2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Balancing Exploration and Exploitation in Motion Planning

Markus Rickert† Oliver Brock‡ Alois Knoll†
†Robotics and Embedded Systems Lab, Department of Computer Science, Technische Universität München
‡Robotics and Biology Laboratory, Department of Computer Science, University of Massachusetts Amherst

***Abstract*— Computationally efficient motion planning must avoid exhaustive exploration of configuration space. We argue that this can be accomplished most effectively by carefully balancing exploration and exploitation. Exploration seeks to understand configuration space, irrespective of the planning problem, while exploitation acts to solve the problem given the available information obtained by exploration. We present an exploring/exploiting tree (EET) planner that balances its exploration and exploitation behavior. The planner acquires workspace information and subsequently uses this information for exploitation in configuration space. If exploitation fails in difficult regions, the planner gradually shifts its behavior towards exploration. We present experimental results demonstrating that adaptive balancing of exploration and exploitation leads to significant performance improvements compared to other state-of-the-art sampling-based planners.**

I. Introduction

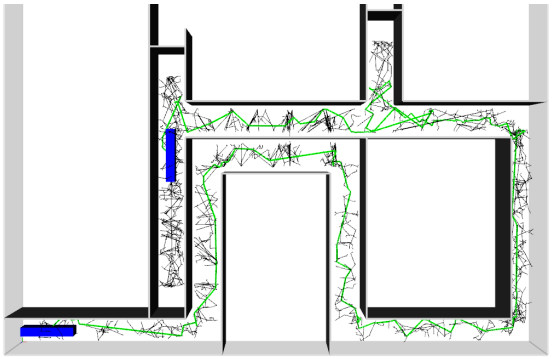
Sampling-based motion planners routinely solve complex, high-dimensional planning problems. This may seem surprising, considering that the general motion planning problem [1] is PSPACE-hard [2]. However, the configuration spaces of many practical problems contain considerable structure that may help in solving a planning problem. In addition, not all parts of configuration space have to be explored to solve a particular motion planning problem. Today's sampling-based planners leverage these properties of practical motion planning problems to achieve computational efficiency.

We cast motion planning as a state space search problem, similar to work in reinforcement learning [3], [4], to gain insights on how to improve the computational efficiency of motion planners. In this formulation, there are two competing goals of planning: *exploration* and *exploitation*.

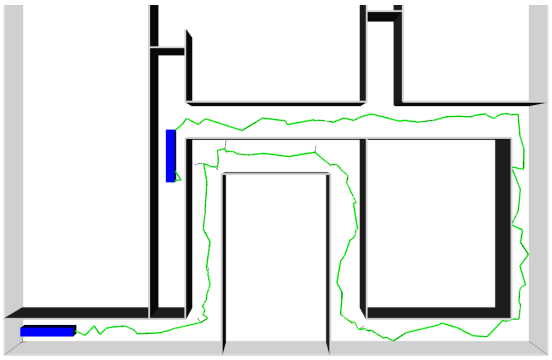
Exploration seeks to understand the connectivity of the configuration space, irrespective of a particular motion planning problem. Exploration thus does not assess if a region of configuration space is relevant for a particular task; rather, it explores to improve the planner's understanding of configuration space. A typical example from motion planning is the initial roadmap building phase of the basic PRM planner with uniform random sampling [5].

Exploitation strives to determine a valid path for a specific task, leveraging available information. Exploitation thus assumes that the information available suffices to solve the problem and just begins to act. For example, the artificial potential field approach performs pure exploitation [6].

Guided exploration, a technique performed by most sampling-based motion planners, improves on pure exploration by leveraging available information to guide exploitation based on characteristics of the underlying state space.



(a) Roadmap obtained by ADD-RRT



(b) Roadmap obtained by EET

Fig. 1. By balancing exploration and exploitation, the EET planner reduces the amount of configuration space search required to solve planning problems: the roadmap obtained by the EET contains fewer branches that are not part of the final solution path (shown in green).

