located in a sub-map which forms a small part of the entire map (e.g. robot to go from one room to a close by room in a big hall), we may intuitively state the betterment of RRG over PRM.

## 6. Conclusions

The tradeoff between small computation time and high quality results is a major factor in the choice of algorithm. In this paper RRG was proposed as a means to add quality to RRT for the case of multiple robots. The algorithm is little more computationally expensive for single robot cases as compared to RRT. The proposed algorithm maintains a graph which grows with time to find a feasible travel plan for given number of robots. Experimental results show that the proposed algorithm can enable navigation of multiple robots in simple to complex maps in a time effective manner. The solutions generated are better in terms of quality of solutions both from perspectives of local optimality and global optimality as compared to RRT. Further from an analytical point of view, the

RRG is seen to be better than PRM.

By using an effective sampling technique, the proposed algorithm may be made suited for narrow corridor and similar problems. Similarly a variety of post processing techniques may be used to quickly improve the travel plan. It may still be better to eliminate post processing by adding its effect to the RRG generation. Heuristics may be experimented for assigning priorities to the robots, or different coordination techniques may be used.

## REFERENCES

[1] G. Sánchez-Ante and J. C. Latombe, Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems, in *Proc. IEEE International Conference on Robotics and Automation,* Washington, DC, pp. 2112 – 2119 (2002)

[2] M. Bennewitz, W. Burgard, and S. Thrun, Optimizing schedules for prioritized path planning of multi-robot systems, in *Proc. 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp. 271 – 276 (2001)

[3] M. Bennewitz, W. Burgard, and S. Thrun, Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots, *Rob. Auton. Syst.* **41**, 89–99 (2002)

[4] J. J. Kuffner and S. M. LaValle, RRT-connect: An efficient approach to single-query path planning, in *Proc. IEEE International Conference on Robotics and Automation vol. 2,* San Francisco, CA, pp. 995-1001 (2000)

[5] S. M. LaValle and J. J. Kuffner, Randomized kinodynamic planning, in *Proc. IEEE International Conference on Robotics and Automation*, Detroit, Michigan, pp. 473–479 (1999).

[6] L. E. Kavraki, M. N. Kolountzakis, and J. C. Latombe, Analysis of probabilistic roadmaps for path planning, *IEEE Trans. Rob. Autom.* **14**(1), 166-171 (1998)

[7] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Rob. Autom.* **12**(4), 566-580 (1996)

[8] R. Bohlin and L E. Kavraki, Path Planning Using Lazy PRM, in *Proc. 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 521-528 (2000)

[9] B. Raveh, A. Enosh, and D. Halperin, A Little More, a Lot Better: Improving Path Quality by a Path-Merging Algorithm, *IEEE Trans. Rob.* **27**(2), 365-371 (2011)

[10] Z. Yao and K. Gupta, Distributed Roadmaps for Robot Navigation in Sensor Networks, *IEEE Trans. Rob.* **27**(5), 997-1004 (2011)

[11] S. Chakravorty and S. Kumar, Generalized Sampling-Based Motion Planners, *IEEE Trans. Syst. Man Cybern. Part B: Cybern.* **41**(3), 855-866 (2011)

[12] S. Carpin and E. Pagello, An experimental study of distributed robot coordination, *Rob. Auton. Syst.* **57**, 129-133 (2009)

[13] L. Jaillet, J. Cortes, and T. Simeon, Transition-based RRT for path planning in continuous cost spaces, in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, pp. 2145-2150 (2008)

[14] C. Urmson, R. Simmons, Approaches for heuristically biasing RRT growth, in *Proc 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems vol.2*, Las Vegas, Nevada, pp. 1178- 1183 (2003)

[15] M. Kalisiak and M. van de Panne, RRT-blossom: RRT with a local flood-fill behavior, in *Proc. 2006 IEEE International Conference on Robotics and Automation*, Orlando, FL, pp. 1237-1242 (2006)

[16] M. Strandberg, Augmenting RRT-planners with local trees, in *Proc. IEEE International Conference on Robotics and Automation vol.4*, New Orleans, LA, pp. 3258- 3262 (2004)

[17] D. Ferguson and A. Stentz, Anytime RRTs, in *Proc. IEEE International Conference on Intelligent Robots and Systems*, Beijing, China, 5369-5375 (2006)

[18] L. Zhang and D. Manocha, An efficient retraction-based RRT planner, in *Proc. IEEE International Conference on Robotics and Automation*, Pasadena, CA, pp 3743-3750 (2008)

[19] C. M. Clark. Probabilistic Road Map sampling strategies for multi-robot motion planning, *Rob. Auton. Syst.* **53**, 244–264 (2005)

[20] R. Kala, Multi-Robot Path Planning using Co-Evolutionary Genetic Programming, *Expert Syst. Appl.* **39**(3), 3817-3831 (2012)

[21] R. Kala, A. Shukla, and R. Tiwari, Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness, *Neurocomput.* **74**(14-15), 2314-2335 (2011)

[22] R. Gayle, A. Sud, E. Andersen, S. J. Guy, M. C. Lin, and D. Manocha, Interactive Navigation of Heterogeneous Agents Using Adaptive Roadmaps, *IEEE Trans. Visual. Comput. Graph.* **15**(1), 34-48 (2009).

[23] R. Gayle, A. Sud, M. C. Lin, and D. Manocha, Reactive Deformation Roadmaps: Motion Planning of Multiple Robots in Dynamic Environments, in *Proc. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, pp. 3777-3783 (2007)

[24] R. Kala, Supplementary Video for this paper, Available at http://youtu.be/6y9jlXwi_V4, Accessed 11th December, 2012