PROTAMP-RRT: A Probabilistic Integrated Task and Motion Planner based on RRT

Supplemental material

Alessio Saccuti, Riccardo Monica and Jacopo Aleotti

Department of Engineering and Architecture
University of Parma
Parco Area delle Scienze 181/A, 43124 Parma, Italy
{alessio.saccuti, riccardo.monica, jacopo.aleotti}@unipr.it

Abstract

This document explains in more detail parts of the method illustrated in the associated paper "PROTAMP-RRT: A Probabilistic Integrated Task and Motion Planner based on RRT". In particular, we provide a discussion of probabilistic completeness of the proposed PROTAMP-RRT algorithm (Section A), a discussion and evaluation of the hyperparameters (Section B) and an analysis of the algorithm scalability (Section C).

A Probabilistic completeness

In this section, we outline a proof that PROTAMP-RRT (Algorithm 1) is probabilistically complete, i.e., that as the planning time increases, the probability that a solution is found, if one exists, approaches one. This proof applies only for a finite symbolic space, i.e. when sampling of new significant object poses is disabled and a finite set of significant object poses is pre-defined. Proving probabilistic completeness of the whole algorithm is likely more complex, as the symbolic space can grow without limit. We also suspect that PROTAMP-RRT is not probabilistically complete, but that it may be made probabilistically complete with minor modifications. For example, it may be made probabilistically complete by randomly selecting a random symbolic action from time to time.

The proof of probabilistic completeness proceeds as follows. First, we prove that if the currently selected plan α_{best} is infeasible, the task planner will eventually be called to find a new symbolic plan, with probability approaching one (Theorem 1). Then, we prove that if there exists at least a feasible symbolic plan, then a feasible plan will be selected as α_{best} a number of times approaching infinity as the planning time increases (Theorem 2). Finally, we will prove

Algorithm 1 PROTAMP-RRT planning algorithm

```
Input: $\mathcal{E}$, $\mathcal{O}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{G}$, $q_0$, $s_0$
Output: $\alpha_{best}$, $\mathcal{T}$
 1: $\mathcal{P} \leftarrow \varnothing$, $\mathcal{T} \leftarrow \{(s_0, q_0)\}$, $P_{best} = 1$
 2: while true do
 3:                                                        ▷ Task planning
 4:     if $\alpha_{best} = \varnothing$ or $\mathcal{P}(\alpha_{best}) < P_{best} P_{mult}$ then
 5:         $\alpha_{best} \leftarrow TaskPlanner(\mathcal{O}, \mathcal{A}, \mathcal{L}, \mathcal{G}, s_0, \mathcal{P})$
 6:         $r \leftarrow ReachableActions(\mathcal{T}, \alpha_{best})$
 7:         while $\mathcal{P}_{obj}(\alpha_{best}) \leq P_{low}^{|\alpha_{best}|-(r-1)}$ do
 8:             $\mathcal{L} \leftarrow \mathcal{L} \cup \{SampleNewObjectPose(\mathcal{R})\}$
 9:             $\alpha_{best} \leftarrow TaskPlanner(\mathcal{O}, \mathcal{A}, \mathcal{L}, \mathcal{G}, s_0, \mathcal{P})$
10:             $r \leftarrow ReachableActions(\mathcal{T}, \alpha_{best})$
11:         $P_{best} \leftarrow \mathcal{P}(\alpha_{best})$
12:     $q_{goal} \leftarrow q_{end}(a_r)$                          ▷ Motion planning
13:     $q_{new} \leftarrow SampleState(\mathcal{T}, q_{goal})$
14:     $C_{new} \leftarrow CheckCollisions(q_{new}, s_{r-1})$
15:     $\mathcal{P} \leftarrow UpdateProb(\mathcal{P}, \mathcal{E}, \mathcal{T}, a_r, q_{new}, q_{goal}, C_{new}, s_{r-1})$
16:     if $C_{new} = \varnothing$ then
17:         $\mathcal{T} \leftarrow Extend(\mathcal{T}, (s_{r-1}, q_{new}))$
18:         if $q_{new} = q_{goal}$ and $r = |\alpha_{best}|$ then
19:             return $\alpha_{best}$, $\mathcal{T}$
20:         if $q_{new} = q_{goal}$ then
21:             $SetSolved(\mathcal{P}, a_r, s_{r-1})$
22:             $r \leftarrow r + 1$
```

that, if the current plan α_{best} is feasible, then there is a nonzero probability that the motion planner finds a solution before the generation of a new plan is triggered (Theorem 3). Therefore, as the motion planner has potentially any number of attempts to find the solution, for planning time approaching infinity the probability that the solution is found approaches one.

