Escaping Local Minima in Search-Based Planning using Soft Duplicate Detection

Wei Du, Sung-Kyun Kim, Oren Salzman, and Maxim Likhachev
{wdu2,osalzman}@andrew.cmu.edu, {kimsk,maxim}@cs.cmu.edu

***Abstract*— Search-based planning for relatively low-dimensional motion-planning problems such as for autonomous navigation and autonomous flight has been shown to be very successful. Such framework relies on laying a grid over a state-space and constructing a set of actions (motion primitives) that connect the centers of cells. However, in some cases such as kinodynamic motion planning, planning for bipedal robots with high balance requirements, computing these actions can be highly non-trivial and often impossible depending on the dynamic constraints. In this paper, we explore a *soft* version of discretization, wherein the state-space remains to be continuous but the search tries to avoid exploring states that are likely to be duplicates of states that have already been explored. We refer to this property of the search as *soft duplicate detection* and view it as a relaxation of the standard notion of duplicate detection. Empirically, we show that the search can efficiently compute paths in highly-constrained settings and outperforms alternatives on several domains.**

I. Introduction

Robots such as self-driving cars, aerial vehicles or bipedal robots often come with kinodynamic constraints [1], [2] or dynamic constraints [3]. One approach to planning with such constraints is using search-based planners. Here, the continuous state space is discretized into cells and the search algorithm traverses the centers of these cells. This is done by defining motion primitives (see, e.g., [4], [5], [6]) which are precomputed actions that the robot can take at every state. The approach relies on the motion primitives to connect the centers of different cells which correspond to solving the two-point boundary value problem (BVP) [7]. When planning with kinodynamic constraints this may be infeasible and hence actions may not terminate at the center of a cell (see Fig. 1). One possible solution might be discretizing the search space using a higher resolution so that actions could start and stop at the center of grids. The size of the search space will then grow quickly, making the approach computationally overly expensive.

In such cases, we could opt to plan in the continuous state space. However, search-based planning in continuous space is intractable: Heuristics that do not explicitly account for the kinodynamic constraints may guide the planner to regions, known as "local minima" [8], causing the planner to keep on expanding similar states. Consequently, the planner progresses slowly towards the goal or cannot progress at all.

Indeed, in order to avoid expanding similar, yet non-identical states, existing approaches group states into equivalence classes [9], [10]. This is analogous to how standard search algorithms (over finite discrete graphs) reduce search effort by identifying previously-visited states, a process known as *duplicate detection* [11], [12].

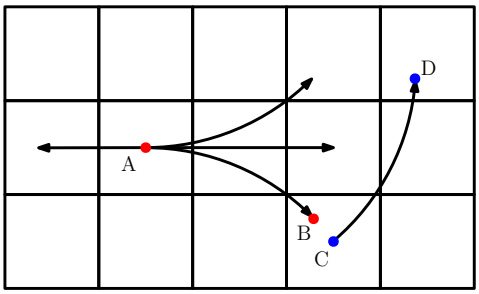


Fig. 1: Discontinuity in discretized space planning with non-holonomic constraints on the motion primitives. Motion primitive A-B does not terminate at the center of a grid whereas C-D start from the center of the same grid.

To this end, we suggest to extend the notion of duplicate detection in discrete spaces to *soft duplicate detection* in continuous spaces. The idea is to estimate how much new states can contribute to the search using what we refer to as *duplicity*. Using the weighted A* framework [13] we then penalize states with high duplicity by inflating their heuristic value and not, as in standard duplicate detection, prune them away completely. This allows the search to be complete while focusing its computational resources. We evaluate our approach for robots with kinodynamic constraints operating in 3D, 5D and 8D state spaces and demonstrate its favorable success rate when compared to existing algorithms.

Before continuing to describe our algorithmic framework (Sec. II) and evaluating our method (Sec. III), we briefly mention alternative approaches for kinodynamic planning and some additional related work. Sampling-based planners [7] have been used in this context by either constructing planners that only extend a state but do not require to exactly connect it to a given state (see, e.g., [14], [15]) or by using motion primitives [16], [17]. We also note that our problem of soft duplicate detection can be seen as a variant to the problem of stagnation detection where a search-based planner is required to identify when the heuristic ceases to be informative (see, e.g., [18], [19]). Alternatively, methods such as potential fields [20], [21] are exploited in pushing planners away from local minima regions. Previous planning experiences are made use of in assisting planners to discern and speed up planning process or to avoid local minima places [22], [23].

