International Journal of Automation and Computing 11(2), April 2014, 153-161
DOI: 10.1007/s11633-014-0777-6

Biologically Inspired Node Generation Algorithm for Path Planning of Hyper-redundant Manipulators Using Probabilistic Roadmap

Eric Lanteigne[1] Amor Jnifene[2]

[1]Department of Mechanical Engineering, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada

[2]Department of Mechanical and Aerospace Engineering, Royal Military College of Canada, Kingston, ON, K7K7B4, Canada

Abstract: This article describes a biologically inspired node generator for the path planning of serially connected hyper-redundant manipulators using probabilistic roadmap planners. The generator searches the configuration space surrounding existing nodes in the roadmap and uses a combination of random and deterministic search methods that emulate the behaviour of octopus limbs. The strategy consists of randomly mutating the states of the links near the end-effector, and mutating the states of the links near the base of the robot toward the states of the goal configuration. When combined with the small tree probabilistic roadmap planner, the method was successfully used to solve the narrow passage motion planning problem of a 17 degree-of-freedom manipulator.

Keywords: Path planning, hyper-redundant manipulators, probabilistic road map (PRM), quasi-deterministic node generation, bi-directional search algorithm.

1 Introduction

Hyper-redundant manipulators are highly dexterous robotic devices typically composed of numerous serially connected modules and actuators. The high level of redundancy allows the robot to conform to the surrounding environment while fulfilling a given task. These mechanisms benefit from the ability to explore narrow passages and to perform new forms of robot grasping and locomotion as well as from having structures with increased robustness to mechanical failure[1]. However, the large numbers of degrees-of-freedom (DOF) produces countless configurations to a given task, and a near infinite number of distinct motion paths between the two poses.

The path planning problem in robot manipulation involves selecting a set of feasible joint motions given an initial and goal arm configuration that adhere to both internal constraints such as restrictions on actuator motions, singularities and joint failures, and external constraints such as obstacles in the workspace and self-collisions. Compared to traditional velocity-based methods, probabilistic roadmap (PRM) planners are particularly effective for solving the motion planning problems of hyper-redundant manipulators[2]. A recent article[3] presented a configuration deactivation algorithm for boosting probabilistic roadmap planning for robots. The algorithm was shown to improve the execution time needed to build the PRM planners.

PRMs are graphs of connected nodes. Each node corresponds to an obstacle-free and self-collision-free robot configuration. The nodes of the graph are connected by edges via a local planner that tests the configurations space between each set of two nodes. The majority of the the sampling strategies used to create the nodes of the roadmap were developed for pre-processing PRM planners. Pre-processing PRM planners have been shown to perform well in many different types of difficult planning situations and are particularly suitable when multiple queries are performed in the same static environment[4]. These typically focused on the difficult regions of the configuration space C to improve the coverage of the roadmaps[5, 6]. These methods either attempted to uniformly cover the C space, to cover the useful areas, to connect the difficult regions containing obstacles, or to connect disjoint roadmap components. Using random reflection at the obstacle boundaries was proposed in [7] to connect disjoint roadmaps. The subgraphs of the roadmap are connected by shooting edges in random directions from one of the nodes in the subgraph. When one of the edges reaches an obstacle boundary in C space, the resulting contact point is checked to see whether it can be connected to the main roadmap graph. In another approach, Amato and Wu[8] proposed a randomized roadmap constructed entirely of nodes on the obstacle boundaries. Although the method can produce connections in difficult situations such as long and narrow passages, there are scenarios in which the strategy does not yield good distributions such as cluttered environments or environments with only several obstacles. In [9], difficult regions in the configuration space are identified and additional configurations in those regions are inserted to increase the network connectivity. This is achieved by associating a weight to each node in the roadmap. The weights represent the ratio between the total number of times the local planner failed to connect the node to a neighbouring node, and the total number of attempts for that nodes. Nodes then are selected with a probability proportional to their weights, and expanded to complete the connectivity of the free-space.

Manuscript received April 11, 2013; revised June 14, 2013

The method described in [10] features a random node sampling strategy which moves all nodes, whether collision-free or not, onto the medial axis of the free-space. The authors demonstrated that the resulting roadmaps contained a significant amount of nodes in narrow passages and corridors. Compared to random sampling strategies, uniformly distributed node generators have shown to produce roadmaps with better coverages of the free-space in certain difficult situations. Roadmaps obtained using Hammersley-Halton sets could solve the narrow passage problem with less nodes compared to roadmaps obtained using a purely random sampling strategy[11]. Probabilistic roadmaps have also been applied to solve single-query path planning problems with no a-priory knowledge of the working environment and no pre-preprocessed roadmap. The single-shot PRM method solves motion planning problems one at a time by building a distinct roadmap for each two input configurations. Single-shot methods are particularly suited for dynamic environments or in situations with just a few input queries such as in applications like spot welding of low production components where the cost of pre-computing a roadmap is too high due to both the high dimensionality of the configuration space and the geometric complexity of the obstacles and the robot[12]. Single-shot PRM planners were developed by a number of researchers[13, 14]. The weighting scheme described in [9] was applied to the sampling strategy of single-shot PRM planners in [15]. New nodes are added to the roadmap by repeatedly sampling the neighbourhood of nodes known to be connected to either the initial or the goal nodes. This is achieved by selecting a parent node from the roadmap and randomly selecting a child node from the set of configurations located in an area surrounding the parent node. An even diffusion of nodes in the configuration space of the roadmap is ensured by selecting the nodes of the roadmap with the lowest density of surrounding nodes. Each node in the roadmap is attributed a weight function $w(n)$ corresponding to the density of nodes surrounding that node. Parent nodes are selected from the roadmap with a probability of $\frac{1}{w(n)}$.

