2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

The Virtues of Laziness: Multi-Query Kinodynamic Motion Planning with Lazy Methods

Anuj Pasricha* and Alessandro Roncone

***Abstract*— In this work, we introduce *LazyBoE*, a multi-query method for kinodynamic motion planning with forward propagation. This algorithm allows for the simultaneous exploration of a robot's state and control spaces, thereby enabling a wider suite of dynamic tasks in real-world applications. Our contributions are three-fold: i) a method for discretizing the state and control spaces to amortize planning times across multiple queries; ii) lazy approaches to collision checking and propagation of control sequences that decrease the cost of physics-based simulation; and iii) *LazyBoE*, a robust kinodynamic planner that leverages these two contributions to produce dynamically-feasible trajectories. The proposed framework not only reduces planning time but also increases success rate in comparison to previous approaches.**

I. Introduction

Robotic manipulation in complex operational environments necessitates the integration of constraints at various levels of abstraction (i.e., kinematics, statics, quasi-statics, and dynamics) in order to effectively govern the interaction between the robot and its surrounding environment [1], [2]. Importantly, dynamics extends the foundational principles of kinematics, statics, and quasi-statics by incorporating the analysis of forces and torques in robot and object motion in real-world tasks such as liquid transport [3], deformable object manipulation [4], and nonprehensile maneuvers [5], [6] (Fig. 1).

The complexities (and opportunities) introduced by dynamic-based analysis of motion planning set the stage for kinodynamic planning, a class of methods that incorporate these dynamic constraints into planning, consequently extending classical kinematic or geometric planning methods beyond the robot's state space to its control space [7], [8]. Within this scope, kinodynamic planning frameworks can be broadly classified into two paradigms: *sequential* and *interleaved*. Sequential approaches engage first in geometric path planning and then in time parameterization, a sequence that may result in dynamically-infeasible trajectories and long planning times [9], [10]. In contrast, interleaved methods present a more integrated solution by enabling simultaneous exploration of state and control spaces, thereby ensuring dynamically-valid trajectories, if they exist [11], [12]. It is worth noting that a majority of methods in the interleaved domain are *single-query* methods. In a single-query context, each new planning problem necessitates the reconstruction of the planning tree, thereby limiting computational efficiency. This issue becomes of particular importance in kinodynamic planning due to the higher dimensional space involved. To this end, our work introduces a multi-query approach in the interleaved planning domain, aimed at reducing computation times across multiple planning queries. This contribution obviates the need to re-explore the state and control spaces and their corresponding search trees for each new task, offering a more efficient solution.

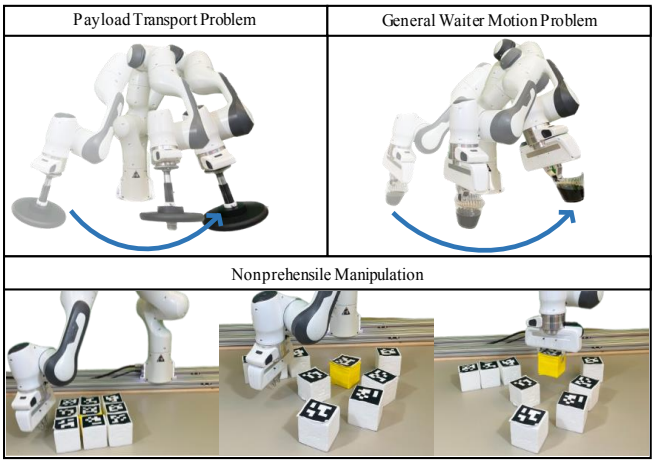

Fig. 1: Several applications motivate the need for considering dynamic constraints in motion planning. The payload transport problem requires the robot to account for the added mass at its end-effector and its effects on the inertia, Coriolis, and gravity matrices in the robot dynamics model [13]. Similarly, liquid transport in the General Waiter Motion Problem imposes acceleration constraints on the end-effector to ensure spill-free trajectories [3]. Nonprehensile actions like poking can be used to singulate target objects and require an understanding of combined robot and object dynamics [5]. In this paper, we present a method for planning robot paths while considering its dynamic constraints.

More specifically, our planner is conceptually aligned with the bundle of edges (BoE) framework, which adapts the probabilistic roadmap method from a kinematic setting to the kinodynamic domain [14]. The BoE planner employs a forward search through disconnected, randomized rollouts of a dynamics model [15]. This process is computationally demanding due to a large branching factor and is contingent upon both accurate simulation models and reliable collision checkers—assumptions that may not hold in complex, real-world conditions. Crucially, it is these very processes—forward propagation and collision checking—that contribute significantly to the computational burden of the planner. This underscores the need for '*lazy*' strategies that postpone these operations until absolutely required, thus achieving a

* Corresponding author.
The authors are with the Department of Computer Science, University of Colorado Boulder, 1111 Engineering Drive, Boulder, CO USA. This work was supported by the Office of Naval Research under Grant N00014-22-1-2482. Alessandro Roncone is with Lab0, Inc. firstname.lastname@colorado.edu

trade-off between efficiency and accuracy. Toward this goal, in this work we introduce the concepts of lazy forward propagation and lazy collision checking in the context of kinodynamic motion planning for BoE, which we refer to as *LazyBoE*. By applying rollouts from the edge bundle to the planning tree directly without resimulating them, these techniques not only significantly reduce planning time but also enable the application of this approach to higher-dimensional systems commonly encountered in real-world applications without compromising the asymptotic optimality guarantees previously established in [15].

