Effective Kinodynamic Planning and Exploration Through Quality Diversity and Trajectory Optimization

Konstantinos A. Asimakopoulos[1,2], Aristeidis A. Androutsopoulos[3], Michael N. Vrahatis[2], and Konstantinos I. Chatzilygeroudis[1,2(✉)]

[1] Laboratory of Automation and Robotics (LAR), Department of Electrical and Computer Engineering, University of Patras, 26504 Patras, Greece
costashatz@upatras.gr

[2] Computational Intelligence Laboratory (CILab), Department of Mathematics, University of Patras, 26110 Patras, Greece
vrahatis@math.upatras.gr

[3] Computer Engineering and Informatics Department (CEID), University of Patras, 26504 Patras, Greece

Abstract. Efficient and rapid kinodynamic planning is crucial for numerous real-world robotics applications. Various methods have been proposed to address this challenge, primarily falling into two categories: (a) randomized planners and (b) trajectory optimization utilizing simplified models and numerical optimization. Randomized planners such as RRT and PRM excel in exploring the state space, while trajectory optimization methods, like direct collocation, are adept at discovering optimal trajectories within well-defined spaces. We aim to achieve effective and efficient kinodynamic planning and exploration by integrating evolutionary algorithms (Quality-Diversity) with trajectory optimization. Our preliminary experiments showcase that using the proposed methodology we get the best from both worlds on two simulated experiments.

1 Introduction and Related Work

Effective exploration of the state space is crucial in real-world robotic applications. However in hard exploration problems a large amount of iterations is usually required to reach areas of high-reward. This necessitates the use of an efficient exploration strategy. Randomized planners like Rapidly exploring Random Trees (RRT) [5] and Probabilistic Roadmaps (PRM) [4] navigate large state spaces efficiently, with RRT* even attempting to find the optimal path [3]. Trajectory optimization excels in well-defined spaces but has limitations for extensive state-space coverage. Go-Explore [2] is a Quality-Diversity (QD) optimization algorithm [1,7] and is able to provide numerous diverse high-performing

This work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "3rd Call for H.F.R.I. Research Projects to support Post-Doctoral Researchers" (Project Acronym: NOSALRO, Project Number: 7541).

P. Festa et al. (Eds.): LION 2024, LNCS 14990, pp. 32–36, 2025.
https://doi.org/10.1007/978-3-031-75623-8_3

solutions despite having a simple random exploration strategy. We propose a more structured approach by substituting this strategy with trajectory optimization for a synergistic solution. Our improvements, outlined in Sect. 2, enhance the Go-Explore framework with the robustness of trajectory optimization, while also improving its exploration abilities in difficult scenarios.

2 Proposed Approach

Go-Explore Framework: Go-Explore [2] addresses detachment and derailment challenges by (1) maintaining a global archive of all uniquely interesting states and (2) dividing exploration into deterministic returns to states followed by stochastic exploration. To enhance robustness, the algorithm operates in two phases: (a) task-solving exploration and (b) solution robustification, involving processes like imitation learning on the discovered trajectories[1]. The initial phase of Go-Explore involves building an archive of interesting states, each associated with a *history* of *actions* and a *score* indicating exploration interest. This state/history/score group is called a *cell*. The archive, starting with the initial state, is denoted as $\mathcal{A}$. At each iteration, the algorithm follows the steps in Algorithm 1. During cell insertion, scores are updated and a binary similarity metric evaluates the state in comparison to existing states. If distinct or higher performing, it is added and it replaces the similar cells in the archive.

Algorithm 1. Go-Explore Framework

procedure GO-EXPLORE($\boldsymbol{s}_0$, n_{batch}, n_{iter})
  $\mathcal{A} \leftarrow$ new_cell($\boldsymbol{s}_0, \emptyset$) ▷ Initialize archive, $\mathcal{A}$, with initial state $\boldsymbol{s}_0$
  for $n = 1$ **to** n_{iter} **do**
    cells $\leftarrow$ SELECTCELLS($\mathcal{A}$, n_{batch}) ▷ Select n_{batch} exploration cells
    new_cells $\leftarrow$ EXPLORE(**cells**) ▷ Run stochastic exploration
    ADDTOARCHIVE($\mathcal{A}$, **new_cells**) ▷ Add explored cells to the archive
    UPDATESCORES($\mathcal{A}$) ▷ Update the scores of cells
  end for
  return $\mathcal{A}$ ▷ The outcome of the algorithm is the archive
end procedure

Go-Explore with Trajectory Optimization: In the original implementation of Go-Explore [2] (Vanilla Go-Explore) the exploration strategy used is essentially a sequence of random actions. We propose to leverage trajectory optimization instead of random exploration. At the start of every exploration phase, *batch_size* cells are chosen from the archive to explore from. For every chosen *cell* we sample a random point within a small distance. We then use trajectory optimization to find the path between the cells and their respective

[1] We only care about the first phase in this work.

random points. In this manner, we leverage the robust solutions provided by optimization on a smaller scale, all the while retaining the exploration benefits inherent in Go-Explore.

Bidirectional Go-Explore: To achieve a faster and more effective exploration, we propose a bidirectional Go-Explore algorithm with two agents—forward and backward—initialized at different state space points. Each agent runs a slightly different version of our trajectory optimization enhanced Go-Explore, featuring separate archives. The forward version reflects our proposed model, while the backward version addresses the inverse optimization problem. During exploration for the backward version, the randomly sampled point is considered the starting point, aiming to return to the selected cell. The bidirectional exploration involves choosing a batch of cells, returning to their states, sampling a nearby random state, using trajectory optimization for the forward and backward version and updating their respective archives with the trajectory optimization results. We use this "reverse" design for the backwards version to facilitate exchange of useful discovered information between the two agents.

