Batch Informed Trees (BIT*)

Shankara Narayanan Vaidyanathan*, Sahasrajit Anantharamakrishnan*, Drake Moore*,
Thejaswini Goriparthi*

***Abstract*— Path Planning in Robotics has always relied on simple approximations to identify solutions. This is due to the difficulty to find one due to the high dimensional nature of the problem. Generally, we can divide the approximations into 2 types: Search-based and Sampling-based. Heuristics are used by search-based planners like A* to effectively search across graphs, but their efficiency is limited by the resolution of the selected approximation. To approximate the problem, sample-based planners such as RRT* employ random sampling. Here, the resolution can be increased until we find a suitable solution. These random samples approximate the region in all directions at the same time, making the search ineffective.**

A recent approach called Batch Informed Trees (BIT*) [1] combines the strengths of both Search-based and sampling-based planners. Heuristics and Sampling is used by BIT* to alternate between searching and approximating.

In this work, we have used the pseudo-code from the paper and coded the algorithm from scratch, and tested its performance in $\mathbb{R}^2$ space for different motion planning scenarios.

The code can be found in this link: https://github.com/marleyshan21/Batch-informed-trees

I. Introduction

Popular path planning algorithms generally belong to Search based planners such as Dijkstra's [2] and A* [3] or Sampling-based planners like RRT (Rapidly-exploring Random Trees) [4] and PRM (Probabilistic Roadmap) [5]. Both approaches have their strengths and limitations.

Search-based planners can exactly solve a discrete approximation of a search space. They always find the optimal solution at the chosen discretization if one exists. However, the quality of the solution and its approximation of a continuous space is dependent on the level of discretization chosen. Increasing the level of discretization leads to better solutions but also significantly increases the computational cost, making it impractical for high-dimensional spaces such as manipulation planning.

Sampling-based planners avoid the discretization problems of graph-search techniques. They sample states incrementally and avoid picking a fixed resolution before the start of the search. This incremental sampling makes the search order dependent, resulting in a randomly ordered search. As a result, while solutions improve with time, this does not ensure a fair convergence rate since random sampling is intrinsically unordered.

The research paper by Gammell et al. [1] introduces an algorithm for planning called Batch Informed Trees (BIT*), which combines the advantages of both search-based and sampling-based techniques. This motion planner has the following properties:

1) *Informed* - Its search is guided around the minimum solution proposed by a heuristic, as in A*.
2) *Asymptotically optimal* - As the number of samples goes to infinity, it converges asymptotically to the optimal solution with a probability equal to 1, if a solution exists.
3) *Anytime Search* - Before finishing the search for the optimal path, it can immediately find a suboptimal path.
4) *Anytime Resolution* - As the number of iterations rises, the representation of the problem domain becomes more accurate.
5) *Probabilistically complete* - When given unlimited samples, the probability that BIT* discovers a feasible optimal solution to a given path planning problem is one.

A. Batch Informed Trees

A state space is a collection of all possible states that a robot can be in during path planning. Path planning algorithms assume this space to be discrete. When this discretized state space is randomly sampled, the resulting set of states can be represented as a graph. Mathematically, this graph can be modelled as a Random Geometric Graph (RGG), where nodes represent states and edges represent transitions between those states. In an RGG, the nodes (states) are distributed randomly based on a probability distribution, and the edges (transitions) between nodes are determined by their geometric proximity. These types of RGG models have been extensively researched in the field of path planning.

In general, the working of BIT* can be explained according to Fig 1.

1) First, an initial random geometric graph (RGG) is created by connecting uniformly distributed random samples from the free space with the start and goal states.
2) Then, using a heuristic, an explicit tree is built from the start state to the goal state. Only edges that are collision-free are added to the tree, and construction stops when a solution is identified or the tree can no longer be expanded. [Fig. 1a]
3) Then, a new batch of samples is added to the existing RGG to make it denser. If a solution has already been found, the new samples are restricted to the subproblem that could contain a better solution, which is represented by an ellipse.
4) To limit the number of samples, the previous solution found is pruned from the tree, keeping only the