In this work, we delineate three key contributions: i) a novel method for the efficient generation of state and control space rollouts from the forward dynamics model of a robot using varying control sequences instead of traditional piecewise-constant controls, aimed at reducing planning times across multiple queries; ii) the introduction of 'lazy' forward propagation and 'lazy' collision checking strategies that employ bounded end-state perturbations and prioritize forward simulations by their likelihood of collision, thereby mitigating the computational load associated with physics-based simulations; iii) the *LazyBoE* kinodynamic motion planner, a robust framework that capitalizes on the aforementioned strategies to rapidly and reliably compute dynamically valid trajectories for complex tasks. We validate our approach on a 7DoF robot arm, and demonstrate how it enables a broad class of real-world tasks.

II. Background and Related Work

Kinodynamic planning involves navigating the complex interplay between the robot's *state space*—comprising position, velocity, and higher-order derivatives—and its *control space*, defined by the set of feasible actions for a given task. The existing literature mainly bifurcates this analysis into two approaches: *sequential* and *interleaved*. Sequential methods, such as TrajOpt [16], CHOMP [17], STOMP [18], KOMO [19], and ITOMP [20], construct a geometric path in the robot's state space first and then assign valid controls to each waypoint. While these methods are advantageous for their minimal design complexity and capability to handle non-linear constraints, they have a number of limitations. They are highly susceptible to the quality of the initial seed trajectories, often confining trajectories to a narrow set homotopic classes [16], [21]. Moreover, they generally lack theoretical guarantees regarding the completeness or optimality of solutions [10]. Within this line of work, there exist hybrid methods that attempt to reconcile these limitations by incorporating sampling-based planning, which in turn brings about theoretical properties such as probabilistic completeness and asymptotic optimality [10] and permits the discovery of multiple solution classes [22]. Time parameterization techniques can also be applied in place of optimization to assign timesteps to each waypoint in the geometric path [9], [23], [24].

In contrast to sequential approaches, interleaved methods cast motion planning as a search problem, alternating between state space sampling and control propagation to find a valid trajectory. These methods often employ steering functions, or inverse models, to delineate a set of controls connecting two arbitrary points in the state space. For instance, DIMT-RRT utilizes a quadratic steering function to account for joint acceleration limits and non-zero initial and goal velocities [25]. AVP computes the range of reachable velocities at the end of a path, given reachable velocities at the start of it [26], [27]. LQR-RRT* uses a quadratic cost function and linearized dynamics in conjunction with a linear-quadratic regulator (LQR) to connect consecutive states in an RRT* tree [28], [29]. Notably, the optimization techniques discussed in sequential methods can also be used as steering functions in interleaved approaches. In all, these techniques enable precise goal state attainment, allow to perform backward searches, and facilitate exact tree connections in bidirectional searches, thus expediting the planning process [30]. However, they come with a set of challenges. Steering functions may not exist or could be difficult to design for a given dynamical system. Furthermore, exact state connections may not be reliable in real-world settings due to uncertainties and the use of optimization as steering may cause planning to get stuck in local minima [11].

The second variety of interleaved methods involves navigating the state and control spaces using forward control propagation. Using a random state and action, these methods determine the resultant state, constructing a tree of dynamically feasible edges [5]. SyCLoP and KPIECE leverage state space discretizations and guide exploration in low-coverage areas [31], [32]. SST optimizes path quality via pruning and heuristic biasing [12]. GBRRT and GABRRT combine multiple propagation methods and backward tree costs as heuristic values for bidirectional kinodynamic search [33]. Other techniques involve pre-mapping the space of valid motions and conducting a search over this state and control space discretization [15], [34]. Existing approaches that exploit lazy strategies for collision checking involve learning-based methods [35] and evaluating paths on an as-needed basis [36]–[38]. We take an empirical approach by computing a collision probability for each kinodynamic action in our state and control space discretization.

Overall, interleaved approaches are easy to design since they leverage general-purpose physics simulations, explore both state and control spaces uniformly, and easily adapt to environmental interactions and multi-object systems. However, they may face slow convergence owing to costly simulation and collision checking methods. Next, we outline how our contribution mitigates these issues, paving the way for fast computation of asymptotically optimal solutions.