Theorem 1. *If the currently selected plan α_{best} is infeasible, the task planner will eventually be called to find a new symbolic plan, with probability approaching one.*

Proof. If α_{best} is infeasible, then there exists an action a_r whose sub-goal $q_{end}(a_r)$ is unreachable in symbolic state s_{r-1}. As there must be an obstacle preventing the action from reaching the goal, there is always a nonzero probability of sampling a robot configuration which is in collision. The collision may be either with the environment (hence it affects the environmental probability) or a movable object (hence it affects the object conditional probability). As the two kinds of probabilities are updated in the same way, in the following we simply use the term "probability" for both. Whenever a colliding configuration is sampled, probability of that action in the probabilistic model is reduced according to Section III-B. Conversely, probability is increased only when the sampled configuration is not in collision and it is closer to the sub-goal than all other configurations in $\mathcal{T}$ with the same symbolic state s_{r-1}. The probability of finding a new non-colliding configuration closer to the sub-goal decreases exponentially as more configurations are added to $\mathcal{T}$, as the new configuration must be closer than all other already-sampled configurations. Therefore, as the number of sampled configurations increases, the expected number of non-colliding configurations sampled closer to the goal approaches a finite value, while the number of colliding configurations grows unlimitedly. Hence, the estimated probability of action a_r in state s_{r-1} in the probabilistic model tends to zero. Therefore, also $\mathcal{P}\left(\alpha_{best}\right) \to 0$, and in particular it will eventually be lower than $P_{best}\, P_{mult}$, so the task planner will be triggered to find a new plan (line 4 of Algorithm 1). □

Theorem 2. *If there exists at least a feasible symbolic plan, then a feasible plan will be selected as α_{best} a number of times approaching infinity as the planning time increases.*

Proof. If $\mathcal{L}$ contains a finite number of significant object poses, the symbolic state is finite. Hence, the number of symbolic plans is also finite, as each symbolic plan traverses each symbolic state at most once. Otherwise, if the same symbolic state was traversed at two points in the plan, a shorter plan with higher probability would be found by removing the actions in-between. According to Theorem 1, if the current symbolic plan is infeasible the task planner is eventually called to obtain a new symbolic plan. We now prove that eventually the returned plan is one of the feasible plans. Let's assume by contradiction that, starting at some point in time, all plans returned by the task planner are from a subset which contains only infeasible plans. By definition, each of these plans has at least one infeasible action a_r, whose sub-goal cannot be reached by the motion planner in symbolic state s_{r-1}. As RRT is probabilistically complete, and the number of plans is finite, eventually it is able to reach the sub-goal of all actions of all plans in the subset, up to the first infeasible action. When this happens, for Theorem 1 the motion planner eventually calls the task planner with the probability of the infeasible action in symbolic state s_{r-1} reduced at least by a factor $1/P_{mult}$. This operation also reduces the probability of the same action in symbolic states different from s_{r-1} (which may belong to feasible plans), but by a lesser extent, as the same action in other symbolic states does not share all the object conditional probabilities. Hence, eventually the probability of all symbolic plans in the subset of infeasible plans will be below one of the feasible plans, and the task planner returns a feasible plan which is not in the subset. We reached a contradiction, hence we can conclude that a feasible symbolic plan will be eventually returned starting from any point in time. Therefore, as planning time increases a feasible symbolic plan is selected a number of times approaching infinity. □

Theorem 3. *Given a feasible symbolic plan α_{best}, the motion planner has a nonzero probability of finding a corresponding motion trajectory before the task planner is triggered.*

Proof. As long as robot configurations which are in collision are not sampled, the probability is never reduced in the probabilistic model, and therefore the task planner cannot be triggered. Hence, the RRT has a nonzero probability of finding the trajectory which is at least the probability of sampling all the robot configuration of the trajectory without sampling a colliding state. $\square$

B Hyperparameter evaluation

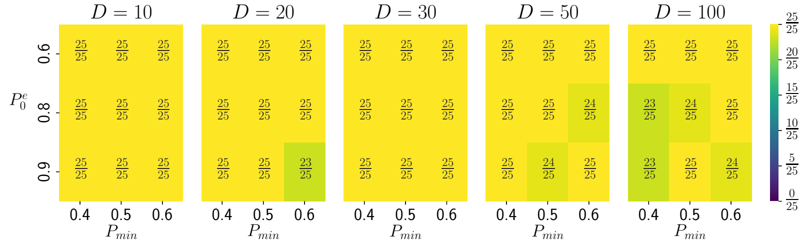
In this section, we discuss the hyperparameters in PROTAMP-RRT and show their effect on the TAMP method performance. In particular, there are ten configuration parameters for our approach: N_0^e, D_0^e, N_0^c, D_0^c, N_{min}, D_{min}, P_H, P_{mult}, P_{low}, P_{goal}. We analyze these parameters separately as two groups: parameters N_0^e, D_0^e, N_0^c, D_0^c, N_{min}, D_{min} affect the probabilistic model, while P_H, P_{mult}, P_{low}, P_{goal} affect the behavior of the task and the motion planner.

