Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)

Task-Aware Waypoint Sampling for Robotic Planning

Sarah Keren[1,2]**,Gerard Canal**[3]**, Michael Cashmore**[4]

[1]Harvard University, School of Engineering and Applied Sciences, Cambridge, USA
[2]The Hebrew University of Jerusalem, School of Computer Science and Engineering, Israel
[3]Department of Informatics, King's College London, UK,
[4]University of Strathclyde, Computer and Information Sciences, Glasgow, UK

Abstract

To achieve a complex task, a robot often needs to navigate in a physical space in order to complete activities in different locations. For example, it may need to inspect several structures, making multiple observations of each structure from different perspectives. Typically, the positions from which these activities can be performed are represented as *waypoints* – discrete positions that are sampled from the continuous physical space and used to find a task plan. Existing approaches to waypoint selection either iteratively consider the entire space or use domain knowledge to consider each activity separately. This can lead to task planning problems that are more complex than is necessary or to plans of compromised quality. Moreover, all previous approaches only consider geometric constraints that can be imposed on the waypoint selection process.

We present Task-Aware Waypoint Sampling (TAWS), which offers two key novelties. First, it is an anytime approach that combines the benefits of random sampling with the use of domain knowledge in waypoint selection by performing a one-time computation of the connectivity graph from which waypoints are sampled. In addition, TAWS is the first approach that accounts for *performance preferences*, which are preferences a system operator may have about the generated task plan. These can account, for example, for areas near doorways where it is preferable that the robot does not stop to perform activities. We demonstrate the performance benefits of our approach on simulated automated manufacturing tasks.

Introduction

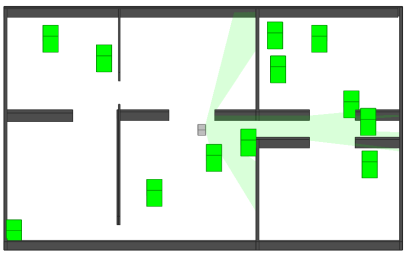
Robots are typically assigned complex missions that require performing various activities in different locations. To complete the overall mission, a mobile robotic agent must reason about a physical space and decide both which activities must be performed as well as how to navigate between the positions from which it can perform each activity. Since the physical space is continuous, task planning is typically performed using an abstraction of the space. A common abstraction approach is to use a finite set of discrete *waypoints* that represent configurations (positions) in the space. The waypoints represent nodes in a *probabilistic road map* (PRM) (Kavraki et al. 1996), in which the edges represent feasible paths between waypoints and their estimated navigation costs. The PRM is used by a *task planner* to find a feasible sequence of activities and navigations between waypoints from which the activities can be performed that accomplishes the assigned task.

When generating waypoints there is a trade-off between the complexity and completeness of the resulting representation. Intuitively, a small set of waypoints is a coarse abstraction of the physical space that limits the positions that can be used to achieve the task, potentially leading to lower quality plans or unsolvable problems. On the other hand, a larger set of waypoints will lead to a higher probability of finding a plan, but may exceed the computational capacity of the task planner.

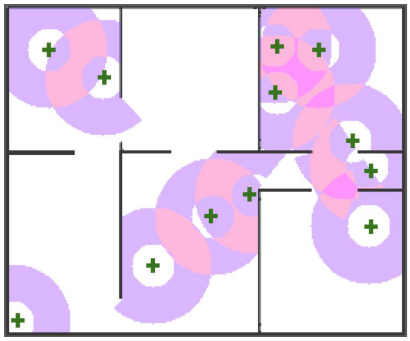
Generally, there are two common approaches to generating a set of waypoints. With *Fixed Waypoint Generation* (FWPG), a set of waypoints is generated by selecting a single waypoint for each possible activity (Edelkamp et al. 2018). FWPG provides a sufficient coverage of the planning space, but may yield representations that are too big for the planner to handle. On the other end, with a *Pure PRM* (PPRM) approach (Kavraki et al. 1996), a PRM is created by randomly sampling waypoints. The size of the graph can be set to comply with the planner's capacity, but since the placement of waypoints is random, the coverage of the space may be insufficient. This may require iteratively generating a new PRM until a solution is found.

In this work, we suggest a novel approach to waypoint generation which bridges the gap between the two common approaches to sampling and provides good coverage of the space, while accounting for the planner's capacity. Our approach, *Task-Aware Waypoint Sampling* (TAWS), is an anytime approach that starts by generating a very *Dense PRM* (DPRM). The DPRM captures a fine representation of the reachability information of the space, and includes with very high probability a representation of a solution to the task. TAWS then launches an iterative, anytime planning process, sampling at each step waypoints from the DPRM according to probabilities induced by the task description. If a plan cannot be found using the current set of waypoints or if there is time to improve the quality of the current best solution, a larger set is re-sampled. In contrast, if the planner's capacity is exceeded, a smaller set is re-sampled.

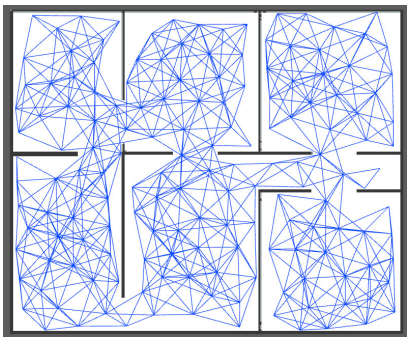
As with PPRM, TAWS relies on sampling a set of waypoints to generate the PRM used by the task planning. However, it avoids the need to reconstruct a PRM at every plan-



(a) 3D representation of an example factory environment, with the robot (gray square in the center), and the machines in green.