III. Methods

In this section, we present *LazyBoE*, our multi-query kinodynamic motion planner. The planner relies on two key contributions: 1) a method for generating a bundle of edges in order to provide a discrete approximation to the robot's state and control spaces (Section III-B), and 2) a quantification of lazy propagation and collision probabilities to reduce planning time (Section III-C). These pieces enable us to

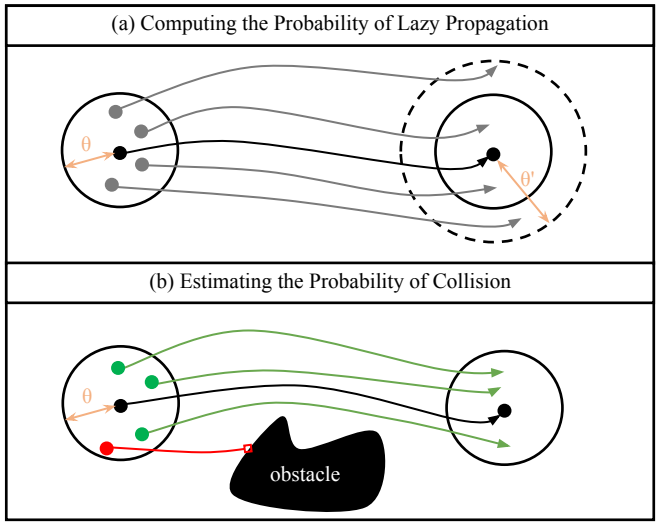

Fig. 2: Perturbing the start state of an edge $e_i \in \mathcal{E}$ (black arrow) by a maximum of θ and applying $e_i.u$ for duration $e_i.\Delta t$ results in a perturbed end state ($\hat{q}_f$). Applying multiple random perturbations for e_i (grey arrows in (a), green and red arrows in (b)) allows us to estimate $e_i.P_{\text{lazy\_prop}} = Pr(\|\hat{q}_f - e_i.q_f\|_2 < \theta) = 2/4$ and $e_i.P_{\text{collision}} = 1/4$ for the provided example. We repeat this process for all edges in $\mathcal{E}$ to integrate lazy approaches to simulation and collision checking in *LazyBoE*.

design and implement a planner that defers computation in the form of simulation and collision-checking until there is a viable path that minimizes a heuristic value (Section III-D).

A. Problem Setup

We formalize the problem of kinodynamic motion planning using edge bundles similarly to [15]. The robot state space $\mathcal{Q} \subset \mathbb{R}^m$ is defined where $q \in \mathcal{Q}$ represents the joint angles of an m-dimensional robot. The control space $\mathcal{U} \subset \mathbb{R}^m$ contains elements $\dot{q} \in \mathcal{U}$ that represent joint velocities. An *edge bundle* is a set of disconnected edges generated by applying randomly sampled control parameters to random robot states for random time durations, thereby connecting random start states to resultant end states:

$$\mathcal{E} = \{(q_0, \dot{\boldsymbol{q}}, \Delta t, q_f) \mid q_0, q_f \in \mathcal{Q},\ \dot{\boldsymbol{q}} \in \mathcal{U}\}$$

such that $f(q_0, \dot{\boldsymbol{q}}) = q_f$, where $f(\cdot)$ defines the forward dynamics model of the robot and $\dot{\boldsymbol{q}} = (\dot{q}_0, \cdots, \dot{q}_{f-1})$ represents a sequence of joint velocities. The i-th edge is denoted as e_i and defined as the tuple $(q_0^i, \dot{\boldsymbol{q}}^{\boldsymbol{i}}, \Delta t^i, q_f^i)$. The objective of *LazyBoE* is to find a continuous, collision-free trajectory

$$\pi(t) : [0,1] \rightarrow \mathcal{Q}_{\text{free}} \subseteq \mathcal{Q}, \quad \pi(0) = q_0, \quad \pi(1) = q_f$$

through the state and control space discretization enabled by $\mathcal{E}$, while respecting the dynamics constraints $\mathcal{D}$ of the robot.

B. Edge Bundle Construction with the Robot Dynamics Model

The backbone of our planner lies in constructing a discrete approximation of the robot's state and control spaces, as a result capturing the nuances of real-world dynamics including nonlinearities like joint friction. Defined as a bundle of edges $\mathcal{E}$, this approximation is a collection of valid, disconnected, and randomly generated forward rollouts of the robot dynamics model over a variable duration Δt. This method of edge generation with a sampled duration is known as Monte Carlo propagation and ensures the preservation of asymptotic optimality in kinodynamic planning algorithms [39]. In order to enable uniform and rapid exploration of the control space, we propagate a sequence of randomly-sampled joint velocity actions, $\dot{\boldsymbol{q}} = (\dot{q}_0, \cdots, \dot{q}_f) \in \mathcal{U}$, from a randomly sampled state, $q \in \mathcal{Q}$. In addition to providing a broader local reachability range (and as a result expanding the explored volume in the state space) from a given state, varying joint velocities over a single propagation leads to longer-duration edges as it prevents the robot from becoming locally "locked" along fixed controls. Moreover, it is necessary that these Monte Carlo propagations are collision-free and obey the Euler-Lagrange robot dynamics model $\mathcal{D}$ to ensure validity in real-world execution. Our approach to bundle generation can be extended to other robot embodiments as long as a valid dynamics model is provided.