This guidance is directed at achieving a complete understanding of the space and not at accomplishing a particular task.

Motion planning will be most efficient if exploration and exploitation are adequately balanced. Exploration is required to understand the connectivity of relevant configuration space regions. Exploitation should be used whenever greedy actions can solve sub-problems quickly. A planner can achieve computational efficiency by employing both exploration and exploitation when appropriate, given the local and global properties of a particular configuration space.

We present a motion planner based on exploring/exploiting trees (EETs). This planner leverages its understanding of the planning problem to balance exploration and exploitation. The planner begins by acquiring information about the workspace. It then leverages this information for exploitation in configuration space. If exploitation fails in difficult re-

gions, the planner gradually shifts to exploration. Our experimental results show that active balancing of exploration and exploitation results in significant performance improvements, up to several orders of magnitude, when compared to state-of-the-art planners (see Fig. 1).

II. Related Work

We classify motion planning methods based on whether they perform exploration, guided exploration, exploitation, or combinations thereof.

The original PRM planner with uniform random sampling performs pure exploration [5]. The exploratory behavior is not affected by the task or by information obtained during the exploration. Note, however, that the refinement step of PRM planners constitutes guided exploration.

A large number of sampling-based multi-query motion planners perform guided exploration. They assess properties of regions of configuration space to guide exploration. These properties can depend on obstacles [8], [9], visibility [10], or narrow passages [11]. Other planners use workspace information to adapt exploration [12], [13], [14], [15]. In another planner, global information about the entire configuration space is used to guide exploration [16].

Artificial potential field approaches employ pure exploitation [6]. The complete elimination of exploration makes these methods computationally efficient but also susceptible to local minima. This is also true for potential field approaches that are applied to entire paths [17].

Many sampling-based motion planners combine exploration and exploitation. However, all of these planners perform (possibly guided) exploration and exploitation as distinct steps of the planning process, rather than deliberately balancing the two within a unified framework.

Fuzzy PRM [18] and Lazy PRM planners [19] initially perform exploitation in a random configuration space graph. When this exploitation fails, these planners perform guided exploration to augment the graph in difficult regions.

A number of planners alternate between exploration and exploitation. The RPP planner combines potential field-based exploitation with random exploratory moves [20]. RRT planners alternate exploration based on the Voronoi bias with exploitation in the extend step [7]. When RRT planners employ two trees, a second exploitative step is taken when the planner attempts to connect both trees [21]. Variants of RRT planners replace exploration based on the Voronoi bias with guided exploration [22], [23], [24].

Other planners combine exploration and exploitation in workspace and configuration space. Some planners initially perform efficient exploitation in the low-dimensional workspace and subsequently use the obtained information to perform exploitation [25], [26] or guided exploration [27] in configuration space. Another planner performs workspace exploration and uses the resulting information to perform guided configuration space exploration [28].

III. Exploring/Exploiting Tree Planner

We develop a tree-based motion planner, called the Exploring/Exploiting Tree (EET) planner, that performs exploitation whenever possible and gradually transitions to exploration when necessary. The planner is based on tree expansion in configuration space, similar to RRT methods [21]. Our objective is to carefully balance exploratory and exploitative behavior so as to leverage the structure inherent in the planning problem for rendering motion planning as efficient as possible. We want to devise a planner that behaves like a potential field planner whenever possible and gradually turns into a complete motion planner when necessary.

The EET planner leverages three sources of information to perform exploitation and to balance between exploitation and exploration. To guide exploitation, the planner acquires global connectivity information for relevant portions of the workspace. This is achieved with a sphere-based wavefront expansion in workspace [25], resulting in a tree of workspace spheres. The branches of the tree capture the connectivity of the workspace. The size of the spheres along the paths in the tree captures the local free workspace (see Fig. 2). These spheres and their connectivity define an approximate navigation function over parts of the workspace [25]. The gradient of this navigation function defines an attractive force for a point on the robot, which "pulls" the robot towards the goal in the workspace. The workspace force is projected into a direction in the robot's configuration space using the Jacobian matrix. Exploitation uses this direction to move in configuration space.