2 Small tree PRM planner

The small tree (ST) algorithm is a PRM planner that creates small alternating roadmaps of limited size at the two input queries until a solution is found[16]. The algorithm begins by building a small uni-directional roadmap, rooted at the initial query configuration, and attempts to find a path connecting it to the final query configuration. If the planner is incapable of finding a solution when the graph reaches its maximum size, the best collision-free portion of the unsuccessful attempts at connecting the roadmap to the final query configuration is inserted in the final solution path. A new roadmap is then constructed from the opposite query towards the tip of the new path found by the previous roadmap. The planner alternates between the two input queries and saves the best collision-free portion of each attempt until a connection fuses the disconnected paths and completes the solution. The ST algorithm is controlled by three parameters: N — the maximum tree size, δ — the distance threshold for connecting the roadmap to the target node, and S — the number of state mutations of the node generator. The input queries to the planner are the initial configuration q_i and the final configuration q_f of the manipulator, and the output is an ordered set of nodes n corresponding to the collision-free configurations of the solution path. In the initialization stage, the initial input configuration q_i is arbitrarily transfered to the root node n_r of the first roadmap tree, and the final configuration q_f is transfered to the target node n_t for that roadmap. The flowchart is composed of three main components: the inner, outer and roadmap loops.

To generate a new node in the roadmap, a parent node is selected at random from the nodes of the roadmap. A collision-free child node n_c is then created by searching the configuration space around in the neighborhood of n_p. This is achieved by randomly mutating S states of the parent node n_p. The manipulator used in the present simulations uses discretely actuated revolute joints such that each state mutation is equivalent to displacing one module from one discrete position to the adjacent position. The algorithm executes η attempts to form a collision-free child node. If the initial η attempts fail, the search area is reduced by decreasing the number of state mutations. If a child node cannot be produced after the η^S attempts, it is assumed that the parent node is trapped by the surrounding obstacles and a child cannot be formed from this node. The planner then simply selects a new parent node and resumes the search.

The forward kinematics of the manipulator are computed for each new node and the module positions are discretized and superimposed with simulated obstacle maps to determine if the manipulator configuration is located in the free-space. If n_c is collision-free, the node is inserted in the roadmap and linked to n_p. The ST planner incorporates the lazy collision checking method to accelerate the convergence of the solution[17]. The edge connecting n_c to n_p is not checked in the roadmap construction phase. The local planner only checks the edges contained in the path of the branch considered close to the target configuration. The child node is considered close to the target node if the length of the edge connecting n_c and n_t is less than a fixed threshold δ. For every new child node, the distance between n_c and the n_t is computed using the following metric:

$$d(n_c, n_t) = \sum_{i=1}^{M} |(s_i)_c - (s_i)_t| \tag{1}$$

where d is a distance metric in configuration space, s_i is the state of the i-th module, and M is the total number of modules in the manipulator. The three steps of the inner loop are illustrated in Fig. 1 (a). If $d(n_c, n_t) < \delta$, the child node is considered close to the target node and the edges of the path containing n_c are checked sequentially from n_r to n_t using the local planner. This path is illustrated by the dotted line in Fig. 1 (b). In Fig. 1 (c), a collision is detected at the 6th edge of the path. The set of nodes in the collision-free portion of the path connected to n_r is saved in the partial solution path (PSP) υ_1, and all subsequent nodes of the branch are deleted from the roadmap. The distance $d_1(n_{\upsilon_1}, n_t)$ between the last node n_{υ_1} of υ_1 and

the target node is also saved. When the roadmap reaches the maximum tree size N, the i-th partial solution path υ_i with the smallest d_i is inserted in the final solution path. In the example of Fig. 1 (d), the partial solution path υ_2 is selected (assuming the linear distance to n_t is representative of the distance in configuration space). Other criteria could be used to select the best PSP, but the distance metric provided the highest number of successful paths based on the current simulations. Once a PSP is selected, it is added to the final solution, and the current roadmap and all of its nodes are removed from the system. The locations of the new n_r and n_t of the new roadmap are given as follow:

1) The root node of the new roadmap is the target node of the previous graph ($n_r \leftarrow n_t$).

2) The target node of the new roadmap is the last node the best PSP ($n_t \leftarrow n_{\upsilon_{\text{best}}}$).