As an example, we focus on one particular embodiment and use an analytically-derived and empirically-tuned dynamics model for the 7 degrees-of-freedom (DoF) Franka Emika Panda arm [40]. This analytical model is defined as $\tau(q, \dot{q}, \ddot{q}) = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + f(\dot{q})$, where q, $\dot{q}$, $\ddot{q}$, and τ are 7×1 vectors representing the robot's joint angles, velocities, accelerations, and torques, respectively. M is a symmetric, positive-definite 7×7 joint inertia matrix, C is a 7×1 matrix that captures the Coriolis and centrifugal effects caused by robot motion, g is a 7×1 vector encapsulating the torque required by each motor to counteract gravitational forces, and f is a 7×1 vector that quantifies the torques necessary to counteract friction at the joints. The model is used to verify whether the propagated joint velocity sequence $\dot{\boldsymbol{q}}$ obeys the robot torque limits.

Furthermore, validating an edge involves three classes of checks: collision checks, robot state limit checks, and continuity checks (Algorithm 1, Lines 20-22). Acceptable ranges for the state and control variables and additionally, dynamics quantities are defined as $q_{min} < q < q_{max}$, $\dot{q}_{min} < \dot{q} < \dot{q}_{max}$, $\ddot{q}_{min} < \ddot{q} < \ddot{q}_{max}$, and $\tau_{min} < \tau < \tau_{max}$. In addition, the dynamics simulations must obey continuity constraints due to the stringent control requirements of this robot. For a given static environment, edges are validated to be free of self-collision and environmental collisions via geometric collision-checking algorithms. This incurs a one-time cost since the edges can be used lazily for a significant portion of our motion planning process. We next explain how we can compute useful quantities from this generated edge bundle to enable lazy kinodynamic planning.

C. Edge Perturbation for Lazy Propagation and Collision-Checking

The task of generating an edge bundle that uniformly samples the state and control spaces of a robot is a com-

Algorithm 1: LazyBoE

Input: Start State q_0, Goal Region $\mathcal{Q}_{goal}$, Edge Bundle $\mathcal{E}$, Neighborhood Radius θ
Output: Path π

1 $\pi \leftarrow \emptyset$
2 $\mathcal{V} \leftarrow \{q_0\}$
3 **while** $\mathcal{V} \neq \emptyset$ **do**
  // equivalent to SELECT(.) in [15]
4 $v \leftarrow$ PICK_NODE($\mathcal{V}$)
  // update π to lower cost path
5 **if** $v \in \mathcal{Q}_{goal}$ ***and*** *!IS_LAZY(EDGE_TO(v))* **then**
6 **if** *cost*(π) > *cost*(*PATH_TO*(v)) **then**
7 $\pi \leftarrow$ PATH_TO(v)
8 **if** *IS_LAZY(EDGE_TO(v))* **then**
9 $\mathcal{E}_{\text{neighbor}} \leftarrow$ RADIAL_NN($v, \theta/2, \mathcal{E}$)
10 **else**
11 $\mathcal{E}_{\text{neighbor}} \leftarrow$ RADIAL_NN($v, \theta, \mathcal{E}$)
12 $\mathcal{V}_{\text{next}} \leftarrow \emptyset$
13 **foreach** $e \in \mathcal{E}_{neighbor}$ **do**
14 $P_{\text{select}} \leftarrow (1 - e.P_{\text{collision}}) * e.P_{\text{lazy\_prop}}$
15 apply_lazy_prop $\leftarrow rand() < P_{\text{select}}$
16 **if** *apply_lazy_prop* **then**
17 $\mathcal{V}_{\text{next}} \leftarrow \mathcal{V}_{\text{next}} \cup e.q_f$
18 **else if** *!apply_lazy_prop* ***or*** *!IS_LAZY(EDGE_TO(v))* ***or*** *(IS_LAZY(EDGE_TO(v))* ***and*** *SHOULD_TERMINATE_LAZY())* **then**
  // simulate
19 $e_{new} \leftarrow f(v, e.u, e.\Delta t)$
20 **if** *COLLISION_FREE*(e_{new})
21 ***and*** *WITHIN_LIMITS*(e_{new})
22 ***and*** *IS_CONTINUOUS*(e_{new}) **then**
23 $\mathcal{V}_{\text{next}} \leftarrow \mathcal{V}_{\text{next}} \cup e_{new}.q_f$
24 $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{V}_{\text{next}}$
25 **return** π

putationally intensive process, particularly in the case of a high-DoF robot and increased environment complexity. A substantial part of this computational workload is incurred again during the planning stage, where edges emerging from θ-regions around planning tree nodes undergo simulation and collision checks [15]. This θ is dependent on state-space dimensions and denotes the maximum level of similarity in the state space, thereby bounding the set of valid controls. Additionally, the greedy, heuristic-biased search strategy used by the BoE planner has the potential to get stuck in local minima, thereby necessitating a method for quickly evaluating these greedy paths without simulating them entirely. This motivates the need for lazy approaches to simulation and collision-checking in the kinodynamic setting. More specifically, we need to address the following two questions for each edge $e_i \in \mathcal{E}$:

1) Can e_i be lazily propagated instead of undergoing a full simulation?
2) What is the probability that a lazily propagated edge will result in a collision?