II. Algorithmic framework

We are given a robot with state space S whose motion is restricted by a set of dynamic constraints. In addition, we are given a start state s_{start} and a goal region $S_{\text{goal}} \subset S$ and wish to compute a dynamically-feasible, collision-free path connecting s_{start} to a state in S_{goal}. We assume that there exists a set of *motion primitives* which are dynamically-feasible actions available at each state. These motion primitives induce a successor function Succ : $S \to 2^S$. Furthermore, we assume the existence of an admissible[1] heuristic function $h: S \times S \to \mathbb{R}$.

A. WA* with soft duplicate detection—general framework

Given some approximation factor $\varepsilon_0 > 1$, our algorithmic framework runs a weighted A* (WA*) search [13]. Recall that WA* expands states in order of their f-value,

$$f(s) = g(s) + \varepsilon_0 h(s). \tag{1}$$

Here $g(s)$ is the (computed) cost to reach state s from s_{start}. We suggest to replace ε_0 with a state-dependent inflation $\varepsilon(s)$ which we compute using the notion of *duplicity* and *soft duplicate detection*.

Given the set of states U that WA* considered at a given point, the duplicity of a new state s measures the (inverse) likelihood that s will contribute to computing a path to the goal region. Namely, a state with high duplicity is likely to be redundant given all the states already explored by the search, whereas a state with low duplicity may be essential in computing a path to the goal. Given the notion of duplicity we perform soft duplicate detection which corresponds to inflating the heuristic value of states proportional to their duplicity—states with higher duplicity are highly penalized, giving them low priority in the priority queue maintained by WA*, and vice versa.

These notions are general and we foresee them being applied to a variety of domains where the way duplicity is measured depends on the domain. However, we assume that given some measure of duplicity between two given states $\text{dup}(\text{s}, \text{s}')$, where $\text{dup}(\text{s}, \text{s}') \leq 1$, the duplicity of a state is defined as

$$\text{dup}(\text{s}) := \max_{\text{u} \in \text{U}} \text{dup}(\text{s}, \text{u}). \tag{2}$$

Furthermore, given an approximation factor $\varepsilon_{\max} > \varepsilon_0$, our framework replaces ε_0 in Eq. 1 with

$$\varepsilon(s) = \max\left(\varepsilon_{\max} \cdot \text{dup}(\text{s}),\ \varepsilon_0\right). \tag{3}$$

For pseudo-code explaining how these notions are used by WA*, see Alg. 1. Here OPEN is a priority queue maintaining states that WA* may expand while CLOSED is a set storing states that have already been expanded. Furthermore, the cost from state s to s' is denoted as $c(s, s')$. The colored Lines 10 and 11 correspond to computing the duplicity of a new state and using it for soft duplicate detection, respectively. We note that if the underlying heuristic is admissible and if indeed $\text{dup}(\text{s}, \text{s}') \leq 1$, then when using a dynamic heuristic with maximal inflation $\varepsilon_{\max}$, WA* is guaranteed to produce results with a bounded sub-optimality value of $\varepsilon_{\max}$ [13], [24].

[1] A heuristic function is said to be admissible if it never overestimates the cost of reaching a state in S_{goal}.

Algorithm 1 WA* with Soft Duplicate Detection

Input: duplicity $\text{dup}(\cdot, \cdot)$, approximation factors $\varepsilon_0, \varepsilon_{\max}$

1: $g(s_{\text{start}}) = 0$; ▷ g-value for other states initialized to ∞
2: CLOSED = $\emptyset$;
3: OPEN $\leftarrow s_{\text{start}}$; ▷ key is f-value (Eq. 1)
4: **while** s_{goal} is not expanded and OPEN $\neq \emptyset$ **do**
5: U = OPEN $\cup$ CLOSED;
6: remove s with the smallest f-value from OPEN;
7: insert s into CLOSED;
8: **for all** $s' \in$ Succ(s) **do**
9: **if** $s' \notin U$ **then**
10: $\text{dup}(\text{s}') = \max_{\text{u} \in \text{U}}\{\text{dup}(\text{s}', \text{u})\}$;
11: $\varepsilon(s') = \max\{\varepsilon_{\max} \cdot (\text{dup}(\text{s}')), \varepsilon_0\}$;
12: **if** $g(s') > g(s) + c(s, s')$ **then**
13: $g(s') = g(s) + c(s, s')$;
14: $f(s') = g(s') + \varepsilon(s') \cdot h(s')$;
15: insert s' into OPEN; ▷ key is f-value (Line. 14)
16: **if** s_{goal} is expanded **then**
17: **return** solution();
18: **return** null;