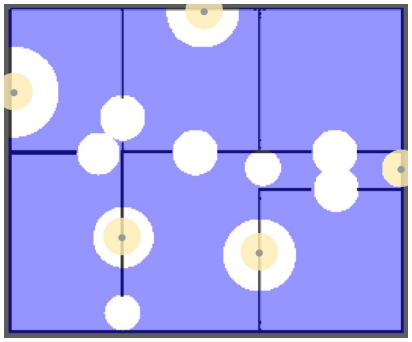


(b) A map of the physical space highlighted by the areas from which the robot can interact with each of 12 different machines (green crosses).

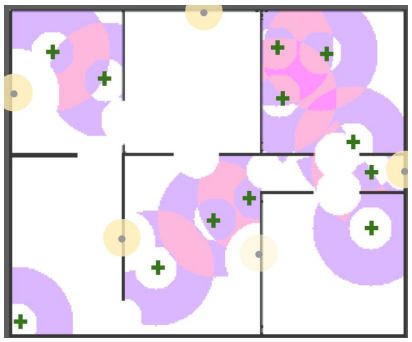


(c) The dense PRM imposed over the map. Each collision-free edge (blue lines) connects two waypoints (configurations).

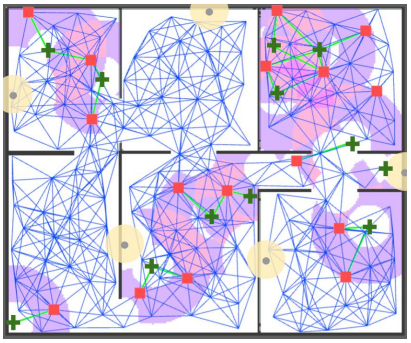
Figure 1: An example setting from the RCLL domain



(a) Map of the physical space specifying first aid stations (yellow) and doorways (white), which represent performance preferences of areas where robots can never block and preferably should avoid, respectively.



(b) Map representing the combination of the performance preferences and the machine interaction areas.



(c) Waypoints (red squares) are sampled from the DPRM for task planning. Machines that can be accessed from each waypoint are highlighted by green edges.

Figure 2: Integrating Performance preferences into the sampling process

ning iteration. As with FWPG, TAWS incorporates domain knowledge into the waypoint selection process, but instead of using it to fix a set of waypoints, it uses it to set the probabilities according to which waypoints are sampled from the DPRM. This can be used to increase efficiency by, for example, prioritizing waypoints from which more than one activity can be performed. Moreover, it makes it possible to account for *performance preferences*, arbitrary user-defined preferences over positions from which the robot can perform its activities but that cannot be directly represented in the map used by the robot for navigation. Such preferences can reflect, for example, social norms (e.g., areas where some social event is taking place and should be avoided by noisy robots), safety and efficiency constraints (e.g., a carpeted area that is hard for robots to traverse, or an area near a first aid kit robots shouldn't block), and areas where performance is enhanced (e.g., it is preferable for a robot to operate near a charging station since it will be able to recharge and recover if its battery is unexpectedly depleted). TAWS can account for these arbitrarily defined preferences by changing the probability of sampling certain positions according to the specified preferences.

Example 1 *Consider the scenario used for the RoboCup Logistics League (RCLL) (Niemueller, Lakemeyer, and Ferrein 2015) and depicted in Figure 1, in which robots must navigate in a factory in order to collect items from a set of machines and deliver them to their destinations. In such scenarios, if the environment is fixed and known, FWPG can be used to prescribe a finite set of waypoints, including a waypoint for each activity robots may need to perform (e.g., picking up an item from a machine). The result may include many redundant waypoints or waypoints that cannot be connected, or lead to inefficient plans since each activity is considered separately. Also, FWPG does not allow for iterations if the planner's capacity is exceeded. On the other hand, the PPRM approach might require many iterations to solve problems of realistic size or produce inefficient plans since the size of the PRM and the accuracy of its cost estimations*

are limited by the planner's capacity.

TAWS takes a hybrid approach by first producing a single dense PRM (DPRM) that is used throughout the search for a plan. This is likely to lead to plans that are more efficient at execution time since the DPRM provides more accurate navigation cost estimates. Moreover, TAWS can be used to reduce the probability that robots' plans include waypoints that block doorways or to guarantee not to include waypoints that block access to first aid stations.

We offer two key contributions. First, we suggest performing a one-time computation of a connectivity graph in a given environment, thus decoupling between the connectivity analysis and the task planning process. This enables the use of high quality navigation cost estimations throughout the planning process, regardless of the size of the current representation of the task planning problem (the current PRM) or the capacity of the task planner. Secondly, we support performance preferences that induce the waypoint sampling probabilities and yield plans that comply with both hard constraints and with user-defined preferences regarding the way a robot accomplishes its task.

To demonstrate the performance benefits of TAWS, we use a set of simulated manufacturing tasks in an automated factory. First, we show that the use of a DPRM instead of an iterative or fixed generation of a PRM yields shorter plans that are computed more efficiently. In addition, we show that TAWS produces plans that maximize compliance with the specified performance preferences when compared to current approaches, without compromising their quality.