We approach the answers to these questions empirically by conducting a perturbation analysis on each edge in the bundle. A disturbance Δq in the range $(0, \theta]$ is added to the start state for each edge. The resultant end state for an e_i is computed as $\hat{q}_f^i = f(q_0^i + \Delta q, \dot{\boldsymbol{q}}^i)$. In the limit that the number of perturbations for each edge (p) approaches infinity, we are able to accurately estimate both the maximum error in the end state, $\Delta q_f^i = \max(\|e.\boldsymbol{q}_f - \hat{\boldsymbol{q}}_f)\|_2)$, and the potential collision likelihood. To address the first question, an edge viable for lazy propagation is characterized as one that confines the end-state error within the region radius θ, so that viable edges for the next level of propagations can be picked from this fixed θ-neighborhood. However, this criterion is subjected to a more stringent requirement, as depicted in Fig. 3, i.e., the end-state error for e_i must be confined to $\theta/2$ if e_i undergoes lazy propagation to ensure the worst-case error between its real simulation and the following connection to the lazy edge is θ. Put differently, the probability that an edge can be lazily propagated $P^i_{\text{lazy\_prop}} = Pr(\Delta q_f^i < \theta/2)$ is the ratio of the number of valid (collision-free and dynamically-feasible) perturbations for which the maximum end state error is $\theta/2$ to the total number of valid perturbations, p_{valid} (Fig. 2a). Similarly, in response to the second question, we can quantify the probability that an edge will collide when lazily propagated, i.e., $P^i_{\text{collision}} = Pr(e_i \text{ collides})$, as the ratio of the number of edges that collide to the total number of perturbations p (Fig. 2b). This is in contrast to prior work which only considers lazy collision checking in the geometric planning domain and takes a deterministic approach, iteratively discarding edges that are not collision-free [36], [38]. The neighborhood-based control propagation in our work allows for a more nuanced and stochastic approach to lazy collision checking by allowing edges to be re-explored, potentially uncovering paths that would be prematurely eliminated by deterministic methods. In this work, we primarily deal with self-collisions and environmental collisions between the robot and its surrounding environment. These two probability values for each edge extend the definition of an edge bundle $\mathcal{E}$ to the following:

$$\mathcal{E} = \{(q_0, \dot{\boldsymbol{q}}, \Delta t, q_f, P_{\text{lazy\_prop}}, P_{\text{collision}}) \mid q_0, q_f \in \mathcal{Q},\ \dot{\boldsymbol{q}} \in \mathcal{U}\}$$

These two metrics, associated with each edge, are used to guide the planner by deciding when to apply lazy methods for faster planning times.

D. Kinodynamic Motion Planner

In order to reduce the computational costs associated with dynamics simulation and collision checking, we present a lazy approach for exploring the space of edge bundles $\mathcal{E}$, deferring computation until a viable, low cost approximation to a true candidate path has been found. Our method, termed *LazyBoE*, builds upon the design decisions laid out in the BoE planner [15]. In addition, it leverages the computed probabilities $P_{\text{lazy\_prop}}$ and $P_{\text{collision}}$ and combines them into the probability of selecting an edge for lazy propagation as $P_{\text{select}} = (1 - P_{\text{collision}}) * P_{\text{lazy\_prop}}$ (Algorithm 1, Line 14). This encourages edges with a lower collision probability to be propagated first, while also promoting lazy propagation. Since the edges are evaluated probabilistically, they can be used more than once, adding diversity and breadth to exploration, countering the downsides of the greedy approach

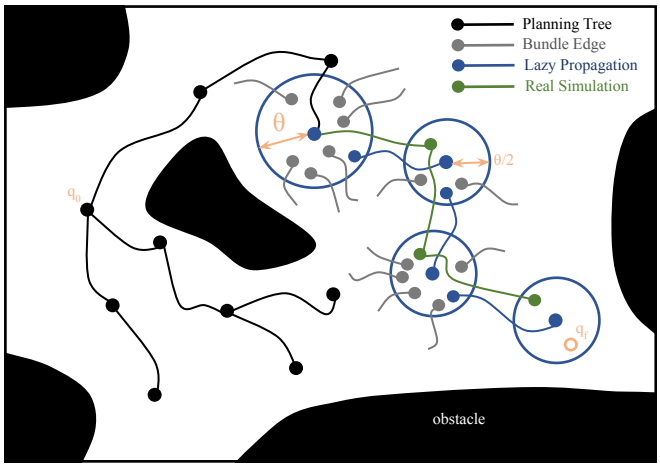

Fig. 3: *LazyBoE* can lazily propagate a series of edges from $\mathcal{E}$ (shown in *grey*) in a way that minimizes heuristic cost. After a lazy candidate path is found (*blue*), a full simulation and collision check is performed along this path (*green*) to extend the planning tree (*black*). To ensure the highest rate of success when performing simulation on lazy paths, neighborhood lookup is restricted to $\theta/2$ for lazy search to allow, in the worst case, the maximum error between the end of a simulated edge and the start of the next lazy edge to be θ.