In the first group , N_0^e and D_0^e are the numerator and denominator of the initial environmental probability $P_0^e = N_0^e/D_0^e$, which represents the *a priori* environmental probability for a new action. Similarly, N_0^c and D_0^c are the numerator and denominator of the initial object conditional probability $P_0^c = N_0^c/D_0^c$, which represents the *a priori* conditional probability $\mathcal{P}_c\left(a|c\right)$ when a new object-pose pair c affecting action a is discovered. Parameters N_{min} and D_{min} are the numerator and denominator used to restore the probability to a higher value $P_{min} = N_{min}/D_{min}$ when the motion planner samples a new valid robot configuration towards an action sub-goal. This probability should be high if the geometric space has many dead ends (local minima) in which the motion planner can get stuck, so that the probability is reset to an higher value when the planner manages to escape one of these local minima. Changing the numerators and the denominators of the first group of parameters, while keeping P_0^e, P_0^c and P_{min} constant, changes the impact of a single probability increase or reduction on the probability value. When numerators and denominators are low, increments and decrements result in a larger impact, hence the probabilities change faster and, therefore, the task planner will be called more frequently if sampling of valid configurations fails. Hence, lower values are suited for simpler scene geometry, where just a few failures likely mean that the action is infeasible and that a different symbolic plan should be selected.

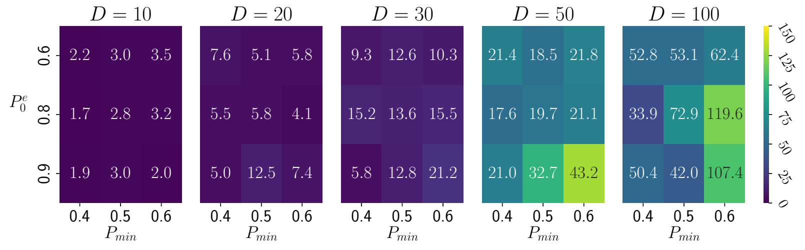
The parameters in the second group (P_H, P_{mult}, P_{low} and P_{goal}) affect the behavior of the task and motion planner. The first parameter P_H is used to compute the $A^\star$ heuristic:

$$H(s_{|\alpha'|}) = (P_H)^{U(s_{|\alpha'|})} \tag{8}$$

where $U(s_{|\alpha'|})$ is the number of unsatisfied conditions in the current symbolic state with respect to the task goal. Parameter P_{mult} is used to trigger replanning of the task plan. That is, when the probability of the current plan is reduced by less than P_{mult} multiplied by its original value, the task planner is called to potentially find a better symbolic plan. A lower value of P_{mult} means that the RRT is allowed more sampling failures before switching to a different task plan,



(a) Success rate.



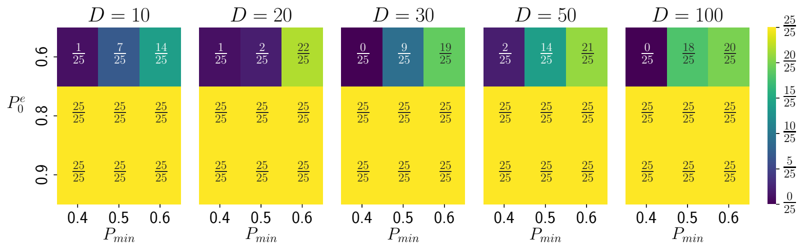
(b) Planning time (seconds).

Figure 11: Success rate (a) and average planning time (b) for the first group of parameters, in the *Transfer 2* task. Values are displayed inside each cell.

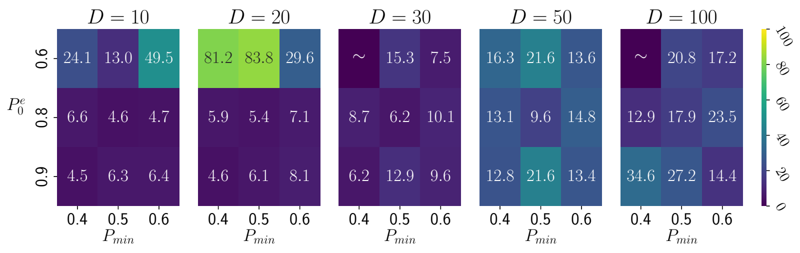
and it is therefore suited for more complex scene geometry where the RRT needs more planning time. Parameter P_{low} is the threshold to sample new significant object poses: the higher P_{low}, the more frequently new poses are sampled. The last parameter, P_{goal}, is the probability that the RRT motion planner attempts an extension towards the action sub-goal, instead of sampling randomly.