Related Work

Typical robotic control systems must determine which activities need to be performed, and how to navigate between the activities. Common approaches to planning for robots combine *motion planning* and *task planning* (Gravot, Cambon, and Alami 2005; Cambon, Alami, and Gravot 2009; Kaelbling and Lozano-Pérez 2011; Dornhege, Hertle, and Nebel 2013; McMahon and Plaku 2014; Srivastava et al. 2014; Toussaint 2015; Fernández-González, Karpas, and Williams 2017; Canal et al. 2018; Garrett et al. 2020). Motion planning is the process of finding a way to perform a basic activity, such as picking up an item or moving between two adjacent locations. Task planning is the search for a sequence of activities that is predicted to achieve the goal while minimizing duration and other costs such as energy use.

When planning in complex scenarios, task planning typically uses an abstraction of the space. One way to abstract the space is by using geometric computations that help the high-level planner make appropriate choices. For example, (Kaelbling and Lozano-Pérez 2011) handle the integration of continuous geometric planning with task planning by using geometric "suggesters", which construct configurations dynamically during an "aggressively" hierarchical planning process. Another approach integrates the motion planner's geometric search for positions into the symbolic forward-search of a task planner. For example, (Cambon, Alami, and Gravot 2009) offer an integrated task and motion planner that reasons about geometric constraints that describe the positions from which it is possible to accomplish some activity as sub-manifolds of the configuration space of the robot. These sub-manifolds are mapped to high-level symbols that can be used by a task planner.

Another approach to abstraction uses *waypoints* that represent discrete positions (Cashmore et al. 2014; McMahon and Plaku 2014; Edelkamp et al. 2018). This reduces the complexity of the problem, making it possible to focus on the task-planning aspect of the problem, i.e., selecting and scheduling activities, while using heuristic approximations to estimate navigation and motion costs. Once a high-level task plan is produced, motion planning is delegated to a low-level motion planner. One of the benefits of this approach is that it is typically agnostic to the specific task and motion planners used.

In this paper we focus on waypoint-based approaches and on the selection of waypoints for task planning. Waypoints can be selected randomly, for example using a PRM (Kavraki et al. 1996), or can be generated using knowledge of the space and task (Plaku and Hager 2010; Edelkamp et al. 2018). The disadvantage of the random approach is that in order to ensure coverage of all interesting areas, a large number of waypoints might be required. For simple problems, such as inspection missions (Cashmore et al. 2014), this can be feasible. However, in a more complex task this will result in problems that are too hard to solve within a reasonable time.

On the other hand, generating fixed waypoints means that for each affordance in the physical space (corresponding to a non-navigation action that the robot might make) a set of waypoints of fixed size is generated. To demonstrate, in the RCLL setting in Example 1, the approach by (Edelkamp et al. 2018) generates a separate waypoint for each item pickup activity by randomly sampling a position around the machine the item is positioned at. This holds even if the machine has more than one item. These waypoints are connected together using a PRM, adding additional waypoints to cover the space, if needed. The resulting representation is guaranteed to include a solution. However, as the number of activities increases, the size of the resulting task planning problem may unjustifiably exceed the planner's capacity, containing many redundant waypoints.

Another limitation of the fixed waypoint selection approach is that it completely relies on domain knowledge to select a waypoint for each activity. In some cases such domain knowledge may not be available. In domains with complex configuration spaces, it may not be possible to explicitly prescribe in advance the region from which an activity can be performed, making it necessary to sample waypoints and determine whether an activity is achievable from them. For example, consider a mobile base carrying an arm with 5-degrees of freedom, performing a picking up task in a cluttered scene. Due to the clutter, it is not possible to describe in advance a region for the base from which it is guaranteed that the arm can reach the target. However, it is possible to sample instead a position and orientation for the base and use a motion planner to determine if there is a collision-free path for the arm to the target.

We suggest a new approach to waypoint sampling that

combines the benefits of random sampling with the use of domain knowledge. In contrast to a fixed approach to waypoint selection, TAWS is an anytime approach that iteratively improves solution quality. In contrast to a random sampling approach, it decouples the connectivity analysis from task planning by creating and reusing a single dense PRM (DPRM). TAWS uses domain knowledge to induce the probabilities according to which waypoints are sampled from the DPRM. This means that the quality of navigation cost estimations does not depend on the current number of waypoints that are used to represent the task. Most notably, all approaches mentioned above only consider geometric constraints that can be imposed on the waypoint selection process. TAWS is the first approach that also accounts for arbitrary performance preferences, thus making it possible to prioritize or discourage specific behaviors.

Task Aware Waypoint Sampling (TAWS)

The input to a *Task Aware Waypoint Sampling* (TAWS) problem is a tuple $p = \langle M, A, F \rangle$, where

- M is the set of configurations $m \in \mathbb{R}^n$, where n represents the dimensions of the space,
- A is a set of non-navigation activities that can be performed, and
- F is a set of performance preferences. Each preference $f \in F$ is a score function $f : M \to \mathbb{R}_{\geq 0}$.