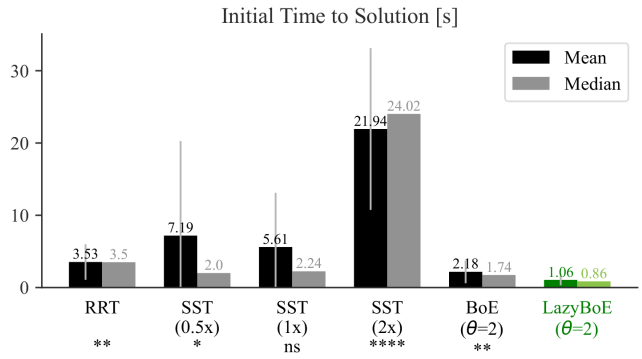

Fig. 4: Our method is able to find the first solution in less time than baseline approaches, owing to its lazy approach to simulation and collision checking. Asterisks indicate significance level when comparing our planner to the baseline methods. *ns* indicates no significant difference.

as taken by the BoE planner. Stochasticity thereby avoids local minima and oscillation issues in greedy search.

In the ideal case scenario, all edges can be lazily propagated, enabling us to search through the edge bundle until the goal is reached and doing a real propagation on the shortest path from start to goal. However, this does not hold due to real simulations that may lead to end states beyond the neighborhood radius $\theta < \theta'$ (Fig. 2a), thereby resulting in $P_{\text{lazy\_prop}} < 1$ for many edges. Therefore, it becomes important to identify the termination conditions for defining when to stop lazy propagation. These termination conditions (as defined by SHOULD_TERMINATE_LAZY() in Algorithm 1, Line 18) are two-fold: i) heuristic value, in our case, the distance to goal state, starts increasing, or; ii) lazy propagation path reaches the goal region $\mathcal{Q}_{goal}$. As a result, simulation along the lazy path is triggered when these conditions hold, if the preceding edge is not a lazy one, or with probability $1 - P_{\text{select}}$ (Algorithm 1, Line 18). Next, we demonstrate how these lazy approaches lead to lower planning times and higher success rates in comparison to baseline kinodynamic motion planning algorithms.

IV. Evaluation

A. Test Setup

We benchmark *LazyBoE* against a number of kinodynamic planners with forward propagation, namely RRT [8], SST [12], and BoE [15].[1] We test three variants of SST based on different selection-to-pruning radius (0.5x, 1x, and 2x). Each algorithm is evaluated over 25 planning problems, i.e., pairs of collision-free, randomly-sampled start and goal joint angles, on four metrics: i) *time to initial solution* in seconds (Fig. 4), ii) *final solution cost* as measured by the arclength of the trajectory in joint angle space (Fig. 5), iii) *success rate* denoted by the fraction of trials that resulted in at least one planned trajectory (Fig. 7), and iv) *number of solutions* per planning problem (Fig. 6). This delineation into time to initial solution and final solution cost is necessitated by the fact that SST, BoE, and *LazyBoE* are asymptotically optimal in nature. All algorithms were implemented in Python 3.8 and evaluated on a 6-core AMD Ryzen 5 laptop with 32GB RAM running Ubuntu 20.04. Each algorithm was given a time budget of 60s, except RRT, which terminates on the first solution.

[1]Links to our code, videos, and demonstrations of the experiments are available here: https://hiro-group.ronc.one/research/lazyboe-icra24.html.

B. Quantitative Analysis

Our *LazyBoE* planner outperformed baseline kinodynamic methods in multiple key metrics. First, it achieved an average solution time of 1.06s, substantially faster than the other methods that were in the range $[2.18\text{s}, 21.94\text{s}]$ (Fig. 4). This is due to deferred propagation and collision checking, which enable a more focused search. Of note, despite this significant performance improvement, *LazyBoE*'s final solution cost was competitive in quality (3.42 cost) to the best performing alternatives, BoE (3.43 cost) and SST (Fig. 5). In essence, our method provides efficiency without sacrificing optimality.

The planner also found an average of 3.12 solutions, over 1.5x more than the other methods (Fig. 6). This increased solution diversity arises from stochastic edge selection and reuse compared to greedy approaches. Lazily reusing edges rather than discarding them expands the search frontier faster. More paths are available to evaluate and iterate on, thereby directly translating to higher success rates. With more valid candidate paths approximated and available to the search, the likelihood of discovering a collision-free solution path increases substantially. The results validate this, with *LazyBoE* succeeding 92% of the time compared to $[80 - 88]\%$ for other methods (Fig. 7). In summary, *LazyBoE*'s use of lazy propagation and collisions to minimize wasted computations allows efficient searches that find high quality

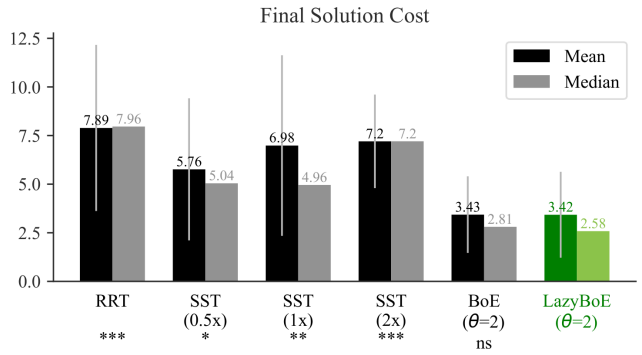

Fig. 5: The final solution cost, i.e., the solution with the lowest cost, for our planner is comparable to the BoE planner. Asterisks indicate significance levels.

