BABIT*: a Bidirectional Advanced BIT* for Fast Path Planning with Implicit Random Geometric Graph

Yihang Xiang[a], Jiajun Ouyang[a], Zhen Huang[a], Hui Yu[b], Junyu Dong[a], and Shu Zhang*[a]

[a]College of Computer Science and Technology, Ocean University of China, Qingdao, China
[b]School of Creative Technologies, University of Portsmouth, Portsmouth, PO1 2DJ, UK

ABSTRACT

Path planning is an active and essential research field for many applications with autonomous mobile robotics. However, popular technologies have limitations in situations where robots require onboard computing to work independently. To this end, this paper proposes a bidirectional advanced batch information tree (BABIT*), which is an asymptotically optimal algorithm path planner enhanced from Batch Informed Trees (BIT*). It uses an edge queue sorted by inflated potential path cost to guide the search of implicit random geometry graph (RGG) to generate explicit solutions while minimizing height calculation tasks such as collision checking. BABIT* promotes the exploration of the entire state space by adopting a more reasonable sampling strategy to achieve a more uniform and decentralized approximation of problems, and ensures a faster discovery of solutions by using symmetric bidirectional search for the state space from both directions. The experimental results show that BABIT* outperforms existing single-query, sampling based planners on the tested problems.

Keywords: path planning, sampling-based planning algorithms, asymptotically optimal planning

1. INTRODUCTION

Reasonable path planning in the environment is of great significance for robots to safely reach their destination and avoid possible obstacles.Popular techniques for path planning include graph based search methods such as A*[1]and Dijkstra's,[2] as well as sampling based methods such as RRT[3] and PRM.[4] However, these methods have limitations in situations where robots require onboard computing to work independently. Usually, compared to regular desktop computers, these on-board computing capabilities are very limited. Those methods will spend more on calculation due to the need for environmental discretization and random sampling respectively.

In order to solve the problem of limited onboard computing capabilities, this paper proposes a path planning algorithm called BABIT*, which is an extension of BIT*.[5,6]We show that through bidirectional search and subspace sampling, we find an optimal solution that competes with the convergence speed of A BIT*[7] and the initial path time of RRT-Connect.[8]

2. RELATED WORK

RRT[3] grows a tree from the start to incrementally approximate the state space. The approximation and search are still coupled by the incremental sampling which also results in a randomly search. RRT Connect[8] grows trees from both start and goal, and biases their growth towards unexplored spaces and each other. Greedy tree connections usually result in fast initial solution times, but the searche is still randomly sorted.

But the solutions provided by these methods are almost impossible to be optimal. RRT*[9] extends RRT with locally rewiring to provide almost-surely asymptotically optimal solutions. Its solutions converge to the optimal with probability one given infinite computational time. For improving the convergence rate of RRT*, Informed RRT*[10] and RRT* Smart[11] reduces the sampling space by different sampling strategies,and Quick RRT* (QRRT*)[12] reduces the number of nodes in the tree through improved locally rewiring. But they still maintain the random search order.

Approximation and search can be separated by processing batches of multiple states, and ordered search can be achieved through heuristic methods. BIT*[5,6] uses the sampling of Informed RRT* to sample multiple batch

states, and views the resulting approximation as a gradually dense implicit RGG.[13] Then BIT* uses incremental search to handle possible edges based on heuristic sorting for its potential solutions, similar to Lifelong Planning A* (LPA*).[14] And Adaptive Information Trees (AIT*)[15] adjust the heuristic method more accurately, thereby saving some computational time.

Of course, there are other improved algorithms for BIT*. RABIT*[16] uses obstacles information and gradient descent method called CHOMP[17] to find potential better paths around obstacles, which accelerates convergence speed. Fast BIT*[18] changes the initial solution search sorting strategy to quickly find an initial solution. But even if the approximation is about to change, they still need to find the optimal resolution path.