Each sampled waypoint corresponds to a configuration $m \in \mathbb{R}^n$. Each activity $a \in A$ is associated with a function $\omega_a : M \to [0, 1]$ specifying the probability of successfully executing a from configuration m. Typically, these probability functions are generated using prescribed templates for each activity type the robot can perform. Preferences $f \in F$ are used to describe areas from which it is (un)desirable that the robot operates.

In Example 1, a robot navigates the factory floor and can interact with a number of stationary machines. For simplicity, we ignore the orientation of the robot, and describe the configuration space as a 2-dimensional map (the floorplan) i.e., $m \in \mathbb{R}^2$. The activity set represents the possible interactions of the robot with each machine (e.g., picking up an item from a machine). The function $\omega_a : \mathbb{R}^2 \to \{0, 1\}$ of each machine is defined by a prescribed template that defines the probability of successfully completing the activity in a given configuration, taking into account adjacent obstacles (e.g., walls) and the extent of the robot's arms. Figure 1a shows an example setting with 12 stationary machines. In this setting, each activity is deterministically mapped to configurations from which it can be achieved. The areas from which it is possible to pickup objects from a machine are depicted by rings around each machine (Figure 1b). The areas in pink are those from which more than one activity can be performed. The performance preferences can prioritize sampling from these areas. This can yield shorter plans in settings in which more than one object needs to be collected from a single machine or settings in which it is possible for a robot to reach more than one machine from a single waypoint without the need to move. The performance preferences can also be used to decrease the probability of sampling waypoints at doorways or to guarantee no waypoint is sampled near first aid stations. Previous approaches that consider each activity separately do not account for such task level considerations when selecting the waypoints that are sent to the task planner.

Sampling Procedure

TAWS separates between the connectivity analysis of a domain, which provides estimations of navigation costs within the physical space, and the task planning process, which finds a sequence of activities that accomplish the assigned task. First, it generates a *Dense PRM* (DPRM) over the configuration space. The process starts from the robot's initial position. The PRM is constructed by iteratively selecting a waypoint for expansion from the existing PRM. A set of new waypoints is cast from the chosen waypoint. Waypoints that are not in collision with any obstacle in the map (and that are traversable by the robot), are added to the graph. The coordinates of each node and the length of each straight edge are stored so that they can be used by the task planning process to estimate the cost of traveling between the waypoints. The accuracy of the estimations is correlated with the resolution of the DPRM. A DPRM for the factory domain in Example 1 is shown in Figure 1c.

After completing the generation of the DPRM, the iterative task planning stage begins. At each iteration, a number of waypoints are sampled from the DPRM and sent to the task planner to find a sequence of reachable activities that accomplishes the task.

TAWS is an anytime approach; even if it finds a solution, it will continue to search for better solutions until it is halted. If a plan is found, it is recorded, and the number of waypoints is incremented in order to find a more efficient solution. If the planner is unable to solve the problem within a time bound, the number of waypoints is decremented. If the planner claims that the problem is unsolvable, the number is incremented. This process is repeated iteratively until timeout is reached.

The selection of waypoints at each iteration is done according to the following procedure:

1. A sampling probability is assigned to each waypoint in the DPRM, using a task specific score which is induced by the activities and the performance preferences $f \in F$ and discussed in detail in the next section.
2. A waypoint is sampled from the DPRM and added to the task plan's model. The distance between the new waypoint and all existing waypoints is calculated by finding the shortest path through the edges of the DPRM. This value is added to the planning model as an estimate of the path's cost. In addition, the planing model is updated with information about all the activities that can be performed from the new waypoint.
3. The score function is updated to reduce the probability of sampling more waypoints near the one sampled. The radius for reducing the probability near a sampled waypoint is defined by the user.

After selecting the required number of waypoints, the resulting model is sent to a task planner that is chosen by the user.

Note that as opposed to previous approaches to waypoint sampling that were mentioned in the previous section, TAWS iteratively selects waypoints from the DPRM, and not from the underlying map. This means that even when the number of waypoints that are sent to the planner decreases, the quality of the estimations of the costs of navigating between the waypoints is not compromised. This is due to the fact that these estimations are evaluated according to the DPRM, which contains connectivity information about the entire space, regardless of the number of waypoints in the model that is sent to the task planner.

Combined Score

In order to account for both the probability of successfully accomplishing an activity from a given configuration as well as the performance preferences, TAWS uses a single *combined score* (CS) that associates a score to each waypoint (corresponding to a sampled configuration) in the DPRM. This score, that can be defined arbitrarily to account for different settings, is normalized over the waypoints in the DPRM, and is used to specify the probability of sampling each waypoint for inclusion in the task planning problem.

Specifically, the CS we suggest in Equation 1 can be used for the type of settings we consider here. The score uses a weighted sum over the different activities, while considering preferences multiplicatively.

$$CS(m) = \prod_{f \in F} f(m) \sum_{a \in A} \omega_a(m) \quad (1)$$

The above scoring approach increases the score (and corresponding sampling probability) of waypoint m from which multiple actions can be achieved by summing the probabilities $\omega_a(m)$ of successfully completing each activity from m. The application-specific performance preferences can be used to account for anything from breaking ties between otherwise equally probable waypoints, to imposing hard constraints that prevent sampling in certain regions. The former case can be achieved by setting $f(m)$ to vary between $1-\epsilon$ and 1 for some small value ϵ, so that the score of a waypoint is scaled down by up to $1-\epsilon$ in areas where it is preferred not to sample a waypoint. This may be relevant, for example, in settings where it is preferred that a noisy robot avoids getting close to a station of a human worker. In the latter case, hard constraints such as for ensuring that a robot never blocks access to a first-aid station, can be enforced by setting $f(m)$ to 0 in the critical area. This ensures that no waypoint can be sampled in that area, as its combined score and therefore sampling probability will be 0.