This is illustrated in Fig. 1 (e). The new n_r is set to the target node n_t of the previous graph, and the new n_t is set to the n_{υ_2} of υ_2. The planner then generates a new roadmap to connect the two new nodes. The planner alternates between the two input queries, repeating the process described above, until a connection fusing both branches is found. Fig. 1 (f) illustrates the final stage of the planner, where υ_2 is able to connect both branches of the final solution path. Once a solution is found, the path is optimized using a simple algorithm that eliminates a large portion of

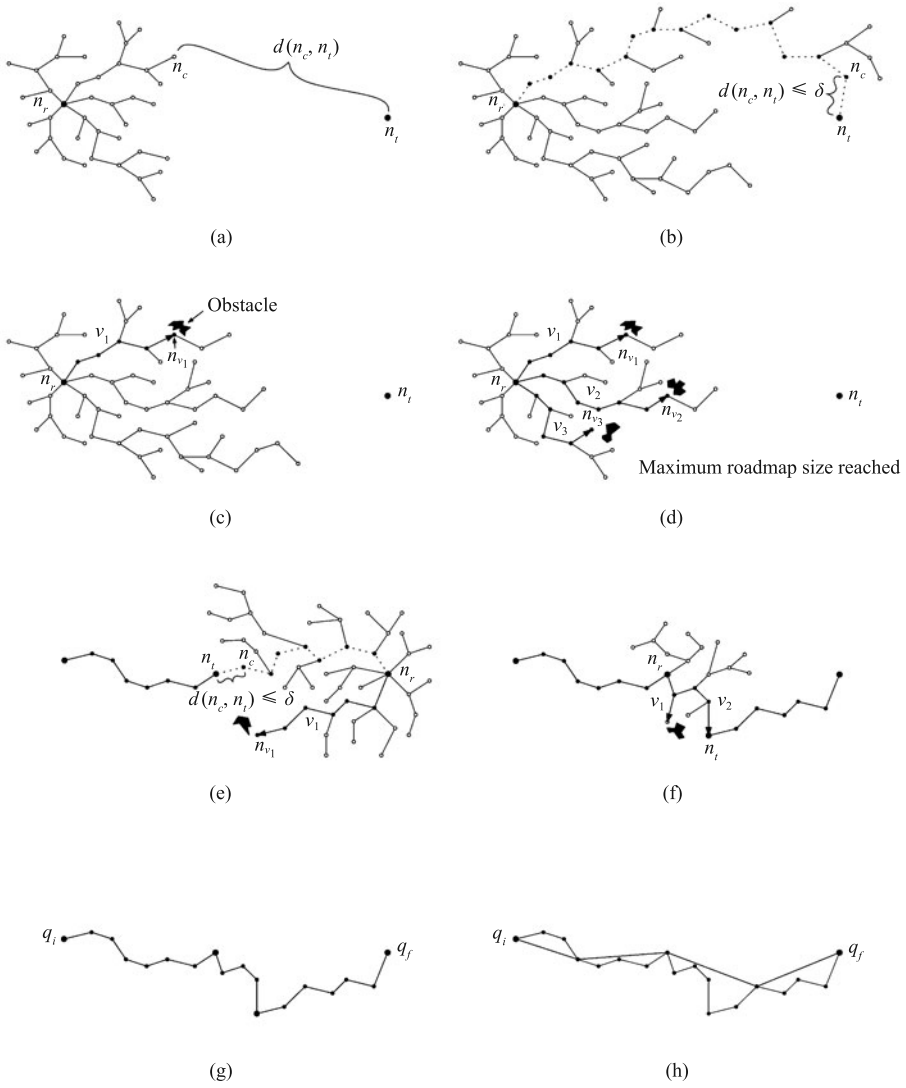

Fig. 1 ST roadmap. (a) Illustrates the generation of the first roadmap; (b) Illustrates a child node in proximity to the target node along with the potential solution path; (c) Illustrates the search direction of the local planner and the elimination of the branch in collision; (d) Illustrates the saved partial solution paths when the roadmap reaches its maximum size; (e) Illustrates the generation of the second roadmap; (f) Illustrates the generation of the third and final roadmap with the collision-free path; (g) Illustrates the solution path with the query configurations; (h) Illustrates the optimized solution path

the configurations in the solution path. The method connects two nodes at random from the solution path and checks the resulting edge using the local planner. If no collisions occur, all the nodes between the two configurations are removed from the solution. The optimization process stops when no further nodes can be eliminated. This is illustrated in Fig. 1 (h).

3 Motion trends in hyper-redundant manipulators

The goal of any single-shot PRM planner is to find a sequence of collision-free configurations that align the dimensions of two known query configurations. In an obstacle-free workspace, the sequence of configurations could simply contain the set of configurations corresponding to the linear transition between the states of the initial and goal configurations. The introduction of obstacles in the workspace forces the robot to choose a different path. The contraction and expansion of the robot structure leads to erratic transitions in the states of the robot actuators. Several observations can be made as to the nature of the transition of states of the hyper-redundant manipulator. Whereas most modules exhibit a highly nonlinear transition between states, a selected number of modules exhibit a smooth transition when moving from the initial to the goal configuration.