Exploitation is only likely to be successful when it is based on accurate information. We therefore attempt to augment the information represented in the workspace spheres with more accurate information about the best configuration space directions for exploitation. This second source of information is derived from repulsive forces exerted by obstacles onto the robot [6]. We combine the repulsive forces with the attractive force derived from the workspace navigation function. The combined force can also be projected into a direction in configuration space using the Jacobian matrix.

To balance exploitation and exploration, the planner leverages a third source of information, namely the information obtained during the tree expansion steps. When exploitation fails, tree expansion will become increasingly exploratory, while successful tree expansions will lead to increasingly exploitative behavior.

We now describe the EET planner in detail (numbers in parenthesis refer to lines in Algorithm 1). The planner builds a configuration space tree, much like an RRT-based planner [21]. However, every vertex q in this tree T is associated with a workspace frame F, consisting of the position and orientation of a control point on the robot. This point can be the end-effector in the case of articulated robots, or an arbitrary point on the robot in the case of rigid body robots. Our EET planner requires this information to leverage workspace-based information for exploitation.

The starting configuration q_{start} is added to the configuration space tree T (1). The sphere-based workspace expansion determines a tree S of workspace spheres to capture workspace connectivity (2, see Fig. 2). We process the tree of workspace spheres S in depth-first fashion, considering

Algorithm 1: EET($q_{\text{start}}, q_{\text{goal}}, \alpha, \beta, \gamma$)

```
1  ADD_VERTEX(T, q_start)
2  S ← EXPLORE(q_start, q_goal)
3  forall s on depth-first traversal of solution path in S do
4      σ ← 1/γ
5      repeat
6          p_new ← GAUSS(s.center, σ · γ · s.radius)
7          R_new ← RAND()
8          q_near ← SELECT(T, p_new, R_new)
9          if σ < β then
10             R_Δ ← GAUSS(0, σ · π)
11             R_new ← R_near R_Δ
12         if q_new ← CONNECT(T, q_near, p_new, R_new) then
13             ADD_VERTEX(T, q_new)
14             ADD_EDGE(T, q_near, q_new)
15             σ ← (1 − α) · σ
16         else
17             σ ← (1 + α) · σ
18             if σ ≥ 1 then
19                 s ← i − 1
20     until ||p_new − s.center|| < s.radius
```

only paths through the tree that lead to the goal location (3). The sphere s captures the free workspace volume into which the robot is trying to move next. If the planner fails to find a solution based on this initial path through the tree S, we can perform backtracking.

We attempt to expand the tree T while balancing exploitation and exploration. Similarly to RRT methods, we create a configuration towards which we want the tree to expand. In contrast to RRTs however, we determine this configuration based on the workspace information contained in S. We want to "pull" either the entire robot (in the case of rigid bodies) or the robot's end-effector (in the case of articulated robots) into the direction indicated by the workspace connectivity information stored in S. This enables our planner to solve a task frame specification in workspace in addition to a specific goal configuration [29].

The variable σ (4) balances exploration and exploitation; its value ranges from zero to one. A value of zero indicates pure exploitation in which the behavior of the planner can be compared to a potential field planner based on an approximate global navigation function [25]. σ will be used to scale the variances of Gaussian distributions for generating samples. We initialize σ to the reciprocal value of the parameter γ. γ indicates how closely our planner follows the workspace information contained in S when generating new samples.

We sample a workspace point p_{new} normally distributed around the sphere's center with a variance that depends on the sphere's radius scaled by σ and γ (6). The function GAUSS takes two parameters: the mean of the Gaussian distribution and the width within which 99.7% of the samples should fall (three times the standard deviation); it returns a random sample from this distribution. We also sample a random orientation R_{new} (7). Together these parameters define a workspace frame F. We obtain from the tree T the configuration q_{near} for which the associated frame most closely matches F (8), using adequate sampling and distance metrics [30]. If the planner is able to perform exploitation, as indicated by a small σ (9), we replace the uniformly sampled orientation R_{rand} with one drawn from a Gaussian distribution centered around the orientation R_{near} of q_{near} and a variance that is scaled by σ (10, 11).