In Figure 2a, critical areas represent doorways and first aid stations. The combined score is assigned according to CS, the cost function in Equation 1, that considers both the performance preferences and the activity information (Figure 2b). The score is normalized and used to set the probability of sampling each waypoint from the DPRM. Once a waypoint is sampled, the probability of sampling nearby waypoints is reduced. The sampled set is used to search for a plan for the task (Figure 2c).

Evaluation

Our empirical evaluation was designed to answer two questions: (1) what is the benefit of using a single Dense PRM (DPRM) from which waypoints are iteratively sampled, and (2) what is the benefit of using TAWS to select waypoints in accounting for performance preferences.

To address these questions we use a dataset that consists of automated factory scenarios from the RoboCup Logistics League (RCLL) (Niemueller, Lakemeyer, and Ferrein 2015). An example from this domain is demonstrated in Example 1. In these scenarios the task of a robot is to complete an order by moving between machines and benches to pick up and place work-pieces. The work-pieces need to be combined and processed at different machines to produce a complete order that can be delivered at a delivery window. The problem description is temporal, such that each activity and navigation action has an estimated duration. In our setup, the objective is to find a plan that minimizes the time it takes to complete an order. Accordingly, we assess the quality of a plan by the total estimated duration of the individual activities and navigations between the waypoints from which the activities are performed. [1]

We varied the number of machines from 1 to 40 with $1-10$ work-pieces per order. Each machine could have several work-pieces, and the same work-piece could be found at different machines. For each machine count, we generated 10 different problems, varying the types and positions of objects, for a total of 400 problems[2].

Assessing The Benefit of Using a DPRM

To assess the benefits of using a DPRM, we compare TAWS, that produces a single DPRM as a preprocessing step and uses it throughout an iterative planning process, against FWPG and PPRM.

We use the FWPG implementation from (Edelkamp et al. 2018), in which each activity is associated with a template which prescribes the area from which it can be performed. In this approach, a single PRM with a fixed number of waypoints is created by randomly sampling a waypoint for *every* possible activity according to its template. This one-shot approach sends the generated PRM to the task planner. Since all activities are represented in the PRM, it is guaranteed to contain a solution to the problem. However, if the planner fails, for example because the size of the PRM exceeds its capacity, the process ends with a failure. For the PPRM approach we construct a (sparse) PRM at every iteration, changing the size of the sampled set of waypoints as needed. Similarly to TAWS, the PPRM approach is an anytime approach, that continues to improve its solution until it is halted.

In this part of our evaluation, no performance preferences were considered beyond the task description, so we are only evaluating the benefit of decoupling the connectiv-

[1] Videos of our simulated scenario are available at https://vimeo.com/user129497320

[2] The source code and experimental setup including all problem files can be found at https://github.com/sarah-keren/ROB-IS

	Solved Instances			Mean Time to Solution (s)			Mean Plan Duration (s)		
Elapsed	PPRM	FWPG	TAWS	PPRM	FWPG	TAWS	PPRM	FWPG	TAWS
5 secs	**6**	0	0	-	-	-	-	-	-
10 secs	16	**20**	14	**3.88 (0.00)**	9.47 (0.36)	9.07 (0.30)	5.80 (0.80)	**5.00 (0.00)**	**5.00 (0.00)**
1 min	27	**103**	65	17.41 (14.66)	**11.14 (1.11)**	18.95 (7.48)	7.12 (1.27)	12.99 (4.12)	**5.00 (0.00)**
2 min	40	**113**	103	36.40 (31.64)	**11.44 (1.24)**	31.22 (20.60)	8.48 (2.31)	14.19 (3.62)	**5.33 (0.64)**
4 min	47	113	**159**	58.76 (45.79)	**11.72 (1.50)**	48.58 (44.78)	10.01 (3.54)	14.03 (3.79)	**6.21 (2.10)**
8 min	56	113	**261**	110.68 (99.93)	**11.60 (1.43)**	54.09 (51.10)	10.38 (3.42)	13.69 (3.88)	**6.17 (2.05)**
10 min	63	**338**	300	174.31 (174.12)	**11.39 (1.42)**	53.55 (52.43)	10.63 (3.37)	13.35 (4.55)	**6.25 (2.15)**

Table 1: Performance per approach; standard deviation in brackets.

	Constraints			Constraints and Preferences		
	PPRM	FWPG	TAWS	PPRM	FWPG	TAWS
Instances solved	63	**337**	304	61	**335**	311
Time to first solution	276.38 (284.23)	57.13 (52.31)	**15.96 (4.80)**	215.84 (209.21)	59.66 (56.71)	**16.52** (5.00)
First quality - duration	10.23 (2.76)	**8.94 (4.51)**	10.53 (4.46)	**10.21** (2.70)	10.46 (4.96)	10.76 (4.21)
First quality - preferences	-	-	-	9.80 (3.09)	16.60 (9.56)	**0.63** (1.08)
Time to best solution	276.38 (284.23)	**57.13 (52.31)**	61.00 (61.68)	215.84 (209.21)	59.66 (56.71)	**42.74** (33.65)
Best quality - duration	10.23 (2.76)	8.94 (4.51)	**6.05 (1.86)**	10.21 (2.70)	10.46 (4.96)	**5.61** (1.11)
Best quality - preferences	-	-	-	9.80 (3.09)	16.60 (9.56)	**0.08** (0.16)

Table 2: Accounting for hard constraints (left), Accounting for constraints and preferences (right).

ity analysis from the task planning process by the one time generation and reuse of the DPRM.