Advanced BIT* (ABIT*)[7] adopts a truncation strategy to start a new batch in advance, and uses an inflation strategy to inflate the heuristic, thereby obtaining faster initial solutions and convergence. Greedy BIT* (GBIT*)[19] uses a greedy strategy to speed up the initial solution search. They avoid the computational cost of finding an optimal resolution path in potentially changing inaccurate approximations.

3. BIDIRECTIONAL ADVANCED BIT* (BABIT*)

BABIT* maintains a gradually dense RGG through batch sampling to obtain an approximation of the environment, and finds the next potential optimal cost edge for orderly processing through current costs and inflated heuristic costs. And BABIT* enhances the reusability of sampling states through bidirectional search and different sampling strategies, further reducing the computational complexity of search and making it more suitable for independent calculations on board robots.

Algorithm 1: Bidirectional ABIT*(x_{start}, x_{goal}, m)

$V_s \leftarrow \{x_{start}\}; E_s \leftarrow \emptyset; T_s \leftarrow \{V_s, E_s\}; X_{unconnected_s} \leftarrow x_{goal}$
$V_g \leftarrow \{x_{goal}\}; E_g \leftarrow \emptyset; T_g \leftarrow \{V_g, E_g\}; X_{unconnected_g} \leftarrow x_{start}$
$q_s \leftarrow |V_s| + |X_{unconnected_s}|; \varepsilon_{infl_s} \leftarrow \infty; \varepsilon_{trunc_s} \leftarrow \infty; V_{closed_s} \leftarrow \emptyset; V_{inconsistent_s} \leftarrow \emptyset$
$q_g \leftarrow |V_g| + |X_{unconnected_g}|; \varepsilon_{infl_g} \leftarrow \infty; \varepsilon_{trunc_g} \leftarrow \infty; V_{closed_g} \leftarrow \emptyset; V_{inconsistent_g} \leftarrow \emptyset$
$Q_s \leftarrow \emptyset; Q_g \leftarrow \emptyset; c_{best} \leftarrow \infty; batch\_search\_finished \leftarrow True$
repeat
  if $batch\_search\_finished$ **then**
    forall T *in* T_s, T_g
      if $isNeedApproximation(\{T_s, T_g\})$ **then**
        $Prune(T, X_{unconnected}, c_{best})$
        $X_{unconnected} \xleftarrow{+} Sample(m, c_{best})$
        $q \leftarrow |V| + |X_{unconnected}|; Q \leftarrow expand(\{x_{start}\}, T, X_{unconnected}, r(q))$
      else if $isFinalSearch()$ **then**
        $Q \xleftarrow{+} expand(V_{inconsistent}, T, X_{unconnected}, r(q))$
      $\varepsilon_{infl} \leftarrow update\_inflation\_factor()$
      $\varepsilon_{trunc} \leftarrow update\_truncation\_factor()$
      $V_{closed} \leftarrow \emptyset; V_{inconsistent} \leftarrow \emptyset$
    $batch\_search\_finished \leftarrow False$
  else
    $(x_p, x_c) \leftarrow \underset{(x_i, x_j) \in Q_s}{arg\,min} \{g_{T_s}(x_i) + \hat{c}(x_i, x_j) + \varepsilon_{infl_s}\hat{h}(x_j)\};\ Q_s \xleftarrow{-}(x_p, x_c)$
    $ProcessEdge(T_s, x_p, x_c, Q_s)$
until *stop*;

3.1 Notation

The state space of the planning problem is denoted by X, the start state by $x_{start} \in X$, and the goal states by $x_{goal} \in X$. The current search is stored in the form of trees, divided into two trees from the start and target directions, $T_s = (V_s, E_s)$, $T_g = (V_g, E_g)$, respectively representing vertices V_s, V_g, and edges $E_s \subset V_s \times V_s$, $E_g \subset V_g \times V_g$. Vertices in the tree are associated with valid states and edges in the tree represent valid connections between states. An edge consists of a parent state x_p and a child state x_c, and is denoted as (x_p, x_c). The set of inconsistent states of two trees is denoted by $V_{inconsistent_s}$, $V_{inconsistent_g}$ and the state sets that are not in the start tree or target tree are represented by $X_{unconnected_s}$ and $X_{unconnected_g}$, respectively. The cost of the current optimal solution is represented by c_{best}.