We analyzed the effect of the parameters on the planning algorithm by executing a grid search for each group. During the evaluation of one group, the parameters of the other group were fixed as reported at the end of Section IV-A in the main article. Planning was executed 25 times for each parameter combination, on two tasks: *Transfer 2* and *Culprit*. The number of successes after 600 seconds and the average planning time were recorded.

For the first group of parameters (N_0^e, D_0^e, N_0^c, D_0^c, N_{min}, D_{min}), all denominators (D_0^e, D_0^c, D_{min}) were set equal to the same value D. Moreover, the numerator of the initial object conditional probability N_0^c was set to $N_0^c = D_0^c - 1$, so that P_0^c is only slightly less than 1. Indeed, when a new object conditional probability is initialized to P_0^c, the collision with that object-pose pair has occurred only once, and therefore the probability of the plan should not change significantly. These choices leave three degrees of freedom for the grid search: D, P_0^e and P_{min}. The following values were tested: $D \in \{10, 20, 30, 50, 100\}$, $P_0^e \in \{0.6, 0.8, 0.9\}$ and $P_{min} \in \{0.4, 0.5, 0.6\}$. Results are reported in Fig. 11 and Fig. 12. Generally, PROTAMP-RRT is rather robust against parameter



(a) Success rate.



(b) Planning time (seconds).

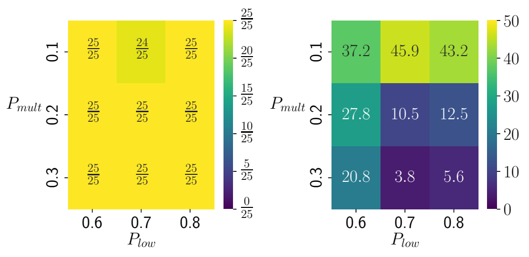
Figure 12: Success rate and average planning time for the first group of parameters, in the *Culprit* task. Values are displayed inside each cell.

changes, as most parameter combinations resulted in only a few failures. However, it can be observed that with $P_0^e = 0.6$ PROTAMP-RRT is unable to find a solution in the *Culprit* task, as probability of new actions is too low and they are never selected. Also, average planning time is lower for smaller D in both tasks, which means that in these particular tasks it is more efficient to change symbolic plan after only a few sampling failures.

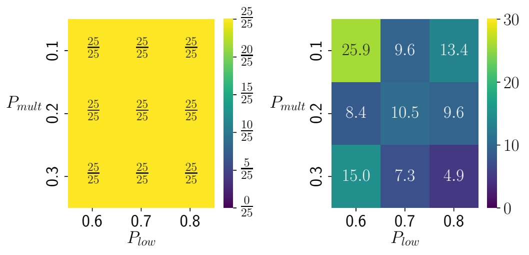
For the second group of parameters (P_{goal}, P_H, P_{mult} and P_{low}), parameter P_{goal} was fixed to 0.3 which is a common value for this parameter in RRT-based approaches (e.g., also used in TM-RRT by Finzi et al.). To provide a consistent heuristic for $A^\star$ (equation 8), parameter P_H should be set greater or equal than the joint probability of the actions needed to change an unsatisfied condition to a satisfied one. To select an approximate value for P_H, we consider that, as in our tasks conditions are represented by objects located in regions, the planner needs at least two actions (a pick and a place) to satisfy a goal condition. Hence, we choose P_H as the squared *a priori* environmental probability $P_0^e = N_0^e / D_0^e$ for an action, increased by 0.5%:

$$P_H = 1.005 \cdot (P_0^e)^2 \tag{9}$$

A grid search was carried out for the two remaining parameters, P_{mult} and P_{low}, with the possible values: $P_{mult} \in \{0.1, 0.2, 0.3\}$ and $P_{min} \in \{0.6, 0.7, 0.8\}$. Results are shown in Fig. 13. The number of failures is very low for all tested



(a) *Transfer 2* task



(b) *Culprit* task

Figure 13: Success rate (left) and average planning time in seconds (right) of the grid search for P_{mult} and P_{min}, in the *Transfer 2* and the *Culprit* tasks. Values are displayed inside each cell.

combinations of the parameters and in both tasks. Generally, there is a noticeable planning time increase for low values of P_{low}, which triggers the sampling of new significant object poses less frequently. Similarly, high values of P_{mult} resulted in lower planning times. These observations confirm that in these particular tasks it is more efficient to change symbolic plan and sample new significant object poses more frequently.