B. Soft duplicate detection scheme

In planning, we want to account both for the similarity between states as well as for the environment. Specifically, we want the duplicity of two states to be proportional to their *similarity* (more similar states should have higher duplicity) and inversely proportional to their proximity to obstacles (states lying in the vicinity of obstacles should have lower duplicity). Thus, for states s and s' we define,

$$\text{dup}(\text{s}, \text{s}') = 1 - \text{dist}(\text{s}, \text{s}')/(\text{R} \cdot \gamma(\text{s}_{\text{parent}})), \tag{4}$$

where dist : $S \times S \to \mathbb{R}$ is some distance metric, R is a constant used to normalize the distance value. The $\gamma(s)$ value is the *valid successor rate* which captures the proximity to obstacles, as is illustrated in Fig. 2. Specifically, $\gamma(s)$ is defined as

$$\gamma(s) = \frac{\text{number of valid successors of } s}{\text{number of all successors of } s}. \tag{5}$$

For efficient computation, we do not need to iterate over all states in OPEN and CLOSE (Alg. 1, Line 10) but we can use nearest-neighbor computation. Given a state s and the set of states U = OPEN $\cup$ CLOSED, let $d_{\text{NN}}(s, U) = \min_{u \in U} \text{dist}(s, u)$ denote the minimal distance to any state in U. Specifically, we store all states in U in a kd-tree [25] and compute the duplicity of a state as,

$$\text{dup}(\text{s}) = 1 - \text{d}_{\text{NN}}(\text{s}, \text{U})/(\text{R} \cdot \gamma(\text{s}_{\text{parent}})). \tag{6}$$

C. Comparison with equivalence-class algorithms

In this section, we compare our approach with the most closely-related planning algorithms—search-based algorithms that use equivalence classes. Specifically, we compare our algorithm (denoted as Penalty-WA*) with Barraquand's algorithm [9] and ε-optimal A* with equivalence

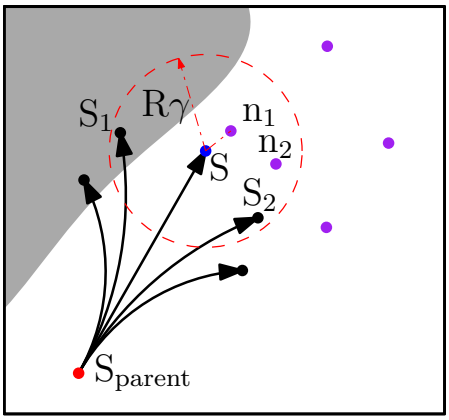


Fig. 2: Illustration of the how we compute the duplicity of a state s (blue). Gray area is the obstacle; The *valid successor rate* ($\gamma(s_{\text{parent}})$) is computed for the parent (red) of s. After filtering out siblings s_1 and s_2 (generally, relatives within checking radius and generated by each necessary step to produce s should be filtered out, otherwise s will be penalized as long as it is produced), nearest-neighbor n_1 is picked for the value of d_{NN}.

classes [10], referred to as Barraquand and Gonzalez, respectively.

The Barraquand algorithm conducts planning with kinodynamic constraints in a discretized space based on the A* algorithm. In each cell, states are compared according to their cost from the start state. States with higher cost are said to be dominated and are pruned away. Consequently, within each cell, only one state with the lowest cost from the start state is stored. This method exploits a fixed-cell decomposition that defines equivalence classes. Unfortunately, this greedy approach causes the algorithm to be incomplete as pruned states may be necessary to find a solution to the goal.

In the Gonzalez algorithm, a list referred to as NOTDOM is created to store non-dominated states. Each state in NOTDOM defines an equivalence class, which has the lowest cost from the start state within that equivalence class. Once a state s is expanded, its nearest neighbor is checked against NOTDOM. If s has better cost from start state or the nearest state does not exist within some radius, s will be added to NOTDOM and the nearest neighbor will be removed, if it exists. If s is dominated, instead of being pruned, it will be penalized by having its heuristic value inflated. This algorithm is more flexible and provides completeness and sub-optimal bounds for the solution.