We use some functions to map all states to non negative real numbers representing state costs (including infinity). The function $\hat{g}(x)$ represents the acceptable estimated cost (i.e. lower bound) from the starting state to the current state. The function $g_T(x)$ represents the current actual cost of reaching the status in the current Search tree. If status x is not connected to the current Search tree, $g_T(x)$ is set to infinity. The function $\hat{h}(x)$ represents an acceptable heuristic, which is also the estimated cost from the current state x to the target state x_{goal}.The function $\hat{f}(x)$ represents the estimated cost from the starting state to the target state through the current state x. $\hat{g}(x) = \hat{g}(x) + \hat{h}(x)$. This function defines the set of state information that can improve the current solution. The function $\hat{c}(x_p, x_c)$ represents the acceptable estimated cost of an edge, while the function $c(x_p, x_c)$ represents the true cost of the edge. $\lambda(\cdot)$ Lebesgue measure representing a group of states, ζ_n represents the Lebesgue measure of the n-dimensional unit ball. m is the number of sampling states per batch.

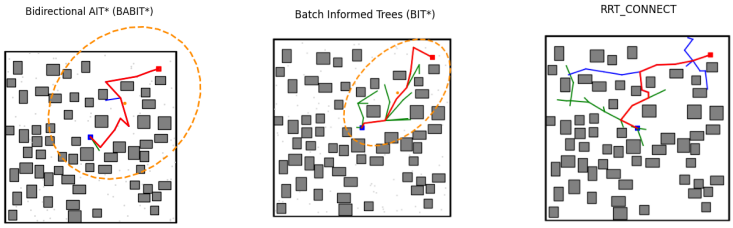
3.2 Initialization and Approximation

BABIT* requires the initialization of search trees in both directions: initialize the search tree T_s with the starting state as its root, and the unconnected state set $X_{unconnected_s}$ initially only contains the target state; Similarly, the search tree T_g is initialized with the target state as its root, and the unconnected state set $X_{unconnected_g}$ initially only contains the starting state. Expansion factor $\varepsilon_{infl_{start}}, \varepsilon_{infl_{goal}}$ and truncation factor $\varepsilon_{trunc_{start}}, \varepsilon_{trunc_{goal}}$ are initialized to be infinity large.The edge-queue Q_s, Q_g are initialized to be empty.

The sampling of BABIT* is related to whether there is currently a solution. Before finding the initial solution, BABIT * uses subspace sampling to divide the state space defined by the current planning problem into multiple subspaces, and samples each subspace one by one. After finding a initial solution, BABIT* uses informed sampling to concentrate the sampling space on the space that may improve the current solution. However,as the number of sampling states increases, the computational complexity also increases. This complexity is reduced by pruning states that cannot improve the current solution and shrinking the connection radius as more states are sampled.The formula for updating the radius is as follows

$$radius = 2\eta\Big(1+\frac{1}{n}\Big)^{\frac{1}{n}}\Big(\frac{\lambda(X_{\hat{f}})}{\zeta_n}\Big)^{\frac{1}{n}}\Big(\frac{log(q)}{q}\Big)^{\frac{1}{n}} \tag{1}$$

Where q is the number of sampled states in the informed set,$\eta \geq 1$ is a tuning parameter, n is the number of dimensions, and ζ_n is the Lebesgue measure, or n-dimensional volume.



(a) BABIT* (b) BIT* (c) RRT Connect

Figure 1. Results for BABIT*, BIT* and RRT Connect in 2D.