C Scalability

In this section we report experimental results about the scalability of our approach. We planned the *Transfer 2* task with a progressively increasing number of movable objects, up to 8 objects. For each number of objects, planning was repeated 25 times. In order to have enough space to relocate the objects, we increased the area of the *Place table* region proportionally to the number of the available movable objects. In particular, for n objects, the table side length was computed as:

$$l_{table} = \frac{l_0}{\sqrt{n_0}}\sqrt{n} \tag{10}$$

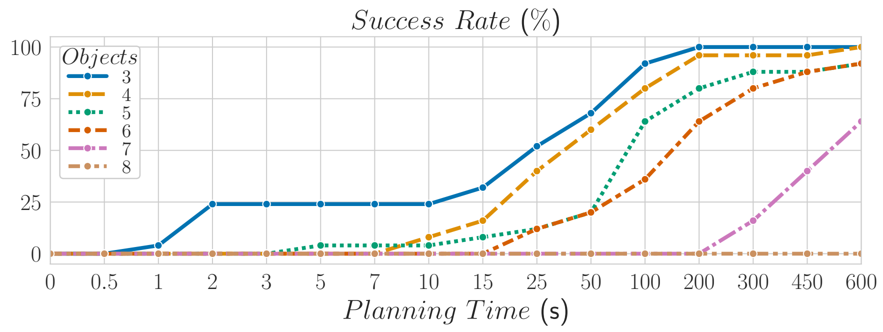


Figure 14: Scalability test in the *Transfer 2* task.

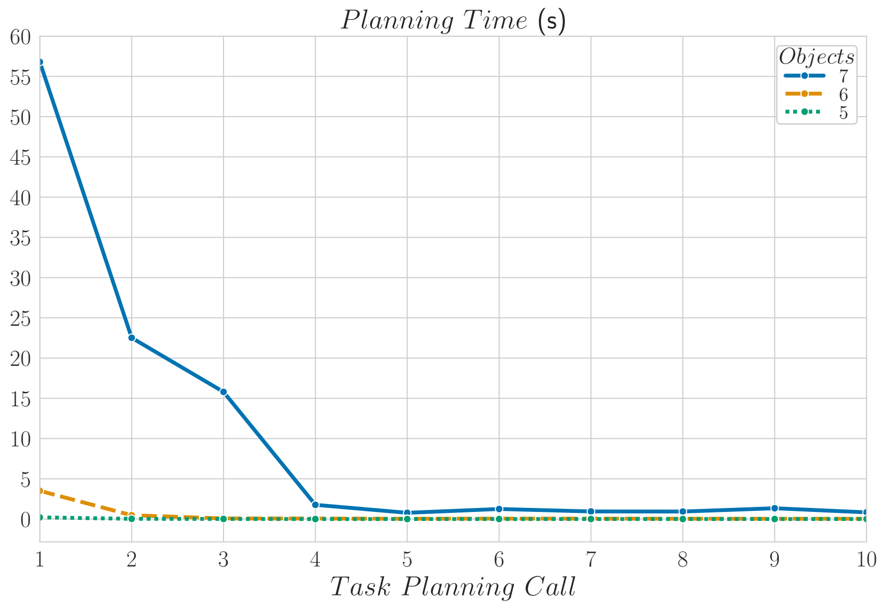


Figure 15: Average planning time of the first 10 task planning calls for 5, 6 and 7 objects.

where $l_0 = 0.4$ and $n_0 = 3$ are, respectively, the initial table side length and the number of boxes in the *Transfer 2* task.

The results of the scalability test are shown in Figure 14, where we report the success rate as a function of planning time for each number of objects. The success rate decreases as the number of movable objects increases, reaching 0 for 8 objects. The main cause is that, for many objects, the $A^\star$ task planner has high planning time and memory usage, eventually filling the 32 GB of available

RAM. The issue is most evident in Figure 15, in which the average planning times of the first 10 task planner calls are shown. At the beginning of the execution, when most actions have the same probability as only a few of them have been attempted by the motion planner, the task planner operates as a simple breadth-first search, and it is forced to enumerate all possible task plans with minimum length. Therefore, the first calls of the task planner require long planning times. We hypothesize that better performance could be obtained with a different task planner which combines both breadth-first and depth-first search.