The Penalty-WA* algorithm, however, does not implement equivalence class. The heuristic of states in Penalty-WA* are inflated proportionally to the duplicity value. These penalties take continuous values in contrast to Gonzalez's approach of either penalizing or not. Moreover, the penalty value added to states in Gonzalez's algorithm is constant for all states, which does not take the domain into account at all. In contrast, our algorithm takes the environment into consideration, i.e., we consider the order of states generated in one local area and the existence of nearby obstacles, making this algorithm adaptive. That is, the algorithm dynamically toggles between a sparse search in obstacle-free regions and a dense search near obstacles.

III. Experiments and Results

In this section we present experiments evaluating the performance of our algorithm and comparing it to existing work on 3D, 5D and 8D spaces. Specifically, the 3D space models a car-like robot with constraints on the turning radius. Here, a configuration (x, y, θ) corresponds to the (planar) location and orientation of the robot, respectively. The 5D space models a UAV (unmanned aerial vehicle) with constraints on the linear acceleration and angular speed. Here, a configuration $(x, y, z, \theta, v_{x,y})$ corresponds to the (spatial) location, orientation and velocity of the robot, respectively. The 8D space models a bipedal robot. Here, a configuration of each of the robot's feet (x, y, z, θ) corresponds to the (spatial) location and orientation of the foot. The distance metrics for these three domains are:

$$\text{dist}(\text{s}, \text{s}') = \text{E}(\text{s}, \text{s}') + \lambda \cdot \text{A}(\text{s}, \text{s}'), \tag{7}$$

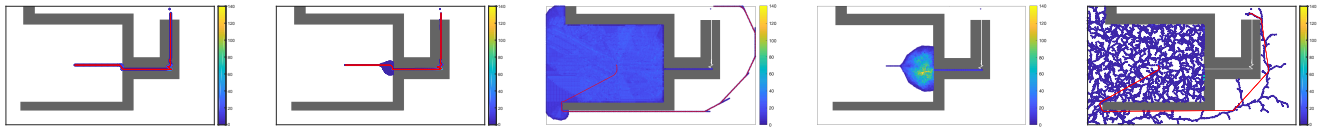
where $E(s, s')$ is the Euclidean distance between state s and s', $A(s, s')$ is the angular distance between state s and s', in radian. The multiplier λ is the weight of the angular distance compared to the Euclidean distance. As a base heuristic, we ran a BFS search from the goal taking into account only the location (planar and spatial locations in 3D, 5D and 8D planning domains, respectively) on a discretized state space.We used this heuristic within the WA* framework as well as with our penalty-based approach and denote the corresponding algorithms BFS-WA*. In addition, we implemented the Barraquand algorithm and the Gonzalez algorithm. Finally, we implemented the RRT algorithm. In all our results, we gave each planner a timeout of 120 seconds and report the solution length as our cost metric. For all algorithms that require an approximation factor (ε_0) we used a value of three. All experiments were run on an Intel i7-3770 CPU (3.40 GHz) with 16GB RAM.

A. 3D Space Planning Results

1) Domain: For this domain, we use 66 benchmarks from *Moving AI Lab* [26] *Starcraft set* as well as a toy scenario used to qualitatively compare the different algorithms (Fig. 3 and 4). The map sizes taken from the *Moving AI Lab* range from 500×500 to 1024×1024 grids with resolution 0.025 m. Start and goal pairs are randomly generated in the free space for each map.

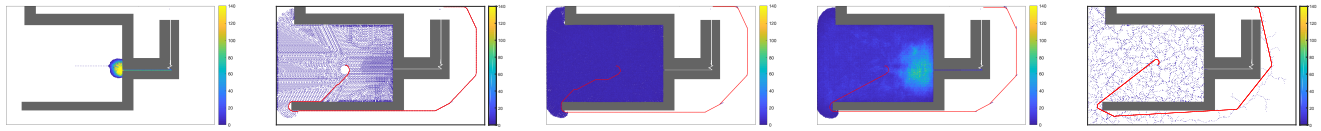
2) Motion primitives: We generated motion primitives using the SBPL[2] library that has a unicycle model with constraints on the turning radius. The motion primitives implemented here share the same prototype as shown in Fig. 1: four successors are generated from the predecessor A via the action set $U_{\text{action}} = \{\Delta x, \Delta y, \Delta\theta\}$. Specifically, we ran two sets of experiments, corresponding to two types of motion primitives generated using different kinematic constraints. The two types, referred to as "short" and "long" correspond to systems that can achieve higher (lower) velocity.