3.3 Processing Edges

Edge processing is shown in Alg. 2. It uses edge queues to select the best next edge for expansion, similar to Anytime Truncated D* (ATD*).[20] This queue is arranged in dictionary order based on potential solution costs (using an inflated heuristic) and future costs. In the search process of BABIT*, two Search tree in two directions are used to complete the search, and the search iteration logic of each Search tree is the same. The search iteration starts by removing the edge with the lowest queue value from the edge queue. If the edge is part of the Search tree, the sub state will be expanded (that is, its outgoing edge will be added to the queue) if the sub state has not been expanded during the current search. Otherwise, it will be checked whether the new edge may contribute to a better solution than the current edge.

If BABIT* obtains the next edge to be processed, it will check whether the edge can improve the current solution and the cost of its sub state, and then check for collisions with that edge. If the child state is already part of the tree, BABIT* will change the connection from the original parent state to the new parent state, also

known as rewire in RRT*. Otherwise, the status will be deleted from the unconnected status set and added to the Search tree. After adding an edge, the sub state will be expanded unless it has already been expanded during the current search, in which case it will be added to the set of inconsistent vertices.

If the optimal cost edge of the search queue cannot improve the current solution, BABIT* exchanges two search trees and uses another tree for search iteration. When the expected boundary for resolution optimization is reached (i.e., there are no edges in the search queues of both search trees that can improve the current solution), the approximation is updated, depending on the expansion and truncation factors.

Algorithm 2: $ProcessEdge(T, x_p, x_c, Q)$

if $(x_p, x_c) \in E$ **then**
  if $x_c \in V_{closed}$ **then**
    $V_{inconsistent} \xleftarrow{+} \{x_c\}$
  else
    $Q \xleftarrow{+} expand(\{x_c\}, T, X_{unconnected}, r(q))$
else if $\varepsilon_{trunc}(g_T(x_p) + \hat{c}(x_p, x_c) + \hat{h}(x_c)) \leq c_{best}$ **then**
  if $g_T(x_p) + \hat{c}(x_p, x_c) < g_T(x_c)$ **then**
    if $g_T(x_p) + c(x_p, x_c) + \hat{h}(x_c) < c_{best}$ **then**
      if $g_T(x_p) + c(x_p, x_c) < g_T(x_c)$ **then**
        if $x_c \in V$ **then**
          $E \xleftarrow{-} \{(x_{prev,x_c} \in E)\}$
        else
          $X_{unconnected} \xleftarrow{-} x_c; V \xleftarrow{+} x_c$
        $E \xleftarrow{+} (x_p, x_c)$
        if $x_c \in V_{closed}$ **then**
          $V_{inconsistent} \xleftarrow{+} x_c$
        else
          $Q \xleftarrow{+} expand(\{x_c\}, T, X_{unconnected}, r(q)); V_{closed} \xleftarrow{+} x_c$
        $c_{best} \leftarrow CalculateShortestPathLengh(V_s, V_g)$
else
  $batch\_search\_finished \leftarrow True; swapTrees(T_s, T_g)$

3.4 Inflation and Truncation

The two search trees of BABIT* have their own inflation and truncation factors, which will be updated after approximation. A large inflation factor can bias the search towards the target and accelerate the search, but it may lead to deviations in the quality of the solution. A smaller inflation factor requires more computational work to complete the search, but maintains stricter solution quality. A large truncation factor promotes the exploration of regions in the state space, which helps to add more samples. A small truncation factor helps to utilize the current approximation of the state space. These factors decrease as the number of vertices in the tree and the unconnected state set increases, attempting to obtain the optimal resolution solution.

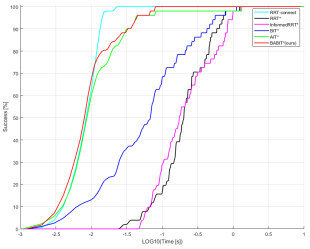
4. EXPERIMENTS AND EVALUATIONS

This paper uses Definition 2 4 i n^9 a s t he a lmost c ertain d efinition of as ymptotic op timality. Th e sufficient condition for proving that a sampling based planner is almost always asymptotically optimal is that (i) the base graph almost always contains an asymptotically optimal path, (ii) the base graph search is resolution optimal.