Next, we attempt to connect frame $(p_{\text{new}}, R_{\text{new}})$ to the tree T (12). The connect step is described in Algorithm 2; it is identical to the connect step for RRT-based planners [21]. Upon success, we add the penultimate vertex (13) and the corresponding edge (14) to the tree T and reduce the value of σ (15). This reduction causes the planner to shift towards exploitation. If the connection attempt fails, we increase σ and shift the balance towards exploration (17). If σ reaches the exploration limit (18), we backtrack to the previous sphere (19). The series of expansion steps ends when the boundary of the sphere s has been reached (20).

To complete the description of the EET planner, we now discuss the extend step (Algorithm 3) used by the connect algorithm (Algorithm 2). The extend step moves the frame F associated with configuration q_{near} towards the frame $(p_{\text{new}}, R_{\text{new}})$ and determines the corresponding new configuration q_{new}. This is accomplished by first determining the vector Δx, pointing from the existing frame towards the new frame (1). This displacement is translated into a displacement in configuration space using the pseudo-inverse of the Jacobian (2). Based on the resulting Δq, a new configuration q_{new} is determined (3). If it is collision free, we return q_{new}, otherwise we report failure. Repulsive forces from obstacles can be translated into configuration space directions in a similar fashion.

IV. Experimental Results

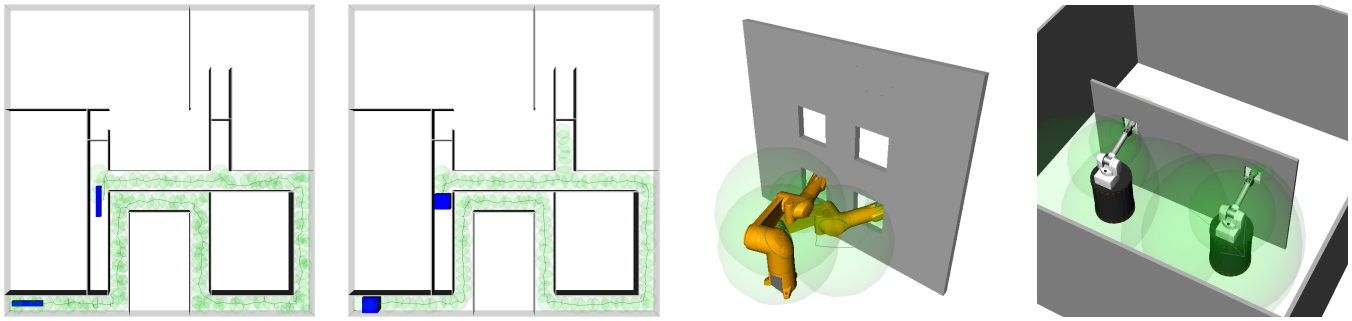
Our goal is to demonstrate the importance of carefully balancing exploration and exploitation in motion planning. We validate the proposed EET planner by comparing its performance with that of a PRM planner with uniform sampling (abbreviated as PRM in Table I) [5], PRM with Gaussian sampling (Gaussian PRM) [9], PRM with bridge

Algorithm 2: CONNECT($T, q_{\text{near}}, p_{\text{new}}, R_{\text{new}}$)

```
1  while q_new ← EXTEND(T, q_near, p_new, R_new) do
2      q_near ← q_new
3  return q_new
```

Algorithm 3: EXTEND($T, q_{\text{near}}, p_{\text{new}}, R_{\text{new}}$)

```
1  Δx ← |(p_near, R_near)(p_new, R_new)|
2  Δq ← J*(q_near)Δx
3  q_new ← q_near + Δq
4  if FREE(q_new) then
5      return q_new
6  else
7      return
```