[2] http://www.sbpl.net/Software



((a)) BFS-WA* ((b)) Penalty-WA* ((c)) Barraquand ((d)) Gonzalez ((e)) RRT

Fig. 3: Solutions (red) and expanded states (heatmap) of each algorithm for 3D planning using short motion primitives. BFS-WA* and Penalty-WA* found better solutions than Barraquand and RRT, however, Gonzalez failed to find a solution within time limit. The number of states expanded in each experiment from (a) to (d) are $6,961$, $1,242,708$, $99,472$ and $896,254$, respectively. The number of samples generated by RRT (e) is $86,640$.



((a)) BFS-WA* ((b)) Penalty-WA* ((c)) Barraquand ((d)) Gonzalez ((e)) RRT

Fig. 4: Solutions (red) and expanded states (heatmap) of each algorithm for 3D planning using long motion primitives. BFS-WA* failed to find a solution without duplicate detection, while Penalty-WA* found one due to soft duplicate detection. Barraquand Gonzalez and RRT found solutions with equivalence classes or via sampling methods. The number of states expanded in each experiment from (a) to (d) are $440,217$, $18,380$, $173,523$ and $879,243$, respectively. The number of samples generated by RRT (e) is $13,434$.

3) Results and Analysis: We start with a qualitative comparison between the algorithms using the toy scenario depicted in Fig. 3. Here, there exists a kinematically-feasible path using the narrow passage, yet all algorithms except Penalty-WA* and BFS-WA* missed it. Here, due to the fact that we are using short motion-primitives, BFS-WA* can find a solution while expanding a small number of states while Penalty-WA* unnecessarily expands more states as it penalized some "useful" states. As we will see in Fig. 4, when using long motion primitives, the penalty will be much more beneficial when compared to BFS-WA*. Without the assistance of duplicate detection, BFS-WA* keeps on expanding states at the entrance of narrow passage until time runs out. On the contrary, Penalty-WA* sparsely searched the open area and densely searched the area near obstacles before it quickly exited this region. Both Gonzalez's and Barraquand's algorithms end up expanding multiple states at the entrance of the narrow passage since they perform (standard) duplicate detection. RRT, on the other hand, does not concentrate its search efforts at the entrance of the narrow passage but explores the entire search space and eventually finds an (highly sub-optimal) alternative path. We now move to a quantitative comparison between the algorithms performed on the benchmarks from the Moving AI lab. Results for the short motion primitives can be found in Table. I and Fig 5. The test cases where at least one algorithm failed are not considered in planning time and solution cost plots (the same for long motion primitives); As we can see Penalty-WA* was able to solve all problem instances doing so significantly and consistently faster than the alternative algorithms. The quality of the solutions obtained by RRT is higher than Penalty-WA* due to the fact that only the solutions of RRT are post processed (smoothed). These results back up our original intention that soft duplicate detection allows the planner to *reason* when and where to expand different states. Regarding long motion primitives, Fig. 4 visually compares BFS-WA* with Penalty-WA* and demonstrates that in highly-constrained environments, the soft duplicate detection employed by Penalty-WA* allows to efficiently reduce the number of states expanded. Results for the Moving AI Lab benchmarks for long motion primitives are consistent with those depicted of short motion primitives and are presented in Table II and Fig. 6

TABLE I: 3D planning using short motion primitives

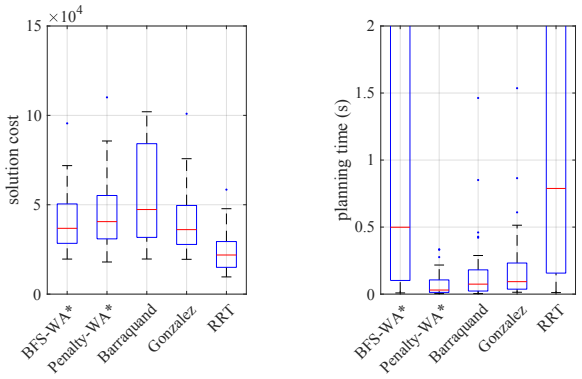
Algorithm	BFS-WA*	Penalty-WA*	Barraquand	Gonzalez	RRT
Success Rate (%)	95	**100**	77	**100**	91
Median Time (s)	0.50	**0.030**	0.075	0.093	0.79
Median Cost	36,815	40,555	47,322	36,079	**21,887**