Consider the planar manipulator shown in Fig. 2. It is composed of $M = 17$ discretely-actuated modules, where each module has a single revolute joint capable of attaining the five equally-spaced positions shown in Fig. 3. In this example, the goal is to move the end-effector across the circular obstacle in the downward direction. The solution to this problem was solved using the small tree (ST) PRM planner. Fig. 4 illustrates a typical transition of states between the initial configuration and the goal configuration for three modules of the manipulator, where module 1 is connected to the fixed reference frame and module 17 is connected to the end-effector. It was found that, in general, the base module (module 1) had a smooth transition between its initial and goal states whereas the effector module (module 17) exhibited an erratic transition. This is physically sound since the distal modules of the manipulator must contract to allow the effector to circumvent the obstacle while conforming to the configuration changes of the proximal modules. A large portion of the solution paths generated by the ST and other planners exhibited a trend similar to Fig. 4 when confronted with an obstacle. The proximal modules moved from the initial to the final state with little or no deviations whereas the distal modules displayed erratic motions. This trend was also observed in the limbs of octopodes and other biological creatures. The image sequence shown in Fig. 5 illustrates an octopus stretching one of its limbs to bring food to its mouth. Although there are no obstacles in the immediate vicinity, the proximal joint region (denoted by the arrow closest to the body) demonstrates a fairly linear motion pattern whereas the distal joint bend performs a complex maneuver to grab and position the food in the mouth[18]. The node generator described in the following section attempts to capitalize on the motion patterns by imposing the motion trends in the state mutations of the node generator.

Fig. 2 End-effector motion of a planar hyper-redundant manipulator

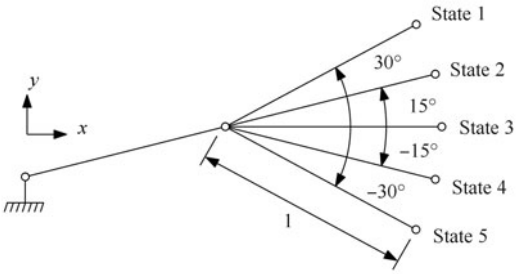

Fig. 3 Discrete actuator states

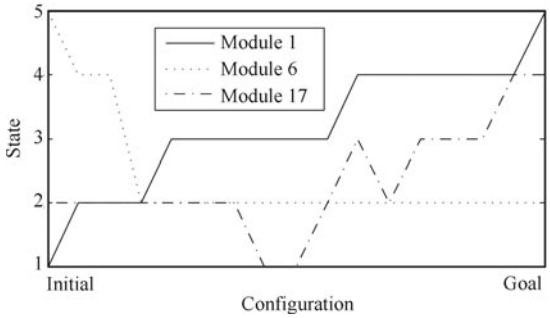

Fig. 4 State sequence for three select modules

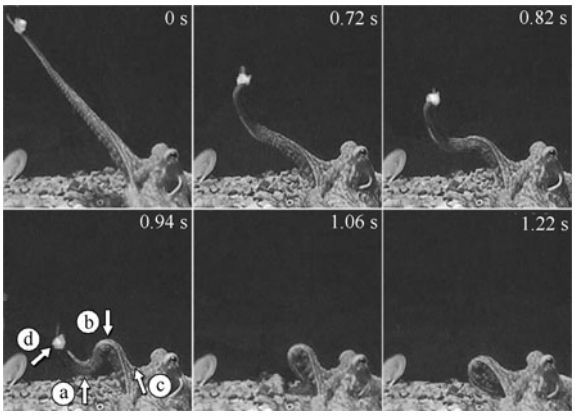

Fig. 5 Octopus food fetching sequence[18]

4 Base-first quasi-deterministic node generator

Two basic strategies can be defined for moving across the circular obstacle of the previous section. Fig. 2 illustrates the strategy of moving the effector past the obstacle first, then following through with the body, and Fig. 6 illustrates

the strategy of moving the trunk past the obstacle, then following through with the effector. In the former, the path of the effector is the effective solution to the motion planning problem and the body of the manipulator adjusts itself to accommodate the effector. This strategy is typically used in resolved-motion rate methods or potential field methods as they typically solve the path planning problem in the operational space, then convert the solution to the configuration space through the inverse kinematics of the manipulator. In the latter, the trunk is the driving component and the effector is the follower. The trunk section attempts to reach the goal configuration while the effector is tasked with avoiding the obstacles in its path. This strategy is inherently suited for PRMs as it is solely dependent on the differences between the states in configuration space, and does not rely on the effector path in operational space or the use of the inverse kinematics. The base-first quasi-deterministic (BFQD) node generator mimics the strategy of Fig. 6 by directly specifying the joint motion of the base modules when generating nodes. The random node generator of the original small tree algorithm creates a child node by mutating S states of the parent node. In the BFQD, the total number of state mutations S is split into two components: S_D, the number of deterministic state mutations, and S_R, the number of random state mutations. The deterministic component aligns the states of the parent node with the states of the target node, and the random component ensures an even diffusion of nodes in the roadmap. In other words, the deterministic component focuses the search area of the planner and the random component maintains the essence of the PRM. Two normal distribution functions, centered at the base module and at the effector module, control the probability of selecting modules for deterministic and random state mutations, respectively. The following two equations select the modules m_d and m_r slated for deterministic and random state mutations.

$$m_d = \text{ceil}\left(|\text{randn}| \times \left(\frac{M}{3}\right)\right) \tag{2a}$$

$$m_r = |\text{ceil}\left(|\text{randn}| \times \text{ceil}\left(\frac{M}{3}\right)\right) - (M+1)| \tag{2b}$$

where the "ceil" command rounds towards positive infinity, the "randn" command generates a normally distributed random number, and M is the number of modules in the manipulator (note: (2) occasionally selects modules out of the range of M, a while loop is executed until both m_d and m_r produce modules within the limits of M). The statistical probability of selecting a module for deterministic or random mutation is illustrated in Fig. 7 by applying 10^4 iterations of (2) on a manipulator with 17 modules.