(a) (b) (c) (d)

Fig. 2. Experimental scenarios: a) free-flying, three-dimensional box in a maze; b) free-flying, three-dimensional cube in a maze with little clearance to the walls; c) stationary, 6 degree-of-freedom manipulator arm with initial and final configurations inside a narrow passage; d) 10 degree-of-freedom mobile manipulator UMan with similarly constrained initial and final configurations. The spheres from the sphere-based workspace expansion are shown in green.

test (Bridge PRM) [11], RRTConnect with one tree (RRT-Connect), RRTConnect with two trees (RRTConnect2) [21], and adaptive dynamic-domain RRT (ADD-RRT) [23]. For these planners, we used a nearest neighbor search based on KD trees [31], with $k = 30$ for all PRM variants. Our planner uses a single tree to explore configuration space. We thus believe that the performance increase afforded by balancing exploration and exploitation can most accurately be assessed by comparing the EET planner to RRTConnect with one tree.

Our experiments are performed in four scenarios. The first scenario (see Fig. 2(a)) consists of a $30\ \mathrm{m} \times 30\ \mathrm{m} \times 2\ \mathrm{m}$ large maze with walls that are $2\ \mathrm{m}$ apart from each other. The robot is a free-flying rigid box (six degrees of freedom, $3\ \mathrm{m} \times 0.5\ \mathrm{m} \times 0.5\ \mathrm{m}$) moving from the lower left side of the maze to a position on the top right. Due to the length of the robot and the distance to the walls, traveling through the corners of the maze is difficult and requires exploration. The straight corridors can be solved by exploitation, provided the robot is aligned with the walls.

In the second scenario, a larger box ($1.5\ \mathrm{m} \times 1.5\ \mathrm{m} \times 1.5\ \mathrm{m}$) moves through the same maze (see Fig. 2(b)). The clearance to the walls is very small so that motion through the straight corridors becomes much more difficult.

The third scenario uses a stationary 6 degree-of-freedom manipulator (see Fig. 2(c)). The task consists of finding a way out of the first hole and into a different hole. Both holes represent narrow passages in the configuration space.

In the last scenario, the holonomic, 10 degree-of-freedom mobile manipulator UMan (UMass Mobile Manipulator) is placed in a $5\ \mathrm{m} \times 5\ \mathrm{m} \times 2.5\ \mathrm{m}$ large room (see Fig. 2(d)). It has to perform the same task as the stationary manipulator, but the holes are further apart and the robot has to move its base to reach from one hole to the next.

Table I summarizes our results, averaged over 20 trials; numbers in parenthesis indicate the standard deviation. If a planner was not able to solve a problem within $20\ \mathrm{min}$, the experiment was aborted and we report the number of vertices, edges and collision detections up to that point. For the EET planner, the computation cost of workspace information based on the sphere-based wavefront expansion is negligible (never exceeded $0.5\ \mathrm{s}$) and is included in the total reported planning time. The parameters of the EET planner were set to $\alpha = 0.01$, $\beta = 0.08$, $\gamma = 18$.

Scenario a) All planners solve this problem within the time limit of $20\ \mathrm{min}$. The single query methods have the advantage of focusing on the correct area of the maze, but still spend considerable time exploring straight corridors instead of performing exploitation. The multi-query approaches waste computational resources exploring irrelevant configuration space regions but still perform better than classic single-query planners in this example. The EET planner is able to perform exploitation in the straight corridors. In the tighter turns, it shifts towards exploration. The resulting EET roadmap has far fewer vertices than that of any other method. The computed path exhibits less of the zigzaggy behavior commonly observed in sampling-based motion planning.

An interesting measure to assess the effectiveness of exploitation is the percentage of collision checks for which the tested configuration was in free space. For an ideal planner this percentage would be 100%, indicating that available information is leveraged to effectively guide the planner through the free configuration space. In this scenario, this percentage was 16% for RRTConnect and 26% for RRTConnect2. In comparison, 89% of the configurations checked by the EET planner were free of collision.