Trajectory Merging: In order to be able to merge information from the forward and backward archives, we use a merging mechanism. After each exploration phase, we check if cells from the two archives are close based on a *merging distance*. If so, trajectory optimization concatenates their paths into one merged trajectory that is then added to the forward archive. The step-by-step merging process involves checking proximity between backward and forward archive cells, using trajectory optimization for optimal paths, concatenating histories, and adding the merged cell to the archive.

3 Experimental Results

In order to showcase the effectiveness of our proposed method, we designed and performed a number of simulated experiments. We tested our proposed method in two distinct environments and compared its performance to that of the Vanilla Go-Explore and RRT algorithms in space coverage and average trajectory length; we compare to RRT only in the Maze scenario.

Maze Environment: For this experiment we use the 2D car-like kinematic vehicle model from [6] which does not allow skidding. We use this model to explore a 2D maze environment (see Fig. 1).

Planar Quadrotor: We use the Planar Quadrotor environment (Fig. 2) because a) it is higher dimensional, and b) the control commands matter (a.k.a., the quadrotor will crash if given bad commands).

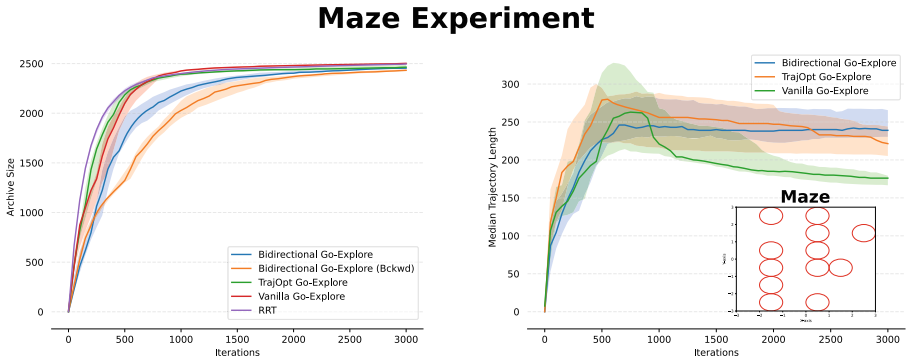

Fig. 1. 2D Maze Experiments. Median and 5th/95th percentiles over 5 replicates.

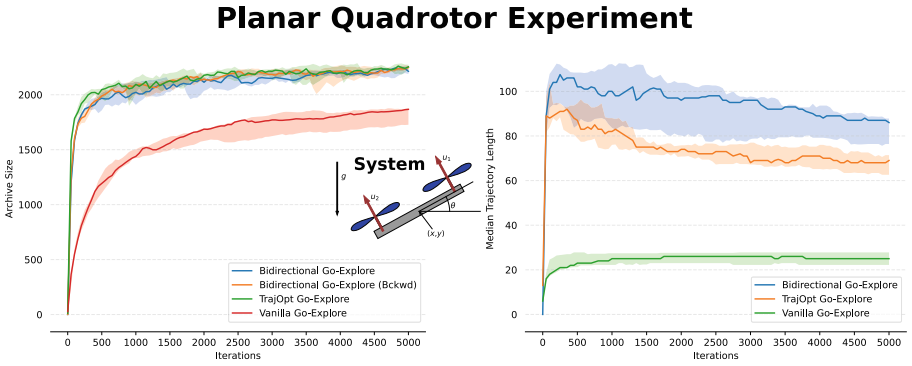

Fig. 2. Planar Quadrotor. Median and 5th/95th percentiles over 5 replicates.

Results: We run 5 replicates of each scenario for each algorithm and report the space coverage and average trajectory length. Trajectory length is important (especially in the quadrotor) as it showcases that the algorithm has found effective control policies. The results showcase that our proposed method on simpler environments like the 2D car maze is on par with state of the art methods (Fig. 1) in both state space coverage and control efficacy, while clearly outperforming other methods in the more complicated and delicate quadrotor environment (Fig. 2).

4 Concluding Remarks

We have introduced a variant of the Go-Explore where we utilize trajectory optimization in a strategic manner in order to boost its performance while keeping its exploration capabilities. We also provided preliminary results on a bidirectional Go-Explore scheme. Overall, our methods are able to efficiently explore the state space, while also discovering effective controllers.

References

1. Chatzilygeroudis, K., Cully, A., Vassiliades, V., Mouret, J.B.: Quality-diversity optimization: a novel branch of stochastic optimization. In: Black Box Optimization, Machine Learning, and No-Free Lunch Theorems, pp. 109–135. Springer (2021)
2. Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O., Clune, J.: First return, then explore. Nature **590**(7847), 580–586 (2021)
3. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. IJRR **30**(7), 846–894 (2011)
4. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
5. LaValle, S.M., Kuffner Jr., J.J.: Randomized kinodynamic planning. In: IEEE International Conference on Robotics and Automation, vol. 1, pp. 473–479 (1999)
6. Pepy, R., Lambert, A., Mounier, H.: Path planning using a dynamic vehicle model. In: 2006 2nd International Conference on Information & Communication Technologies, vol. 1, pp. 781–786. IEEE (2006)
7. Pugh, J.K., Soros, L.B., Stanley, K.O.: Quality diversity: a new frontier for evolutionary computation. Front. Robot. AI **3**, 40 (2016)