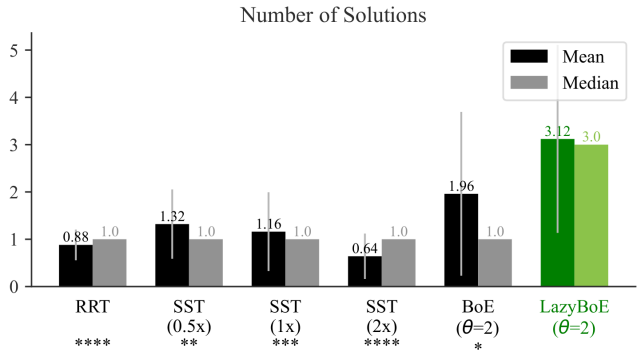

Fig. 6: Our planner is able to explore a larger number of solutions than the baseline methods. RRT terminates after finding the first solution, but the mean number of solutions is less than 1 because RRT does not succeed in finding a solution for every planning problem Fig. 7. Asterisks indicate the significance levels.

solutions quickly, while maintaining a broad exploration. This quantitative analysis validates lazy techniques can improve performance in kinodynamic planning.

C. Significance Testing

Statistical analysis using the Mann-Whitney U test further reinforces the results above. $p-$values for each test are highlighted in Fig. 4, Fig. 5, and Fig. 6 and marked by 'ns' for $p > 0.05$, '*' for $p \leq 0.05$, '**' for $p \leq 0.01$, '***' for $p \leq 0.001$, and '****' for $p \leq 0.0001$.

LazyBoE's speedup improvement is statistically significant with respect to most baseline algorithms, with notably low $p-$values in comparison to RRT, SST (0.5x), and SST (2x). However, the comparison with SST (1x) did not reach statistical significance (p = 0.0529), indicating that the time difference with this baseline is not conclusive.

For a more robust analysis, we look at the median value, which provides a more accurate representation of the data's central tendency, particularly in the presence of a skewed distribution. *LazyBoE* has a lower median cost when compared to various iterations of SST (0.5x) and RRT, demonstrating a potential advantage. It is worth noting that the cost comparison with the BoE algorithm did not yield a significant difference (p = 0.8142), suggesting comparable cost-

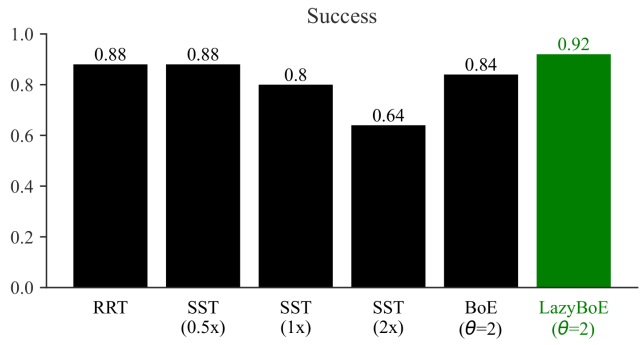

Fig. 7: Our algorithm has a higher success rate compared to the baselines owing to the exploration of a greater number of potential solutions (Fig. 6).

efficiency between these two algorithms. Our algorithm also generated a higher median number of solutions compared to all baselines, with statistically significant results in all comparisons. In summary, this significance testing reinforces our claims that *LazyBoE* offers performance improvements in terms of both speed and cost-effectiveness.

V. Conclusion and Discussion

In this work, we presented *LazyBoE*, a multi-query approach to kinodynamic motion planning that takes advantage of lazy propagation and collision checking to outperform existing baselines in a number of key metrics. Specifically, our method reduces planning time and demonstrates a high success rate, given its ability to explore a wider range of solutions. However, our method faces certain limitations. The varying control sampling for a given Monte Carlo propagation introduces jitter into the trajectory which has to be smoothed with a low-pass filter before real-world execution. Future work can explore biased sampling techniques that consider the history of control states. Due to memory limitations, *LazyBoE* is confined to a limited subset of the robot's state and control spaces. Scaling the approach to cover the entire robot workspace is a pressing challenge, stemming from memory constraints, especially when planning dynamically-feasible trajectories that may involve higher-order derivatives of the state space variables. While strategies such as selective loading or using a database could alleviate this, they require significant engineering optimizations.