Scenario b) The EET planner is the only planner capable of solving this task every time in less than $20\ \mathrm{min}$. The RRTConnect variants with two trees achieve success rates between 85 and 90%. The reduced clearance around the robot makes it difficult for other tree-based methods to move through the straight, narrow corridors. The PRMs have difficulties placing valid samples in the corridors. Again, the EET planner benefits from its ability to balance exploitation and exploration. In the straight corridors, it performs exploitation using the workspace information. In the more difficult turns, the planner increases exploratory behavior, as indicated by the increased number of vertices and edges in the resulting roadmap. In this scenario, only 2% of the RRTConnect2's collision checks were placed in free configuration space, compared to 54% for the EET planner.

For this most difficult scenario, we tested an additional source of information for exploitation. As described in

TABLE I

COMPARISON OF PLANNER PERFORMANCE IN FOUR EXPERIMENTAL SCENARIOS (STANDARD DEVIATION IN PARENTHESES).

Scenario	Planner	Vertices		Edges		Collision Checks		Time (s)		%
	PRM	11,783	(4,560)	11,684	(4,529)	1,410,541	(428,159)	15.2	(6.0)	100
	Gaussian PRM	9,543	(2,526)	9,455	(2,508)	1,291,460	(275,173)	12.4	(3.2)	100
	Bridge PRM	9,338	(2,965)	7,764	(2,469)	2,114,102	(597,712)	18.4	(5.8)	100
(a)	RRTConnect	15,973	(4,783)	15,972	(4,783)	825,590	(258,647)	130.6	(52.6)	100
	RRTConnect2	10,366	(2,332)	10,364	(2,332)	441,148	(148,455)	50.0	(22.5)	100
	ADD-RRT	9,916	(3,374)	9,914	(3,374)	223,749	(74,377)	39.4	(24.7)	100
	EET	288	(49)	287	(49)	9,563	(919)	1.2	(0.1)	100
	PRM	350,157	(3,021)	339,582	(2,943)	57,981,363	(461,026)	1,200.0	(0.0)	0
	Gaussian PRM	369,347	(3,787)	358,269	(3,684)	63,948,945	(584,712)	1,200.0	(0.0)	0
	Bridge PRM	277,910	(649)	202,089	(548)	97,810,673	(185,560)	1,200.0	(0.0)	0
(b)	RRTConnect	12,797	(2,522)	12,796	(2,522)	7,601,936	(349,121)	1,200.0	(0.0)	0
	RRTConnect2	7,554	(3,499)	7,552	(3,499)	3,295,736	(3,984,754)	384.1	(455.1)	85
	ADD-RRT	5,672	(1,554)	5,670	(1,554)	294,238	(332,154)	179.3	(352.6)	90
	EET	942	(206)	941	(206)	18,874	(8,044)	4.6	(3.4)	100
	EETRepulsive	329	(263)	328	(263)	7,817	(2,420)	1.9	(0.7)	100
	PRM	143,934	(19,245)	143,813	(19,230)	8,948,412	(1,077,893)	1,158.3	(150.2)	10
	Gaussian PRM	2,633	(1,932)	2,613	(1,924)	438,030	(266,766)	43.6	(26.7)	100
	Bridge PRM	2,660	(1,989)	2,478	(1,866)	532,231	(328,777)	53.9	(33.3)	100
(c)	RRTConnect	286,521	(2,813)	286,520	(2,813)	10,282,837	(84,377)	1,200.0	(0.0)	0
	RRTConnect2	642	(218)	640	(218)	30,389	(17,034)	2.8	(1.5)	100
	ADD-RRT	893	(402)	891	(402)	33,109	(29,763)	3.0	(2.6)	100
	EET	46	(47)	45	(47)	2,957	(2,402)	1.0	(0.7)	100
	PRM	436	(617)	435	(617)	103,387	(128,881)	28.0	(34.3)	100
	Gaussian PRM	802	(1,583)	801	(1,582)	187,029	(319,531)	52.8	(90.6)	100
	Bridge PRM	585	(562)	543	(522)	166,499	(148,905)	45.6	(40.2)	100
(d)	RRTConnect	46,835	(16,712)	46,834	(16,712)	3,724,356	(1,276,819)	1,048.8	(371.6)	15
	RRTConnect2	185	(100)	183	(100)	9,914	(7,010)	2.2	(1.6)	100
	ADD-RRT	121	(66)	119	(66)	5,991	(5,309)	1.3	(1.2)	100
	EET	26	(12)	25	(12)	1,269	(215)	1.2	(0.2)	100