We embedded all three approaches in ROS using the ROSPlan framework (Cashmore et al. 2015) and used the POPF temporal planner for task planning (Coles et al. 2010). For each problem, each approach was given a total time bound of 10 minutes to compute a solution. Since FWPG involves a single call to the planner while PPRM and TAWS are anytime approaches that iteratively call the planner, we experimented with different planning time allocations. Due to space considerations we only report here the results achieved for the setting in which each call to the planner by the PPRM and TAWS within the 10 minute time bound was limited to 10 seconds, while FWPG had 10 minutes of planning time [3]. The initial sample set size for TAWS and PPRM was 1 and the sampling step size was 4. For FWPG the sample set size is fixed by definition.

To compare the performance of the approaches we measured (1) the number of instances that were solved by each approach, (2) the quality of the first and best solutions found within seven time increments (5s, 10s, 1m, 2m, 4m, 8m and 10m), and (3) the amount of time it took to compute the first and best solutions (for TAWS preprocessing times are included in the results). The quality of a plan was measured according to the total duration of the plan (shorter is better).

For each approach, Table 1 shows the number of instances solved within each time increment. For problems solved by all approaches, the table shows the mean time to solution and mean plan duration in seconds. The standard deviation is indicated in brackets. The results show that FWPG solves the largest number of problems within the 10 minute bound, and that the most successful approach changes within the different time intervals. In terms of computation time, FWPG outperforms the other two approaches on the instances solved by all approaches. The notable achievement of TAWS is in terms of plan quality (plan duration). For all time intervals, TAWS finds shorter solutions, with up to 40% reduction.

Since PPRM only solved a limited number of instances, the results in Table 1 only reflect the performance of the approaches on the smallest instances. We therefore compared the performance of TAWS against FWPG on instances solved by these approaches. In Figure 3 we compare plan duration, and in Figure 4 we compare the time to solution in seconds for an increasing problem size for FWPG, and for the first and best solutions of TAWS. In Figure 3 instances below the line are those for which TAWS achieved a better result; TAWS achieved better plan quality over most instances, with an average 88% plan duration compared to FWPG across all problems solved by both approaches.

We have seen in Table 1 that for the small instances solved by all three approaches, FWPG outperforms the other two approaches in terms of the mean computation time. By investigating problems according to their size (corresponding to the number of activities that need to be performed to achieve a task), Figure 4 reveals that FWPG's computation time tends to be fixed around either 600 or 5 seconds for any problem size. In contrast, TAWS's time to best solution increases with problem size, and is much lower than FWPG for the majority of instances. On average, TAWS reaches the first solution in 17% and the best solution in 62% of the time taken by FWPG solution.

These results can be explained by recalling that FWPG generates a single and potentially unnecessarily large

[3] The additional results can be found in the online appendix at *https://github.com/sarah-keren/ROB-IS*.

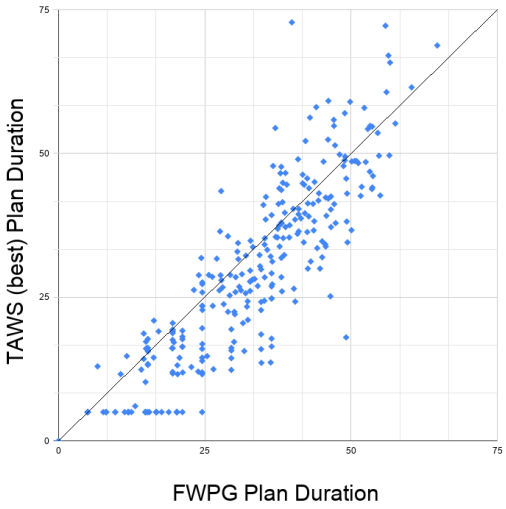


Figure 3: Comparison of best plan quality (plan duration in seconds) for problems solved by both TAWS (vertical axis) and FWPG (horizontal axis). Points below the line indicate higher quality for TAWS.