Fig. 6 Base motion of a planar hyper-redundant manipulator

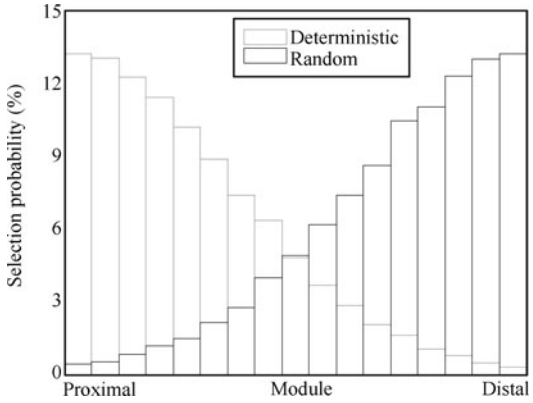

Fig. 7 Probability of selecting a module for mutation

5 Simulation results

The quasi-deterministic ST path planner was implemented in MATLAB^{TM} and tested on a 2.5 GHz AMD Athlon 64 X2 processor with 2 GB of internal memory. Over 10^3 tests were performed for each simulation and individual tests were terminated if the total number of generated nodes exceeded 5×10^4. For comparison purposes, two other quasi-deterministic node generators and a purely random node generator were tested on the circle obstacle scenario shown in Fig. 6 to determine the effectiveness of the proposed method. The circle obstacle has a diameter of 4 units and its centre is at a horizontal distance of 15 units from the base of the manipulator (with respect to the unit length of one link). The two alternate methods include the uniformly deterministic and effector-first deterministic node generators. The uniformly deterministic method selects random nodes for both deterministic and random state changes, whereas the effector-first method uses a probability function opposite to that of Fig. 7. The effector nodes are favoured for deterministic state changes and the base nodes are favoured for random state changes. The parameters controlling ST and the node generators were standardized as follows: $N = 125$, $\delta = 8$, $S = 4$ for the random node generator, and $S_D = S_R = 2$ for the three quasi-deterministic node generators such that a total of $S = S_D + S_R = 4$ states are modified. The results are shown in Fig. 8.

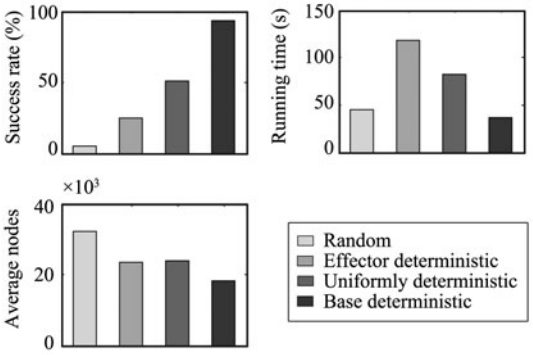

Fig. 8 Results of various node generators

The BFQD generator obtained the highest success rate of all the methods. It could solve 93.3% of the simulations, compared to 5% for the ST planner with the random node

generator. The ratio between the random and the deterministic components of the BFQD method was also investigated to determine the effects of both components on the performance of the planner. To illustrate the effects of the deterministic component, the ST planner with various combinations of S_R and S_D was subjected to the circle obstacle scenario of Fig. 6. In all tests, the total number of state mutations was held constant at $S = S_R + S_D = 4$. The results of the ST planner with $N = 125$ and $\delta = 8$ are shown in Fig. 9. For this set of parameters, both the fully random and fully deterministic node generators were incapable of solving 100% of the path planning simulations. The generator with equal random and deterministic components $S_R = S_D$ was chosen for all subsequent simulations as it demonstrated the best combination of minimum running time and average generated nodes, and could easily be adapted to other values of S ($S = 6, 8, 10, \cdots$). The three main parameters controlling the ST planner, N, S and δ, were examined in a number of different simulations to explore the behaviour of the method and to establish its performance. The total number of state mutations S in the original ST planner described in [19] is replaced by S_R and S_D of the BFQD ST planner. The two components of the node generator are equal in all simulations, and the parameter $S = S_D + S_R$ is again used to represent the total number of state mutations. The first series of simulations were performed on the circle obstacle scenario. Table 1 lists the success rate, the average execution time and the average number of nodes generated by the planner for roadmap sizes between 25 and 300 nodes, with different combinations of S and δ. The planner equipped with the BSQD generator exhibited a 100% success rate when the threshold was set at $\delta = 8$, with a slight decrease in the execution time given a higher mutation rate. The difference between high and low values of N was also examined on the narrow passage path planning problem shown in Fig. 10. The results for fixed values of $N = 25$ and 125 are given in Tables 2 and 3, respectively.