Section III, we combine the attractive force provided by the workspace spheres with repulsive forces from obstacles. This additional information should further guide exploration towards the correct direction in configuration space. Indeed, as indicated by the results reported for EETRepulsive, this additional information reduces the number of vertices and edges required to solve the planning problem, relative to EET. This demonstrates that the EET planner is able to balance exploration and exploitation adequately. When more accurate information is available for exploitation, the planner shifts further towards exploitative behavior. Collision checks performed by the EETRepulsive planner are 95% collision free, indicating that our planner leverages the available workspace information to perform highly effective configuration space search.

Scenario c) The PRMs with Gaussian and Bridge sampling are the only PRM variants able to solve this scenario reliably. The roadmap mainly covers the open configuration space regions around the robot's base. The resulting path was of poor quality. In contrast, the RRT variants with two trees are able to solve this scenario much more efficiently. The EET planner benefits from balancing exploitation and exploration. Exploitation enables the robot to withdraw from narrow passage quickly. The same is true for the insertion into the narrow passage. The resulting path is smooth, indicating that little unnecessary exploration was performed.

Scenario d) In this scenario, the PRM with uniform sampling succeeds in all trials. The RRTConnect variants with two trees solve this task faster than the third scenario. This is due to the fact that the robot has more degrees of freedom and a suitable direction for expansion is easier to find. The single tree variant is mostly unable to solve the narrow passage leading to the goal configuration. Again, the EET planner's ability to balance exploration and exploitation results in the best planning performance.

Summary: The behavior of PRM planners is governed by exploration. As a result, they waste computational resources on attempting to understand the entire configuration space. In our test scenarios, RRTConnect planners with two trees benefit from the fact that their exploration is seeded with the initial and final configuration of the robot, both of which are inside the only two narrow passages. Once the RRTConnect planners have found a way out of the narrow passage, the connect step (exploitation) allows them to solve scenarios 3 and 4 efficiently. However, as scenario 3 illustrates, the exploratory behavior of RRT variants has difficulties in finding paths into narrow passages. The failure of RRTConnect with a single tree to solve scenarios 3 and 4 supports this observation.

The EET planner outperforms all other planners in all of the experiments. It is able to overcome the difficulties of PRM-based and RRT-based planners by carefully balancing exploration and exploitation. Using workspace information, it performs exploitation without getting trapped in irrelevant regions of configuration space. The ability to balance exploration and exploitation during tree expansion based on the difficulty of local configuration space leads to highly effective motion planning. The paths generated by the EET

planner are of higher quality that those generated by other sampling-based planners.

We believe that the EET planner would fail or perform poorly for problems such as the alpha puzzle, where helpful workspace information is difficult to obtain. This is because the current version of the EET planner cannot perform pure exploration; its exploratory behavior is always guided by workspace information.

V. Conclusion

We propose a new categorization of sampling-based planning methods. This categorization relies on the concepts of exploration and exploitation, commonly used to characterize reinforcement learning techniques [3], [4]. Our discussion reveals that existing sampling-based motion planners do not deliberately balance between exploration and exploitation.