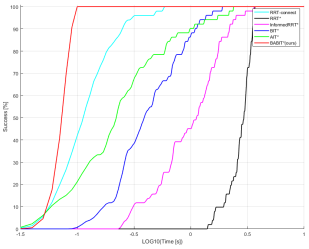
BABIT* used subspace sampling during finding the initial solution, and changed to using informed sampling after finding a initial solution. Due to the fact that searching for the initial solution does not require consideration of convergence and subsequent use of informed sampling similar to ABIT*, BABIT* uses the same increasingly dense RGG approximation as ABIT*. Since ABIT* is almost certainly an asymptotically optimal algorithm, this approximation almost certainly includes an asymptotically optimal path. According to the definition and proof of ABIT*, if the product of the expansion factor and truncation factor approaches 1 and q approaches infinity, BABIT* can asymptotically find the optimal resolution solution.In summary, we can obtain similar properties:

$$P\Big(\lim_{q\to\infty} sup\, c_{best,q}^{BABIT*} = s*\Big) = 1 \tag{2}$$

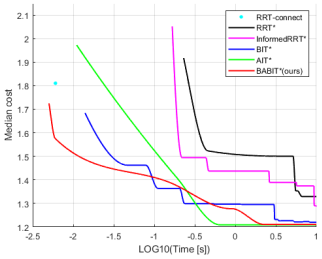
To test the performance in different problems, we compared BABIT* with previous sampling based asymptotic optimal programming algorithms ABIT*,[7] BIT*,[6] RRT*,[9] Informed RRT*,[10] and fast non optimal planner RRT Connect.[8] The goal of these simple optimal planning algorithms is to minimize path length.



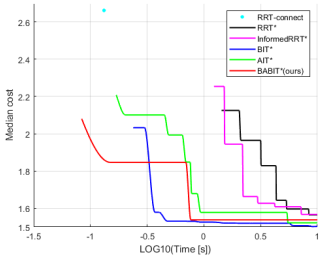
(a) 2D: Success rate vs. time



(b) 3D: Success rate vs. time



(c) 2D: Median cost vs. time



(d) 3D: Median cost vs. time

Figure 2. Results from different planners in 2D and 3D.

The RGG constant of all planners is set to 1.1. The RRT based planner is set with a target deviation of 0.05 and a maximum edge length of 0.2 (2D) and 0.25 (3D). The planner based on BIT* is set to have 100 batches and Euclidean distance is heuristic. For inflation and truncation factors, first use a highly inflated heuristic, $\varepsilon_{infl} = 10^6$, then use a lower factor again,$\varepsilon_{infl} = 1 + 10/q$. All searches use a single truncation factor,$\varepsilon_{trunc} = 1 + 5/q$. The collision check for each planner is set to 0.01. The calculation time of each asymptotic optimal planner is limited to 10 seconds.

We tested the algorithms using random environmental maps in 2D and 3D. This is similar to the original BIT* test.[5] These environments involve random world cubes with a width of 2 through each dimension. 10 random worlds were created in each scenario and tested using 50 random seeds. The environment is filled with obstacles consisting of rectangles and cubes, and is set to fill 30% of the environment.

These experiments indicate that in 2D (Fig: 1 2a, 2c) and 3D (Fig: 1 2b, 2d), BABIT* typically finds better solutions faster than other sampling based optimal planners. Compared to these planners, it has a higher likelihood of finding a solution within a given computational time and converges to the optimal value faster. In addition, we also conducted the same test using raspberry pie and the results showed that BABIT* was also able to find the initial solution faster. Therefore, BABIT* is a very ideal path planning algorithm applied to automated robots with limited computing resources.

5. CONCLUSIONS

BABIT* shows that by using bidirectional search and subspace sampling, you can find the initial solution faster and more stably than ABIT*, and even faster than RRT Connect in most cases. This allows BABIT* to use informed sampling earlier, resulting in faster overall rate of convergence.

If the search process in one direction is truncated, BABIT* can find or improve a solution by searching in the other direction. This can reduce the number of samples and make it more likely to find solutions when there are fewer sampling points, thereby reducing computational complexity and better applying to onboard calculations.