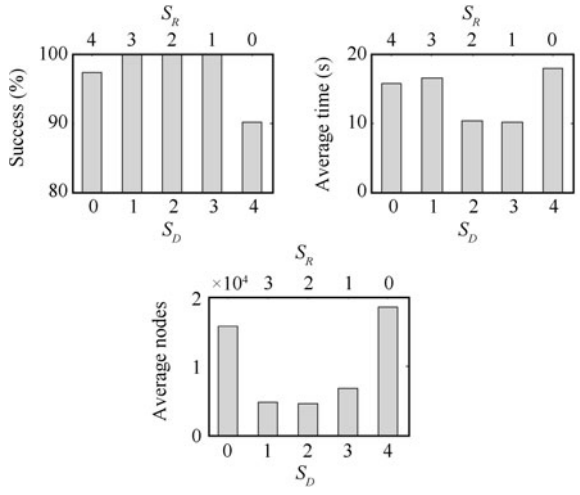

Fig. 9 Results of various degrees of randomness

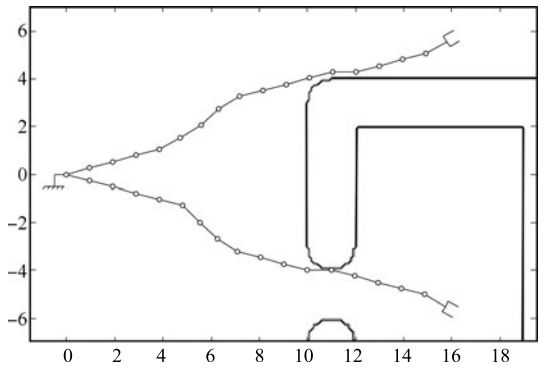

Fig. 10 The narrow passage scenario

Table 1 Variation of N

(a) Success rate (%)

δ	S	N			
		25	75	125	300
4	4	35.0	94.8	94.6	83.0
4	8	98.8	98.0	97.7	90.1
8	4	100	100	100	100
8	8	100	100	100	100

(b) Execution time (s)

δ	S	N			
		25	75	125	300
4	4	28.7	20.5	18.4	28.8
4	8	20.8	18.5	19.9	24.5
8	4	9.1	13.0	14.2	18.2
8	8	6.5	9.2	9.9	11.5

(c) Average nodes ($\times 10^3$)

δ	S	N			
		25	75	125	300
4	4	27.4	20.2	18.1	22.6
4	8	14.2	14.8	15.9	18.0
8	4	2.7	3.8	4.2	6.3
8	8	1.8	2.4	2.9	3.1

Table 2 Variation of S and δ with $N = 125$

(a) Success rate (%)

S	δ			
	6	8	10	12
4	80.4	93.3	95.0	96.1
8	96.4	99.5	99.8	99.9
12	99.1	99.6	99.7	100

(b) Execution time (s)

S	δ			
	6	8	10	12
4	41.2	36.8	47.1	64.9
8	38.1	33.8	44.0	62.1
12	34.4	32.0	44.0	65.0

(c) Average nodes ($\times 10^3$)

S	δ			
	6	8	10	12
4	25.9	18.2	16.0	15.1
8	15.8	11.4	10.0	9.2
12	12.5	8.9	8.0	7.7

Table 3 Variation of S and δ with $N = 25$

(a) Success rate (%)

S	δ			
	6	8	10	12
4	37.9	99.9	99.8	99.9
8	99.8	99.8	100	100
12	99.5	100	100	100

(b) Execution time (s)

S	δ			
	6	8	10	12
4	50.8	21.2	22.8	28.2
8	29.3	13.7	15.9	21.8
12	32.8	13.6	15.5	22.0

(c) Average nodes ($\times 10^3$)

S	δ			
	6	8	10	12
4	29.0	10.0	8.0	7.9
8	11.0	4.6	4.0	4.2
12	10.7	4.1	3.4	3.6

The performance of the planner with the BFQD node generator displayed a similar trend to the original results of the ST planner with the random node generator. Planners with smaller roadmaps were faster and more repeatable than those with larger roadmaps. The ST planner with $N = 25$ could solve the narrow passage problem with a 100% success rate in over half the amount of time compared to the ST planner with $N = 125$ (for several different combinations of S and δ). The optimum performance, labelled in bold in Table 3, was observed when $S = 12$ and $\delta = 8$. With these parameters, the planner could solve the problem with an average time of 13.6 s, and a standard deviation of 10.5 s. These results are consistent with the observations that smaller roadmaps produce better results.

A typical solution to the narrow passage path planning problem is shown in Fig. 11. The image illustrates the optimized solution path generated by the ST planner with the BFQD node generator, along with the intermediate configurations generated by the local planner. The parameters of the planner were set to $N = 25$, $S = 4$, and $\delta = 8$. For this particular solution, 5 068 nodes were generated in 21 s, and the final solution path contained 852 nodes. The optimized path illustrated in Fig. 11 contained 4 nodes in addition to the initial and goal nodes (shown in black along with the intermediate configurations in grey) and was obtained in less than 2 s.