*Equal contribution to the work

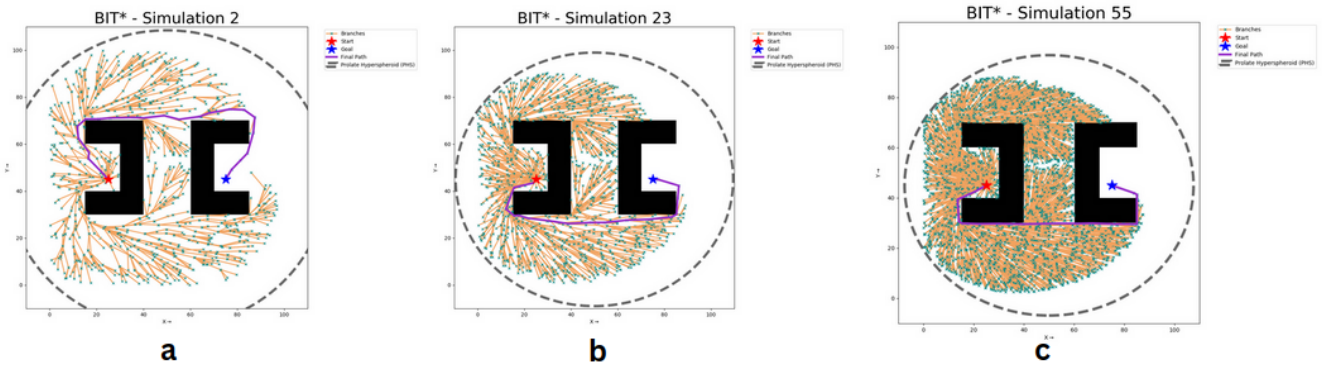

Fig. 1: **An example showing the general working of BIT*:** The start and goal states are represented in red and blue respectively. The current best path is highlighted in violet. (a) The initial batch of the search is directed by a heuristic and expands until a solution is identified. (b) The search continues after pruning the representation and increasing its accuracy with more samples inside a subproblem, represented by an ellipse. This search stops when an improved solution is found. (c) This process of searching and refining the search space continues until the globally optimal solution is found or until a certain time limit is reached.

branches that contribute to the suboptimal path.

5) With the pruned path, more samples are added within the ellipse, and the search is restarted. [Fig. 1b]
6) This process continues and restarts each time an improved solution is found. [Fig. 1c] The construction of the tree stops when the solution cannot be improved further. This could also be if there are no more collision-free edges to traverse, or when a set time limit is reached.

BIT* algorithm uses batches of samples to perform a search in an ordered manner on a continuous planning domain, while also maintaining anytime performance. Similar to A*, it uses a heuristic to guide its search towards the minimum solution. It also processes multiple batches of samples, enabling it to converge asymptotically to the globally optimal path and have anytime resolution like RRT*. Thus it is able to utilize the advantages of search algorithms that use heuristics to perform an ordered search and achieve anytime performance and asymptotic optimality through random sampling.

B. Our project

- The objective of this work is to gain an in-depth understanding of Batch Informed Trees (BIT*) and its performance.
- For this project we have written the BIT* code in python solely based on the pseudocode presented in the paper [1]. We have coded Algorithm 1 (BIT*), Algorithm 2 (Expand Next Vertex), and Algorithm 3 (Prune). Additionally, we have also implemented the informed sampling method mentioned in the paper using Algorithm 1 (Sample) and Algorithm 2 (SamplePHS) from [6].
- We have conducted a performance analysis of BIT* in a two-dimensional space ($\mathbb{R}^2$) for various path lengths using 5 different random seeds and 5 different well-known motion planning scenarios as described in [7].
- Furthermore, we have compared the performance of BIT* against other well-known sampling-based planners, such as RRT* [8] and RRT Connect [9], in terms of path length and execution time.