TABLE II: 3D planning using long motion primitives

Algorithm	BFS-WA*	Penalty-WA*	Barraquand	Gonzalez	RRT
Success Rate (%)	95	**100**	29	**100**	61
Median Time (s)	0.0665	0.0585	**0.0195**	0.073	13.426
Median Cost	35,197	37,153	–	35,713	**33,422**

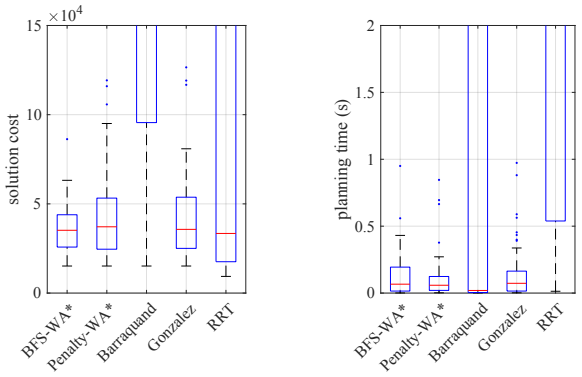
B. 5D Space Planning

1) Domain: In this setting we used 22 different scenarios where maps are mesh models of the real world with no-fly zones configured as obstacles. Start and goal pairs are generated randomly in the free space for each map. Map dimensions are 19.20 km $\times$ 19.20 km $\times$ 1.50 km with resolution of 20 m.



((a)) solution cost ((b)) planning time (s)

Fig. 5: 3D planning using short motion primitives.



((a)) solution cost ((b)) planning time (s)

Fig. 6: 3D planning using long motion primitives.

2) Motion primitives: As solving the two point BVP for actions with acceleration and non-linear turning radius is highly non-trivial, we generated motion primitives using a local controller. The input for the controller is the robot's acceleration a_{xy} planar and vertical velocity v_{xy} and v_z, respectively and yaw angular velocity ω.

3) Results and Analysis: We present a sample solution for the UAV in Fig. 7 and present the success rate, solution cost and planning time in Table. III and Fig. 8. In terms of success rate, Penalty-WA* is able to solve all but two of the scenarios. We attribute the failure to solve all scenarios to the fact that the BFS heuristic is not as informative in 5D as in 3D state space and note that using only this heuristic, BFS-WA* managed to solve only one scenario.

Similar to BFS-WA*, neither Barraquand's nor Gonzalez's algorithm were able to consistently solve these high-dimensional problems. This comes in stark contrast to RRT

TABLE III: 5D planning

Algorithm	BFS-WA*	Penalty-WA*	Barraquand	Gonzalez	RRT
Success Rate (%)	4.5	91	36	32	**100**
Median Time (s)	120	**8.80**	120	120	11.87
Median Cost	–	**10,585**	–	–	12,559

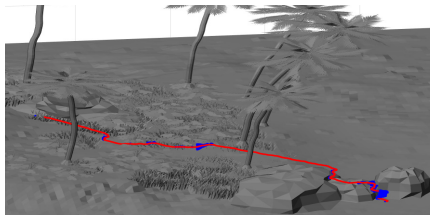


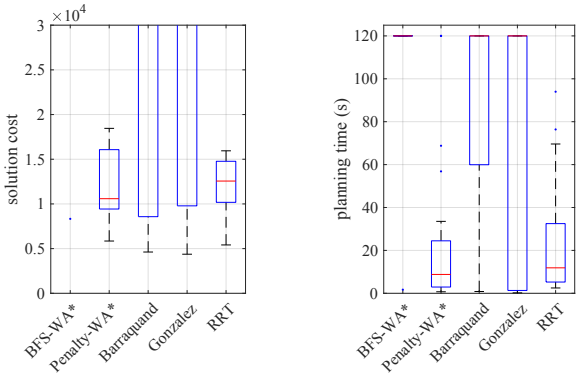
Fig. 7: Sample 5D solution. The blue area indicates local minima area, where more states are expanded. The red line is solution.

that managed to solve all problems (though with larger planning times). Indeed, it is often believed that sampling-based planners are preferable to search-based planners in solving high-dimensional motion-planning problems. However, as shown in previous work, we demonstrate that, by exploiting domain knowledge [27], search-based planners can outperform sampling-based planners in planning time and solution quality while providing their favorable traits such as deterministic behavior and provable bounds on the quality of the solution.