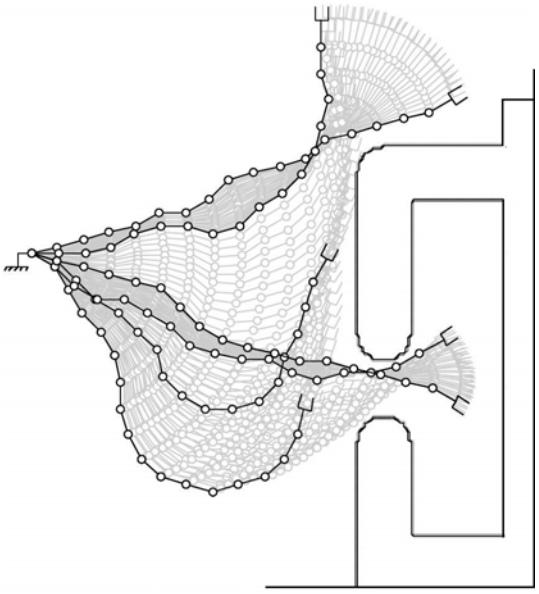

Fig. 11 Optimized solution path to the narrow passage scenario

The BFQD ST planner was also tested on a 45-DOF manipulator composed of the same discretely-actuated modules as described above. The motion planning problem consists of moving the effector past a circular obstacle of diameter 8 units centred at $x = 35$ is shown in Fig. 12. Approximately 10^2 tests were performed for each simulation on a 2.0 GHz Intel Core 2 processor with 2 GB of internal memory. Individual tests were terminated if the total number of generated nodes exceeded 5×10^4. The roadmap size N of the ST planner was examined in a number of different simulations with various values of S. The two components of the BFQD node generator are again equal in all simulations, and the parameter $S = S_D + S_R$ is again used to represent the total number of state mutations. The success rate, average running times, and average number of nodes for $\delta = 20$ are given in Table 4.

The BFQD equipped planner is capable of repeatedly solving the high DOF problem using many different combinations of parameters. Although the number of tests for each combination is small, the planner required larger roadmaps to successfully solve this problem across all values of S. However, the optimal performance was obtained when $N = 30$, as the average execution time, the average number of nodes and the standard deviation of the number of nodes was lower for that roadmap size.

Table 4 Variation of N and S on a 45-DOF manipulator

(a) Success rate (%)

S	N						
	10	30	50	100	200	300	500
14	0	51.2	95	100	100	100	100
20	0	99	100	100	100	100	100
26	100	100	100	100	100	100	100
32	100	100	100	100	100	100	100

(b) Execution time (s)

S	N						
	10	30	50	100	200	300	500
14	NA	84.0	49.2	31.0	23.7	22.7	26.7
20	NA	40.2	16.9	14.4	16.6	18.6	19.2
26	10.3	8.8	10.1	9.9	8.9	10.2	9.2
32	9.6	10.2	9.0	10.7	14.1	14.5	18.1

(c) Average nodes ($\times 10^3$)

S	N						
	10	30	50	100	200	300	500
14	NA	25.4	13.5	6.9	4.9	4.7	5.4
20	NA	11.4	3.8	2.9	3.1	3.6	3.6
26	2.0	1.8	2.0	2.0	1.8	2.0	1.9
32	2.0	2.1	1.6	1.8	2.3	2.5	3.0

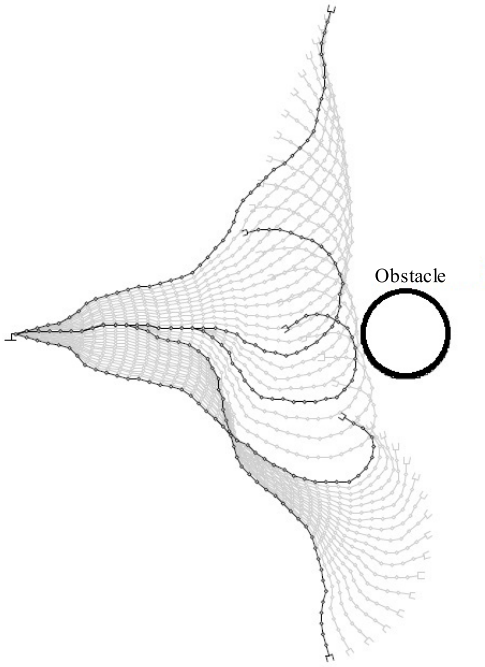

Fig. 12 Optimized solution path of a 45-DOF manipulator

6 Discussion

In all test cases, the BFQD node generator accelerated the execution time, increased the repeatability of the planner, and reduced the variability of the results. The strategy does have several limitations. Firstly, the node generation cannot be fully deterministic. Of all the simulations, the planner with the fully deterministic sampling strategy generated the poorest results. Secondly, the selection criteria for random or deterministic state changes plays an important role in the performance of the planner. Although it was demonstrated that the base-first strategy with the given probability functions generated the best results, other distribution functions might be capable of producing better results.

The ST planner with the BFQD node generator operated at its peak performance when the roadmap size was less than 100 nodes, and produced repeatable results when $S > 8$ and $\delta > 8$. The performance of the ST was found to be dependent on the combination of S and δ as well as on the complexity of each scenario. Higher values of S and δ generated preferable results in simpler environments but increased the running time by a considerable amount for more difficult environments. The running time of the ST planner was particularly sensitive to variation in δ in difficult path planning scenarios. Increasing δ from 8 to 12 approximately doubles the running time when applied to the narrow passage scenario.