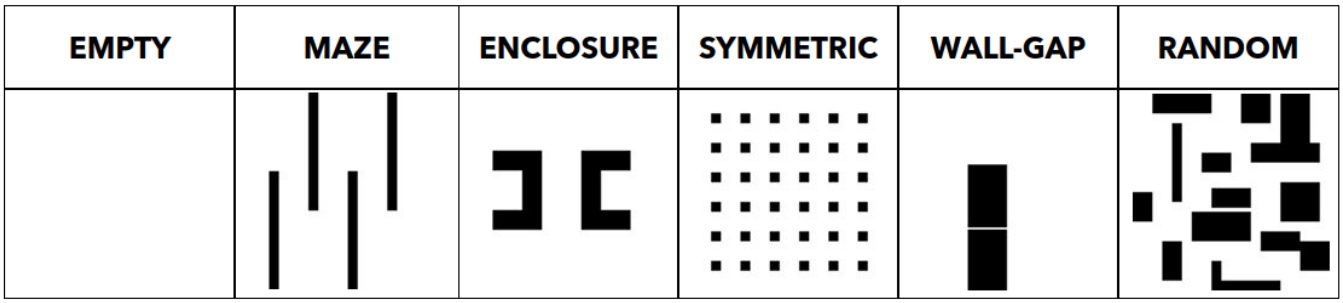

Fig. 2: An illustration of our planning scenarios with obstacles in black. The scenarios include an empty room setting (Empty), a maze, start and goal enclosures (Enclosure), regularly repeating obstacles (Symmetric), a scenario with a narrow gap (Wall-gap), and randomly generated obstacles (Random).

II. Experimental Setup

The planning problem is known to be difficult in various ways. To isolate specific challenges of motion planning, we have developed five planning scenarios as illustrated in Fig. 2, in accordance with the ones mentioned in Planner Developer Tools [7]. These scenarios are designed to help us understand the performance of the algorithms in various settings. These scenarios include:

1) **Empty:** This is a simple one without any obstacles. This helps us understand the basic performance of the algorithms in the simplest setting.
2) **Wall Gap:** This scenario involves a narrow gap, which helps us understand how well the sampling occurs in such spaces, and how the heuristic influences the search in finding the optimal solution that passes through the narrow gap.
3) **Maze:** A setting with a circuitous solution. This setting helps in breaking the alternating changes in cost and thus helps in understanding branching and time to the optimal solution.
4) **Enclosure:** This scenario has the goal and start enclosed by obstacles, which tests the performance of the algorithm in scenarios where the heuristic influences the path to go along the obstacle, and the optimal solution involves going against the Euclidean heuristic to find the optimal path with no collisions.
5) **Symmetry:** This scenario involves multiple repeating obstacles, and it helps us understand the behaviour of the algorithms in a more controlled environment. By creating regular obstacles, we can ensure that the algorithm always faces the same challenges, which makes it easier to identify potential problems.
6) **Random:** This scenario involves obstacles placed at random locations to test the robustness of the algorithms.

These environments are of size 100 pixels x 100 pixels.

III. Analysis

Batch Informed Trees algorithm was evaluated on different planning scenarios to assess its performance quantitatively and qualitatively. Additionally, BIT* was compared against RRT* and RRT-Connect to understand its efficiency in terms of path length and time taken. RRT-Connect is known to be faster in providing solutions, but they are not guaranteed to be optimal. On the other hand, RRT* is probabilistically complete, which means it can provide an optimal path given enough samples or a larger stop time. All experiments were conducted across all scenarios using five different random seeds, and the median time and path length were analyzed. The results of the evaluations can be found in Table. I.

A. Empty Scenario

In this scenario, there are no obstacles in the environment. The start was set to be at [0, 0] and the goal at [99, 99]. The start and goal positions are located at opposite corners of a 100x100 grid. Therefore, the shortest path is a diagonal line from the start to the goal. The resulting path generated by BIT* for this scenario is shown in Fig.3a. The actual path length between the start and goal points is 140.0071 units, and after limiting the planning time to 10 seconds, BIT* produced a median path length of 140.035 units.

B. Wall gap scenario

This scenario is designed to test the algorithm's ability to sample points in narrow spaces. The start is set at [0, 0], and the goal is set at [99, 0]. The environment has a narrow gap of only 1 pixel for the path to pass through. The start and goal are located at opposite ends of the image. The purpose of this scenario is to see how the algorithm performs in sampling points in such constrained spaces. The ground truth path length is 119.8807. BIT* was able to find a path close to the global optimum with a median path length of 120.4459 units after restricting the planning time to 90 seconds. BIT*'s output for this scenario is shown in Fig. 3b.

RRT* and BIT* took a similar amount of time, around 74.955 seconds, to reach a path close to the global optimum. However, RRT-Connect was significantly faster, finding a solution in just 3.39 seconds. But this solution had a median path length of 175.513, indicating that it was not as optimal as the solutions found by RRT* and BIT*.