C. 8D Space Planning

1) Domain: The experimental setup is inspired by recent work by Ranganeni et al. [28] who employed search-based planning for a humanoid robot (though using a different approach). Specifically, we have one house setup (as shown in Fig. 9), with 80 randomly generated start and goal pairs. The dimension of the house are 23.8 m $\times$ 15.3 m $\times$ 5.2 m with resolution of 0.1 m.

2) Motion primitives: The motion primitives[3] are pre-computed. Before each action, one foot is selected as the pivot foot and the other foot is the active foot. The stop point of the active foot is calculated based on the action set $U_{\text{action}} = \{\Delta x, \Delta y, \Delta z, \Delta\theta\}$, which is precomputed given the joints limits of the robot. The linear part of actions in the set U_{action} are discretized with a resolution of 0.01 m



((a)) solution cost ((b)) planning time (s)

Fig. 8: Simulation results of 5D planning.

[3] https://github.com/wweedd/footstep_planner/tree/master/proto

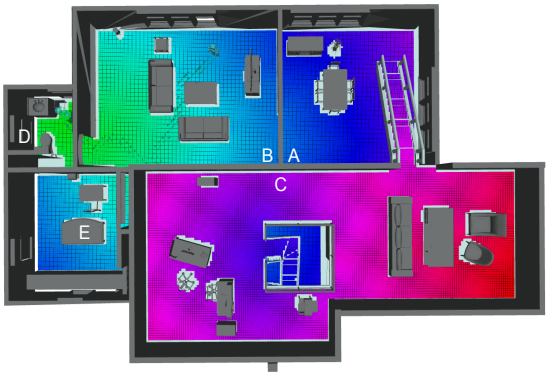


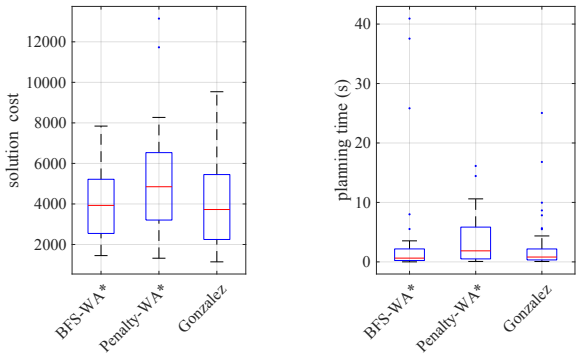
Fig. 9: Sample 8D solution. The color gradient from red to green represents the BFS heuristic value from high to low. The robot starts from the living room (region B), trying to figuring out a way to the bath room (region D).

TABLE IV: 8D planning

Algorithm	BFS-WA*	Penalty-WA*	Gonzalez
Success Rate (%)	56	**67**	55
Median Time (s)	**0.63**	2.58	2.32
Median Cost	3,929	4,850	**3,726**

in each direction with maximum step size of 0.3 m; The angular action resolution is 11.25°.

3) Results and Analysis: The experiment results from BFS-WA*, Gonzalez and Penalty-WA* are compared in terms of success rate, solution cost and planning time in Table. IV and Fig. 10. As is presented in Table. IV and Fig. 10(b), the Penalty-WA* algorithm still has the highest success rate compared to the other two algorithms. A possible explanation why the success rate of Penalty-WA* decreases for the 8D-dimensional space could be accounted for by an increased amount of local minima regions and an increased size of these minima (when compared to the lower-dimensional spaces). We believe that these local minima are caused due to (i) the complex kinodynamic constraints on the robot's feet and (ii) the BFS heuristic we are using in 8D space planning is less informative than it is in 3D or 5D. In terms of solution quality, penalizing states that could be useful leads to lower-quality solutions. For a given certain time limit, the Penalty-WA* algorithm has higher chances of finding one bounded solution. In terms of planning times, penalizing states requires performing an additional nearest-neighbor look up which has complexity of $O(\log |U|)$ [25] (recall that U is the set of all states in the OPEN and CLOSED lists). This is in contrast to the O(1) update time for the case of BFS-WA*. Notice that the big-O notation hides an exponential dependency on the dimension which explains why this phenomenon is accentuated in this high-dimensional setting. However, as is shown in Fig. 10(b) and Table. IV, although the Penalty-WA* algorithm runs slightly slower than the Gonzalez algorithm in terms of median values, the planning-time variance of Penalty-WA* is smaller. This implies that Peanlty-WA* has more consistent behavior in terms of planning time.