We argue that balancing exploration and exploitation is the most effective way of minimizing the amount of configuration space exploration required to solve a motion planning problem. We present an exploring/exploiting tree (EET) planner that deliberately balances exploration and exploitation based on information about the workspace and configuration space. Workspace information is used to avoid getting trapped globally. Information about specific regions of the configuration space is acquired during configuration space sampling. This information is used to adjust the balance between exploration and exploitation in accordance with the difficulty of the local planning problem. Our experimental results demonstrate that the balancing of exploration and exploitation performed by the EET planner leads to greatly improved planning performance in a variety of difficult, real-world motion planning problems.

Acknowledgments

We thank the anonymous reviewers for constructive comments and Yuandong Yang for helpful discussions about the definition of exploration and exploitation. We gratefully acknowledge support by the EU FP6 IST Cognitive Systems Integrated Project JAST (FP6-003747-IP) and by the National Science Foundation under grants CNS-0454074, IIS-0545934, CNS-0552319, and CNS-0647132.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[2] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. of the Symposium on Foundations of Computer Science*, 1979, pp. 421–427.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[4] S. Thrun, "The role of exploration in learning control," in *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, 1992.

[5] L. E. Kavraki, P. Švestka, J.-C. Latombe, and 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] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Intl. J. of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[7] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Technical Report 98-11, 1998.

[8] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Proc. of the Intl. Workshop on the Algorithmic Foundations of Robotics*, 1998, pp. 155–168.

[9] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 1999, pp. 1018–1023.

[10] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Journal of Advanced Robotics*, vol. 14, no. 6, pp. 477–494, 2000.

[11] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2003, pp. 4420–4426.

[12] M. Foskey, M. Garber, M. C. Lin, and D. Manocha, "A Voronoi-based hybrid motion planner," in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, vol. 1, 2001, pp. 55–60.

[13] C. Holleman and L. E. Kavraki, "A framework for using the workspace medial axis in PRM planners," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2000, pp. 1408–1413.

[14] J. P. van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *Int. J. of Robotics Research*, vol. 24, no. 12, pp. 1055–1071, 2005.

[15] Y. Yang and O. Brock, "Adapting the sampling distribution in PRM planners based on an approximated medial axis," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2004, pp. 4405–4410.

[16] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Proc. of Robotics: Science and Systems*, 2005, pp. 105–112.

[17] B. Baginski, "Local motion planning for manipulators based on shrinking and growing geometry models," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 1996, pp. 3303–3308.

[18] C. L. Nielsen and L. E. Kavraki, "A two level fuzzy PRM for manipulation planning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2000, pp. 1716–1722.

[19] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2000, pp. 521–528.

[20] J. Barraquand and J.-C. Latombe, "A Monte-Carlo algorithm for path planning with many degrees of freedom," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 1990, pp. 1712–1717.

[21] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2000, pp. 995–1001.

[22] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2007.

[23] L. Jaillet, A. Yershova, S. M. LaValle, and T. Siméon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2005, pp. 2851–2856.

[24] S. Rodríguez, X. Tang, J.-M. Lien, and N. Amato, "An obstacle-based rapidly-exploring random tree," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2006.

[25] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2001, pp. 1469–1474.

[26] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation," in *Proc. of Robotics: Science and Systems*, 2006, pp. 279–286.

[27] ——, "Efficient motion planning based on disassembly," in *Proc. of Robotics: Science and Systems*, 2005, pp. 97–104.

[28] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Proc. of Robotics: Science and Systems*, 2007.

[29] M. Stilman, "Task constrained motion planning in robot joint space," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2007, pp. 3074–3081.

[30] J. Kuffner, "Effective sampling and distance metrics for 3D rigid body path planning," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2004, pp. 3993–3998.

[31] A. Yershova and S. M. LaValle, "Improving motion-planning algorithms by efficient nearest-neighbor searching," *IEEE Trans. on Robotics and Automation*, vol. 23, no. 1, pp. 151–157, 2007.