Beyond the engineering required to scale this method, we also intend to explore application-centric extensions, such as planning for heavy payloads, liquid transport, and non-prehensile manipulation, which can benefit from assigning task-specific weights to edges for more biased, multi-query exploration. While our implementation focuses on kinodynamic planning for robotic manipulation, the foundational principles of our planner are designed with broader applicability in mind. Generalizing the planner to accommodate a wider array of kinodynamic planning problems, beyond the realm of robotic manipulation, represents a promising direction for future work.

REFERENCES

[1] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.

[2] M. T. Mason, "Toward robotic manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 1–28, 2018.

[3] J. Ichnowski, Y. Avigal, Y. Liu, and K. Goldberg, "Gomp-fit: Grasp-optimized motion planning for fast inertial transport," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 5255–5261.

[4] G. Salhotra, I.-C. A. Liu, M. Dominguez-Kuhne, and G. S. Sukhatme, "Learning deformable object manipulation from expert demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8775–8782, 2022.

[5] A. Pasricha, Y.-S. Tung, B. Hayes, and A. Roncone, "PokeRRT: Poking as a skill and failure recovery tactic for planar nonprehensile manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4480–4487, 2022.

[6] Y.-B. Jia, M. Gardner, and X. Mu, "Batting an in-flight object to the target," *The International Journal of Robotics Research*, vol. 38, no. 4, pp. 451–485, 2019.

[7] E. Schmerling and M. Pavone, "Kinodynamic planning," *Encyclopedia of Robotics. Springer,*, 2019.

[8] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[9] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, 2014.

[10] A. Kuntz, C. Bowen, and R. Alterovitz, "Fast anytime motion planning in point clouds by interleaving sampling and interior point optimization," in *Robotics Research: The 18th International Symposium ISRR*, Springer, 2019, pp. 929–945.

[11] Z. Littlefield, "Efficient and asymptotically optimal kinodynamic motion planning," Ph.D. dissertation, Rutgers The State University of New Jersey, School of Graduate Studies, 2020.

[12] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[13] S. F. Saramago and M. Ceccarelli, "An optimum robot path planning with payload constraints," *Robotica*, vol. 20, no. 4, pp. 395–404, 2002.

[14] 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.

[15] R. Shome and L. E. Kavraki, "Asymptotically optimal kinodynamic planning using bundles of edges," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 9988–9994.

[16] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *Robotics: science and systems*, Berlin, Germany, vol. 9, 2013, pp. 1–10.

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

[18] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 4569–4574.

[19] M. Toussaint, "Newton methods for k-order markov constrained motion problems," *arXiv preprint arXiv:1407.0414*, 2014.

[20] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proceedings of the international conference on automated planning and scheduling*, vol. 22, 2012, pp. 207–215.

[21] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Science Robotics*, vol. 5, no. 48, eabd7710, 2020.

[22] J. Kamat, J. Ortiz-Haro, M. Toussaint, F. T. Pokorny, and A. Orthey, "Bitkomo: Combining sampling and optimization for fast convergence in optimal motion planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 4492–4497.

[23] L. Berscheid and T. Kroeger, "Jerk-limited Real-time Trajectory Generation with Arbitrary Target States," in *Proceedings of Robotics: Science and Systems*, Virtual, Jul. 2021. DOI: 10.15607/RSS.2021.XVII.015.

[24] S. Macfarlane and E. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," English, *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 42–52, Feb. 2003, ISSN: 1545-5955. DOI: 10.1109/TRA.2002.807548.

[25] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 3713–3719.

[26] Q.-C. Pham, S. Caron, and Y. Nakamura, "Kinodynamic planning in the configuration space via admissible velocity propagation.," in *Robotics: Science and Systems*, vol. 32, 2013.

[27] J. Johnson and K. Hauser, "Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path," in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 2035–2041.

[28] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 2537–2542.

[29] T. M. Caldwell and N. Correll, "Fast sample-based planning for dynamic systems by zero-control linearization-based steering," *Robotics Research: Volume 2*, pp. 453–469, 2018.

[30] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[31] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, 2010.

[32] I. A. Sucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 116–131, 2011.

[33] S. Nayak and M. W. Otte, "Bidirectional sampling-based motion planning without two-point boundary value solution," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3636–3654, 2022.

[34] R. Bordalba, L. Ros, and J. M. Porta, "Randomized kinodynamic planning for constrained systems," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 7079–7086.

[35] C. Yu and S. Gao, "Reducing collision checking for sampling-based motion planning using graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4274–4289, 2021.

[36] L. Kavraki and R. Bohlin, "Path planning using lazy prm," in *Proc. IEEE Int. Conference on Robotics and Automation. San Francisco*, 2000.

[37] A. Mandalika, S. Choudhury, O. Salzman, and S. Srinivasa, "Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 745–753.

[38] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 2951–2957.

[39] T. Kunz and M. Stilman, "Kinodynamic rrts with fixed time step and best-input extension are not probabilistically complete," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2015, pp. 233–244.

[40] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. D. Luca, "Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4147–4154, Oct. 2019. DOI: 10.1109/lra.2019.2931248. [Online]. Available: https://doi.org/10.1109/lra.2019.2931248.