The application of the ST algorithm to the longer 45-DOF manipulator demonstrates the limitations of the local planner described in [19]. The intermediate configurations, shown in grey in Fig. 12, do not consistently cover the operational space. There are regions where too many or too few intermediate configurations are generated. In the first case, this unnecessarily increases the execution time of the planner. In the second case, this could potentially cause the local planner to miss small obstacles and generate false collision-free paths.

7 Conclusions

In this article, we describe and evaluate a novel quasi-deterministic node generator for the path planning of serially-connected hyper-redundant manipulators using PRM planners. The node generator increases the repeatability of the planning process in complex environments compared to strictly random generators, and accelerates the planning process in all test cases. The concept of using probability functions to obtain the deterministic and random components of the generator could be further investigated by analyzing the motion of other hyper-redundant limbs found in nature.

References

[1] G. S. Chirikjian, J. W. Burdick. A hyper-redundant manipulator. *IEEE Robotics and Automation Magazine*, vol. 1, no. 4, pp. 22–29, 1994.

[2] A. Ladd, E. E. Kavraki. Generalizing the analysis of PRM. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Washington. DC, USA, vol. 2, pp. 2120–2125, 2002.

[3] M. T. Rantanen, M. Juhola. A configuration deactivation algorithm for boosting probabilistic roadmap planning of robots. *International Journal of Automation and Computing*, vol. 9, no. 2, pp. 155–164, 2012.

[4] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proceedings of the 3rd Workshop on the Algorithmic Foundations of Robotics on Robotics: the Algorithmic Perspective*, A. K. Peters, Ltd., Natick, MA, USA, pp. 155–168, 1998.

[5] L. E. Kavraki, P. Svestka, J. C. Latombe, M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[6] V. Boor, M. H. Overmars, A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Detroit, MI, USA, vol. 2, pp. 1018–1023, 1999.

[7] T. Horsch, F. Schwarz, H. Tolle. Motion planning with many degrees of freedom-random reflections at C-space obstacles. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, San Diego, CA, USA, vol. 4, pp. 3318–3323, 1994.

[8] N. M. Amato, Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Minneapolis, MN, USA, vol. 1, pp. 113–120, 1996.

[9] L. E. Kavraki, J. C. Latombe. Randomized preprocessing of configuration for fast path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, San Diego, CA, USA vol. 3, pp. 2138–2145, 1994.

[10] S. A. Wilmarth, N. M. Amato, P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Detroit, MI, USA, pp. 1024–1031, 1999.

[11] J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*, Berlin, Heidelberg: Springer, 1999.

[12] M. Saha, T. Roughgarden, J. C. Latombe, G. Sánchez-Ante. Planning tours of robotic arms among partitioned goals. *International Journal of Robotics Research*, vol. 25, no. 3, pp. 207–223, 2006.

[13] J. Barraquand, J. C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[14] D. Vallejo, I. Remmler, N. M. Amato. An adaptive framework for single shot motion planning: A self-tuning system for rigid and articulated robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Seoul, South Korea, vol. 1, pp. 21–26, 2001.

[15] D. Hsu, J. C. Latombe, R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Albuquerque, NM, USA, vol. 3, pp. 2719–2726, 1997.

[16] E. Lanteigne, A. Jnifene. Small tree probabilistic roadmap planner for hyper-redundant manipulators. In *Proceedings of the 2nd International Conference on Autonomous and Intelligent Systems*, Springer-Verlag, Berlin, Heidelberg, Germany, vol. 6752, pp. 11–20, 2011.

[17] R. Rohlin, L. E. Kavraki. Path planning using lazy PRM. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, San Francisco, CA, USA, vol. 1, pp. 521–528, 2000.

[18] G. Sumbre, G. Fiorito, T. Flash, B. Hochner. Octopuses use a human-like strategy to control precise point-to-point arm movements. *Current Biology*, vol. 16, no. 8, pp. 767–772, 2006.

[19] E. Lanteigne, A. Jnifene. A probabilistic roadmap planner for highly redundant manipulators. In *Proceedings of the CSME Forum*, pp. 1–7, 2010.

Eric Lanteigne received his B. Sc. and M. Sc. degrees from the University of Ottawa, Canada in 2003 and 2006, respectively, and his Ph. D. degree from the Royal Military College of Canada, Canada in 2011. He is currently an assistant professor in the Department of Mechanical Engineering at the University of Ottawa.

His research interests include design, control and automation of robotic manipulators, unmanned aerial vehicles, as well as mechanism kinematics and design, particularly those related to the trench-less rehabilitation of municipal water mains.

E-mail: Eric.Lanteigne@uottawa.ca

Amor Jnifene received his M. Sc. and Ph. D. degrees from the Department of Mechanical Engineering, University of Ottawa, Ottawa, Canada in 1989 and 1996, respectively. He is currently a full professor and the associate head undergraduate at the Department of Mechanical and Aerospace Engineering, Royal Military College of Canada, Kingston, Canada. He is on the editorial board of the *Transactions of the Canadian Society of Mechanical Engineering*.

His research interests include the dynamics and control of flexible structures, collaborative multi-robot systems, adaptive structure, nero-fuzzy control, and bio-robotics.

E-mail: Amor.Jnifene@rmc.ca (Corresponding author)