C. Maze Scenario

This scenario involves obstacles in a lawn mower-style pattern, and it allows us to evaluate the algorithm's effectiveness in dealing with alternating changes in costs. The starting point is [0, 0], and the goal is [99, 99]. The output of BIT* can be seen in Fig. 3c, and the ground truth path length is 218.83. After setting a planning time limit of 60 seconds, BIT* was able to achieve a median path length of 222.9786, which is close to the global optimum.

BIT* outperformed RRT* in this scenario by taking only 53.259 seconds to reach close to the global optimum. In contrast, RRT* took 76.65 seconds to converge. Interestingly, RRT-Connect was unable to find a path in any of the 10 trials due to the environment's construction.

Experiment	Ground Truth	BIT*		RRT Connect		RRT*	
		Time taken (s)	Path length	Time taken (s)	Path length	Time taken (s)	Path length
Wall Gap	119.8807	64.6911	120.445	3.3915019	175.513	74.95518	121.296
Maze	218.83	53.259	222.979	Failed	Failed	76.65529	225.338
Enclosure	112.3606	80.4461	113.977	8.1440262	159.372	121.39378	114.087
Symmetry	60.77	39.1863	60.838	2.0315244	68.377	437.24997	62.495
Random	N/A	72.9016	82.378	5.2485	106.566	95.560036	83.082

TABLE I: **Comparison of median path lengths and time taken**: Evaluations of BIT*, RRT-connect and RRT* resulted in these results.

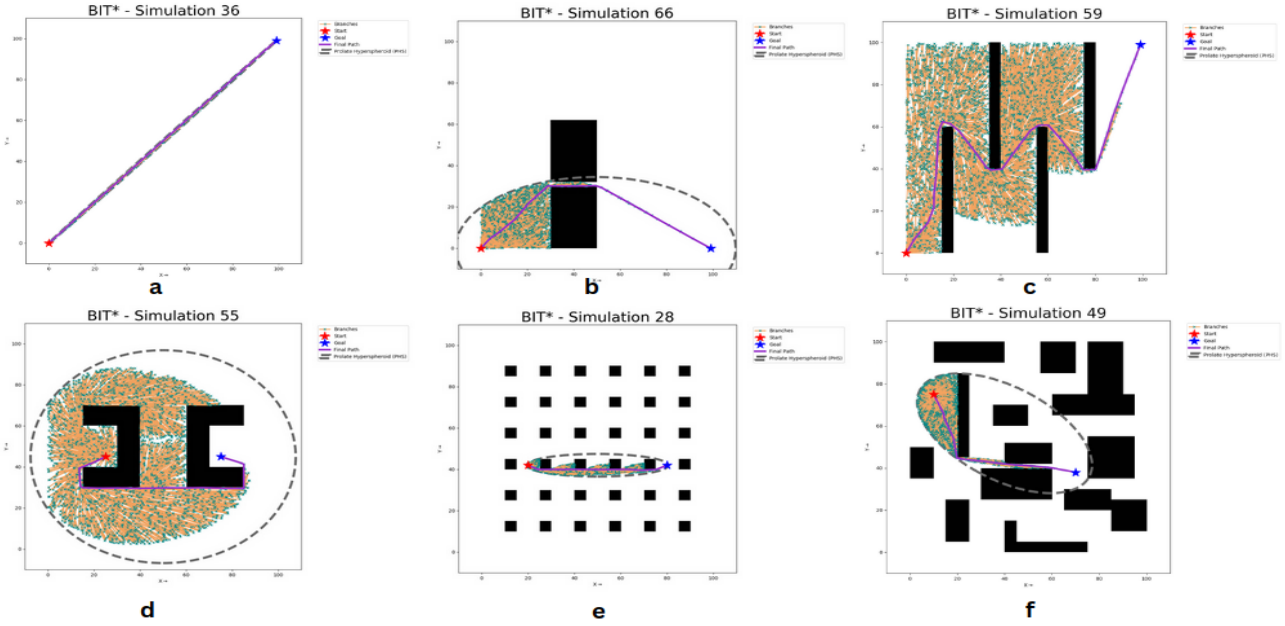

Fig. 3: An illustration of the final path calculated by BIT* in all our planning scenarios. (a) Empty, (b) Wall gap, (c) Maze, (d) Enclosure, (e) Symmetric, and (f) Random.