((a)) solution cost ((b)) planning time (s)

Fig. 10: Simulation results of 8D planning.

IV. CONCLUSIONS AND FUTURE WORK

We presented an approach to escaping local minimum regions for search-based planning in continuous spaces. While improving the success rate of planner, the Penalty-WA* algorithm requires longer planning time compared to other baseline algorithms. Improving its speed is one of the directions for future work. During the planning process, the duplicity of states are frequently calculated. Unfortunately, designing an appropriate metric is non-trivial and more research effort on this topic is required to improve the performance of the current algorithm. As search-based planners commonly suffer from the curse of dimensionality, we will also strive to improve the success rate in high-dimensional spaces.

V. ACKNOWLEDGEMENTS

This work was in part supported by ONR grant N00014-18-1-2775.

REFERENCES

[1] B. Triggs, "Motion Planning for Nonholonomic Vehicles: An Introduction," 1993, survey paper presented at Seminar on Computer Vision and Robotics, Newton Institute of Mathematical Sciences, Cambridge, England.

[2] J. P. Laumond, "Finding collision-free smooth trajectories for a non-holonomic mobile robot." in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 1987, pp. 1120–1123.

[3] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie, "Planning walking patterns for a biped robot," *IEEE Trans. Robotics and Automation*, vol. 17, no. 3, pp. 280–289, 2001.

[4] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 2172–2179.

[5] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2902–2908.

[6] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2872–2879.

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

[8] T. Ishida, "Moving target search with intelligence," in *Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, July 12-16, 1992.*, 1992, pp. 525–532.

[9] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, pp. 122–155, 1993.

[10] J. P. Gonzalez and M. Likhachev, "Search-based planning with provable suboptimality bounds for continuous state spaces," in *Symposium on Combinatorial Search, (SoCS)*, 2011, pp. 60–67.

[11] P. A. Dow and R. E. Korf, "Duplicate avoidance in depth-first search with applications to treewidth." in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2009, pp. 480–485.

[12] R. Zhou and E. A. Hansen, "Domain-independent structured duplicate detection," in *AAAI Conference on Artificial Intelligence*, 2006, pp. 1082–1088.

[13] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc., Reading, MA, 1984.

[14] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *The International Journal of Robotics Research (IJRR)*, pp. 378–400, 2001.

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

[16] C. Danilo, K. Przemyslaw, F. Mirko, and P. Lucia, "Motion primitive based random planning for loco-manipulation tasks," in *International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1059–1066.

[17] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *arXiv preprint arXiv:1809.02399*, 2018.

[18] A. J. Dionne, J. T. Thayer, and W. Ruml, "Deadline-aware search using on-line measures of behavior," in *Symposium on Combinatorial Search, (SoCS)*, 2011, pp. 39–46.

[19] F. Islam, O. Salzman, and M. Likhachev, "Online, interactive user guidance for high-dimensional, constrained motion planning," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018, pp. 4921–4928.

[20] F. Matoui, B. Boussaid, and M. N. Abdelkrim, "Local minimum solution for the potential field method in multiple robot motion planning task," in *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. IEEE, 2015, pp. 452–457.

[21] J. Velagic, B. Lacevic, and N. Osmic, "Efficient path planning algorithm for mobile robot navigation with a local minima problem solving," in *IEEE International Conference on Industrial Technology (ICIT)*, 2006, pp. 2325–2330.

[22] S. Vats, V. Narayanan, and M. Likhachev, "Learning to avoid local minima in planning for static environments," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2017, pp. 572–576.

[23] M. Phillips and M. Likhachev, "Speeding up heuristic computation in planning with experience graphs," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 893–899.

[24] I. Pohl, "The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving," in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 1973, pp. 12–17.

[25] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[26] N. Sturtevant, "Benchmarks for grid-based pathfinding," *Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144 – 148, 2012. [Online]. Available: https://movingai.com/benchmarks/grids.html

[27] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-Heuristic A*," *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 1-3, pp. 224–243, 2016.

[28] V. Ranganeni, O. Salzman, and M. Likhachev, "Effective footstep planning for humanoids using homotopy-class guidance," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2018, pp. 500–508.