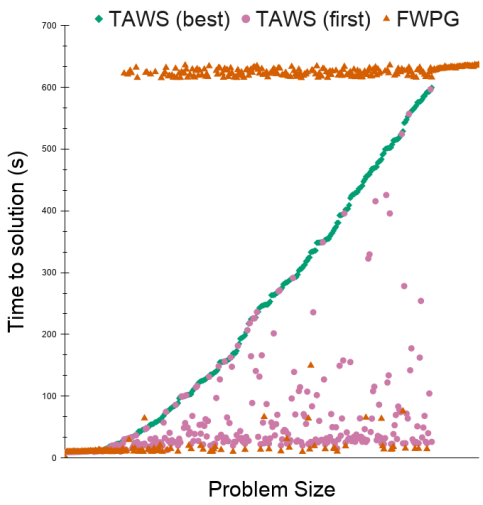


Figure 4: Comparison of time to solution for FWPG and time to first and best solutions for TAWS, on problems solved by both approaches, for increasing problem sizes.

problem description that is sent to the task planner, and which includes a separate waypoint for each activity that can be performed, including many activities that are irrelevant to achieving the task. In contrast, TAWS is an anytime approach, that iteratively generates and sends to the planner representations of the problem of varying size. In particular, it can take advantage of the fact that more than one activity can be performed from a single waypoint. Most notably, TAWS avoids the need to compute path costs at each iteration, and instead performs a one-time computation of the connectivity information, which it uses throughout the iterative search for a solution (and its iterative improvement). Our results show that despite the fact that TAWS is only allowed 10 seconds of planner time per iteration, its anytime approach is still able to find solutions more quickly than FWPG and those solutions are of higher quality.

Accounting for Performance Preferences

To examine the best way to account for performance preferences we introduce two kinds of preferences to the automated factory domain: *hard constraints*, and *soft preferences*. *Hard constraints* represent restrictions that cannot be violated, such as disallowing dwelling at locations (waypoints) near first aid stations, as exemplified in Figure 2a. As depicted in our score function in Equation 1, we account for *overlap information* by increasing the probability of sampling a waypoint according to the number of activities that can be performed from it. In the factory domain, this prioritizes waypoints from which more than one machine can be accessed and waypoints at machines from which more than one work-piece can be collected.

Since neither FWPG nor PPRM account for performance preferences, we extended both approaches to do so. For FWPG, in which waypoints are chosen randomly from within the area from which each activity can be performed, we used performance preferences to set the probability of sampling a specific position within each area. Performance preferences were used in two different stages of the PPRM generation. The first variant uses the performance preferences to influence which existing node in the PRM is chosen for expansion, while the alternative uses them during expansion to set the probability of sampling a new waypoint around the selected waypoint. As the results were similar for both variants we only report the results of the latter.

In addition to assessing plan quality according to its duration and keeping track of the times to solution, we used the following measure to evaluate a plan π according to its compliance with the performance preferences specified by the user.

$$P(\pi) = \sum_{m \in M_\pi} \left(dur(m) \sum_{f \in F} (1 - f(m)) \right) \quad (2)$$

where M_π are the set of waypoints visited in the plan, $dur(m)$ is the total time spent at waypoint m, and $f(m) \in \{0, 1\}$ is the normalized performance preferences score of that waypoint's position. This measure penalizes plans according to the time spent in undesirable locations.

Note that the performance preferences score is accounted for by the waypoint selection process and not directly modelled in the resulting task planning domain. This allows our approach to be used with any task planner, including temporal, probabilistic, or contingent planners. Specifically, the planner used in our experiments optimises plan duration.

Table 2 (left) shows the results achieved for instances for which only hard constraints were specified. For each approach, the table shows the number of instances solved within the time bound. For problems solved by all three approaches, we show the mean time to the first and best solution and the quality of the solutions in terms of duration. Table 2 (right) shows the results for which both hard con-

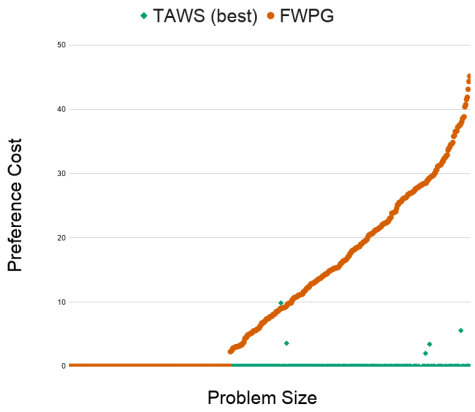


Figure 5: Comparison of preference cost for FWPG and TAWS, for increasing problem sizes.

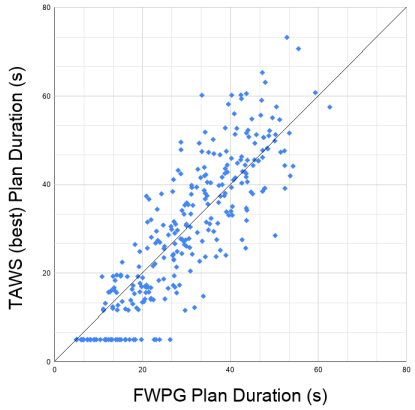


Figure 6: Comparison of plan duration for FWPG and TAWS in problems with soft preferences. Plan quality in terms of plan duration is comparable.

straints and soft preferences were specified. In addition to measuring plan duration, we include the mean performance score according to Equation 2 (the breakdown according to time intervals is included in the appendix (which can be found at https://github.com/sarah-keren/ROB-IS).

The results show that adding performance preferences does not have a substantial effect on the number of problems solved by each approach. For instances with only hard constraints, the mean quality of the first solution is best for FWPG (which finds only a single solution), but TAWS reaches the first solution much faster. Moreover, TAWS achieves the best plan quality (lowest duration) within the time bound. As shown in Table 2 (right), when soft constraints are added TAWS achieves both the lowest plan duration and the best preference score according to Equation 2.