D. Enclosure Scenario

This scenario involves a situation where the start and goal are surrounded by obstacles. It helps in analyzing the behavior of the algorithm when the heuristic-influenced path would be along the obstacles, but the algorithm has to move against the direction of the goal location to avoid colliding with the obstacles. The start is located at [25, 45], and the goal is located at [75, 45]. The output of BIT* for this scenario is shown in Fig.- 3d. The ground truth path length for this scenario is 112.3606 units. BIT* provided a median path length of 115.1904 units after restricting the planning time to 90 seconds. This is close to the global optimum.

In this scenario, RRT* was able to find the global optimum path length, but it took 121.3938 seconds to converge. On the other hand, BIT* was able to perform better than RRT* by converging to the global optimum in 80.44 seconds. Although RRT-Connect found a solution faster, it had a median path length of 159.372 units, which is significantly higher than the ground truth path length of 112.3606 units. Thus, BIT* was able to strike a balance between time and path length and performed better than RRT-Connect in this case.

E. Symmetry Scenario

The Symmetry scenario involves regularly spaced obstacles throughout the environment, making it a useful test of the fundamental capabilities of a motion planning algorithm. The starting point is at [20, 42] and the goal is at [80, 42]. BIT*'s output for this scenario is shown in Fig. 3e, with a ground truth path length of 60.77 units. After restricting the planning time to 60 seconds, BIT* achieved a median path length of 60.838 units, indicating that it successfully found the global optimum.

In the same scenario, RRT* was able to achieve the optimal path length but took significantly more time than BIT* to complete. The median time taken by RRT* was 437.24997 seconds while it was only 39.18 seconds for BIT*. However, in comparison to RRT-Connect, BIT* was not able to outperform RRT-Connect. RRT-Connect was able to find a solution in just 2.031 seconds with a median path length of 68.377 units, which was slightly worse than the ground truth path length of 60.77 units. Therefore, in this scenario, RRT-Connect performed better than both BIT* and RRT*.

F. Random Scenario

In this scenario, the obstacles are randomly placed to test the algorithm's robustness. The start is [10, 75] and the goal is [58, 65]. As the obstacles are randomly generated, there is no ground truth available. Its output can be observed in Fig. 3f. BIT* algorithm gave a median path length of 82.6324 units, and it took 72.9016 seconds to reach a value that remained mostly unchanged for a considerable period, indicating that it is close to the global optimum. RRT* took 95.56 seconds, and RRT Connect took just 5.2485 seconds. However, the median path length of RRT Connect is 106.566 units, which is longer than BIT*.

IV. Conclusion

The experiments conducted in this work showed that BIT* converges asymptotically to the optimal solution in all the scenarios. This can be attributed to the sampling method used in BIT*, which allows for better solutions to be found inside specific sub-problems. Although RRT-Connect was

fast, it produced suboptimal solutions in most cases. Overall, BIT* balances the benefits of heuristically guided search with anytime performance and asymptotic optimality. This study demonstrates that a heuristically guided tree construction, as used in search-based planners like A*, greatly improves the performance of the algorithm, and that BIT* is a good option for balancing time and path length.

V. Future Work

The paper also presented additional features for the BIT* algorithm, including Just-in-time sampling and delayed rewiring. These could be worth exploring in future experiments during the summer. The authors also suggested that BIT* would perform even better in higher dimensions such as R^7 and R^{16}, which could be explored. However, we faced some issues with the Python bindings of OMPL in Pybullet, which prevented us from testing in these spaces. Resolving these issues could enable further exploration of the algorithm's performance in higher dimensions.

Acknowledgement

This project was made possible with the exceptional support of Professor Micheal Everett.

References

[1] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[4] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[5] L. E. Kavraki, P. Svestka, 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] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, 2018.

[7] J. D. Gammell, M. P. Strub, and V. N. Hartmann, "Planner developer tools (pdt): Reproducible experiments and statistical analysis for developing and testing motion planners," in *Proceedings of the Workshop on Evaluating Motion Planning Performance (EMPP), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.