ACKNOWLEDGMENTS

The research was supported by the Natural Science Foundation of China (41927805, 41906177), the Grant for the National Key R&D Program of China (2022ZD0117201), the Hainan Provincial Joint Project of Sanya Yazhou

Bay Science and Technology City (2021JJLH0061), and the Shandong Provincial Natural Science Foundation, China (ZR2023MF012).

REFERENCES

[1] Hart, P. E., Nilsson, N. J., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics* **4**(2), 100–107 (1968).

[2] Dijkstra, E. W., "A note on two problems in connexion with graphs," in [*Edsger Wybe Dijkstra: His Life, Work, and Legacy*], 287–290 (2022).

[3] LaValle, S. M. and Kuffner Jr, J. J., "Randomized kinodynamic planning," *The international journal of robotics research* **20**(5), 378–400 (2001).

[4] Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation* **12**(4), 566–580 (1996).

[5] Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D., "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in [*2015 IEEE international conference on robotics and automation (ICRA)*], 3067–3074, IEEE (2015).

[6] Gammell, J. D., Barfoot, T. D., and Srinivasa, S. S., "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research* **39**(5), 543–567 (2020).

[7] Strub, M. P. and Gammell, J. D., "Advanced bit (abit): Sampling-based planning with advanced graph-search techniques," in [*2020 IEEE International Conference on Robotics and Automation (ICRA)*], 130–136, IEEE (2020).

[8] Kuffner, J. J. and LaValle, S. M., "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)*], **2**, 995–1001, IEEE (2000).

[9] Karaman, S. and Frazzoli, E., "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research* **30**(7), 846–894 (2011).

[10] Gammell, J. D., Barfoot, T. D., and Srinivasa, S. S., "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics* **34**(4), 966–984 (2018).

[11] Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., and Muhammad, M. S., "Rrt*-smart: A rapid convergence implementation of rrt," *International Journal of Advanced Robotic Systems* **10**(7), 299 (2013).

[12] Jeong, I.-B., Lee, S.-J., and Kim, J.-H., "Quick-rrt*: Triangular inequality-based implementation of rrt* with improved initial solution and convergence rate," *Expert Systems with Applications* **123**, 82–90 (2019).

[13] Dall, J. and Christensen, M., "Random geometric graphs," *Physical review E* **66**(1), 016121 (2002).

[14] Koenig, S., Likhachev, M., and Furcy, D., "Lifelong planning a$*$," *Artificial Intelligence* **155**(1-2), 93–146 (2004).

[15] Strub, M. P. and Gammell, J. D., "Adaptively informed trees (ait): Fast asymptotically optimal path planning through adaptive heuristics," in [*2020 IEEE International Conference on Robotics and Automation (ICRA)*], 3191–3198, IEEE (2020).

[16] Choudhury, S., Gammell, J. D., Barfoot, T. D., Srinivasa, S. S., and Scherer, S., "Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning," in [*2016 IEEE International Conference on Robotics and Automation (ICRA)*], 4207–4214, IEEE (2016).

[17] Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S., "Chomp: Gradient optimization techniques for efficient motion planning," in [*2009 IEEE international conference on robotics and automation*], 489–494, IEEE (2009).

[18] Holston, A. C., Kim, D.-H., and Kim, J.-H., "Fast-bit*: Modified heuristic for sampling-based optimal planning with a faster first solution and convergence in implicit random geometric graphs," in [*2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*], 1892–1899, IEEE (2017).

[19] Chen, L., Yu, L., Libin, S., and Jiwen, Z., "Greedy bit*(gbit*): greedy search policy for sampling-based optimal planning with a faster initial solution and convergence," in [*2021 International Conference on Computer, Control and Robotics (ICCCR)*], 30–36, IEEE (2021).

[20] Aine, S. and Likhachev, M., "Truncated incremental search," *Artificial Intelligence* **234**, 49–77 (2016).