Again, since PPRM solves only a limited number of instances, our analysis in Table 2 only accounts for smaller instances. In Figures 5 and 6 we therefore exclude PPRM, and compare solution quality for the instances solved by FWPG and TAWS.

The results in Figure 5 show that FWPG is limited in its ability to account for the performance preferences when compared to TAWS, which manages to achieve a penalty of 0 (according to Equation 2) for most instances. In Figure 6 we see that the better score in terms of performance preferences achieved by TAWS does not compromise the plans' quality in terms of plan duration, and that the plan duration of the solutions achieved by both approaches is similar.

The superior performance of TAWS compared to FWPG is due to the fact that TAWS samples a smaller set of waypoints that are more likely to respect the performance preferences. In contrast, since FWPG samples a fixed number of waypoints regardless of the problem size, it may result in many redundant waypoints that not only yield a more complex problem, but that also increase the probability that the resulting plan will make use of undesirable areas (or reduce the probability of using desirable areas).

Conclusion

We presented Task-Aware Waypoint Sampling (TAWS) as a new approach to selecting waypoints for task planning. TAWS's novelty is in the way it decouples the connectivity analysis of a domain from the task planning process, and its ability to account for user defined performance preferences.

The connectivity information is captured through a Dense PRM (DPRM), which is generated once and reused through the anytime iterative planning process to provide high quality estimations of navigation costs. The task planning problem at each iteration is constructed by sampling waypoints from the DPRM according to probabilities that are defined by both the activities that are part of the domain description and the performance preferences.

Our empirical evaluation on a set of automated factory problems shows that TAWS finds solutions that maximize compliance with the specified preferences, without compromising computation time and plan duration.

In the future, we intend to evaluate TAWS in other settings beyond the factory use-case. Specifically, we intend to investigate high-dimensional exploration scenarios, in which the sampling probability of TAWS can be used to specify areas in which a more meticulous search is desired. In addition, evaluation in this work focused on comparing the estimated plan duration of each approach. As a next step we intend to include in our evaluation the plan execution, comparing the execution time of the plans generated by each approach. Finally, in this work desirable behaviors were induced by changing the way in which the model for task planning was generated. We plan to investigate how TAWS can be adapted to settings in which robots are treated as "black boxes" and their inner implementation cannot be modified. In such settings, their behavior can instead be influenced by changing the information that is provided to them, for example by modifying the map that is used for navigation.

Acknowledgements

Sarah acknowledges the support of the Center for Research on Computation and Society (CRCS) at Harvard University.

References

Cambon, S.; Alami, R.; and Gravot, F. 2009. A Hybrid Approach to Intricate Motion, Manipulation and Task Planning. *The International Journal of Robotics Research* 28 (1): 104–126.

Canal, G.; Pignat, E.; Alenyà, G.; Calinon, S.; and Torras, C. 2018. Joining high-level symbolic planning with low-level motion primitives in adaptive HRI: application to dressing assistance. In *IEEE International Conference on Robotics and Automation (ICRA)*, 3273–3278.

Cashmore, M.; Fox, M.; Larkworthy, T.; Long, D.; and Magazzeni, D. 2014. AUV mission control via temporal planning. In *Proceedings IEEE International Conference on Robotics and Automation*, 6535–6541.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtós, N.; and Carreras, M. 2015. Rosplan: Planning in the robot operating system. In *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, 333–341.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proceedings International Conference on Automated Planning and Scheduling (ICAPS)*, 42–49.

Dornhege, C.; Hertle, A.; and Nebel, B. 2013. Lazy evaluation and subsumption caching for search-based integrated task and motion planning. In *IROS workshop on AI-based robotics*.

Edelkamp, S.; Lahijanian, M.; Magazzeni, D.; and Plaku, E. 2018. Integrating Temporal Reasoning and Sampling-Based Motion Planning for Multigoal Problems With Dynamics and Time Windows. *IEEE Robotics and Automation Letters* 3(4): 3473–3480.

Fernández-González, E.; Karpas, E.; and Williams, B. C. 2017. Mixed Discrete-Continuous Planning with Convex Optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAI)*, 4574–4580.

Garrett, C. R.; Chitnis, R.; Holladay, R.; Kim, B.; Silver, T.; Kaelbling, L. P.; and Lozano-Pérez, T. 2020. Integrated task and motion planning. *arXiv preprint arXiv:2010.01083* .

Gravot, F.; Cambon, S.; and Alami, R. 2005. aSyMov: A Planner That Deals with Intricate Symbolic and Geometric Problems. *Robotics Research. The Eleventh International Symposium. Springer Tracts in Advanced Robotics* 15: 100–110.

Kaelbling, L. P.; and Lozano-Pérez, T. 2011. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, 1470–1477.

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

McMahon, J.; and Plaku, E. 2014. Sampling-based tree search with discrete abstractions for motion planning with dynamics and temporal logic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3726–3733.

Niemueller, T.; Lakemeyer, G.; and Ferrein, A. 2015. The RoboCup logistics league as a benchmark for planning in robotics. *Planning and Robotics (PlanRob-15)* 63.

Plaku, E.; and Hager, G. D. 2010. Sampling-Based Motion and Symbolic Action Planning with geometric and differential constraints. In *IEEE International Conference on Robotics and Automation*, 5002–5008.

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S.; and Abbeel, P. 2014. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 639–646.

Toussaint, M. 2015. Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1930–1936.