Conflict-Based Multi-Robot Multi-Goal Task and Motion Planning

Junho Lee and Derek Long

Department of Informatics, King's College London, UK
{*junho.lee, derek.long*}*@kcl.ac.uk*

Abstract

Task and Motion Planning (TAMP) has been a subject of extensive research in goal-directed robotic control systems. The main challenge is to determine the relationship between the symbolic representation that supports goal-oriented task planning and the continuous configuration space representation that is used by a motion planner. In this paper, we explore a complex version of this problem, in which multiple robots operate within the same work area, performing multiple tasks as they move between work locations.

Introduction

Task and Motion Planning (TAMP) is an essential issue in robotics and AI Plnaning. The challenge it raises is that tasks requiring a robotic system to perform motion must respect the constraints on that motion, but motion paths are expensive and complex to compute, so it is important to generate motion paths relevant to solving the task planning problem. This creates a chicken-and-egg problem, in which the relevant motion plans require knowledge of the task plan to identify, while the task plan cannot be constructed without knowing which motion plans are possible. A top-down approach is to construct a task plan that simply assumes motion plans can be constructed, and then relies on a motion planner to complete the required paths to make the task plan executable. This decoupling approach can cause the task planner to generate an inefficient plan or provide an impossible plan to realise, since the high-level planner has no access to the constraints that inform the motion planner, such as geometry constraints. Therefore, most TAMP research has focused on how to abstract continuous information at a high level.

The multi-robot multi-goal problem, in which multiple robots efficiently complete their mission without collision, is more challenging for TAMP studies. The robots' trajectories and the collision regions change depending on the order of tasks that the task planner determines. Therefore, if the task planner does not have enough information about the environment, it must compare all combinations of paths to obtain a plan with conflict-free paths, which is too expensive.

In this paper, we propose a new TAMP framework, Conflict-Based Task and Motion planning (CBTAMP), that can solve the multi-robot multi-goal problem. First, CBTAMP covers the space using the roadmap of the sampling-based motion planning method and provides a basis for representing continuous space in the discrete domain in a waypoint-based manner. After that, the problem is solved through continual interaction between the PDDL-based task planner and motion planner. Our framework allows any PDDL temporal task planner that can use durative actions (Fox and Long 2003), increasing versatility and compatibility with motion planners. The motion planner and collision detector determine the collision area of a given plan and provide it to the task planner for better reasoning.

Another contribution of our paper is to confirm the feasibility and utility of the framework by presenting scenarios. Each scenario presents a narrow space where many collisions are expected, showing whether CBTAMP efficiently solves problems in a challenging space. The results are compared to our predicted output and discussed, along with their scalability.

Motivating Scenario

We begin by describing a family of problems that illustrate one of the ways in which task and motion planning can interact, motivating the need to consider the interactions in solving problems that feature mobile robotic systems performing tasks in shared space.

We consider problems in which there is a collection of workstations at which various tasks must be completed. These tasks are, for the purposes of the following work, just two: welding and bolting some machined parts. In this simple setting, the tasks are treated as independent, but cannot be performed simultaneously. Each problem features a collection of mobile robots, each capable of performing both of the tasks, but with differing competency, represented by the tasks taking a different duration according to which robot is performing the task. The tasks can only be performed by a robot sufficiently close to the workstation, so robots must move to position themselves at a workstation in order to carry out one of the tasks. Obstacles in the work area can make navigation between workstations more or less difficult. The goal is to complete all the tasks in the shortest possible time. The family of scenarios supports differently structured solutions depending on the relative durations of travel between the workstations and the execution of the tasks at the

workstations. For example, if the robots are slow at moving, but fast at completing the tasks, then the solution will assign the closest robot to each workstation and have each one complete all tasks at their assigned workstation. Conversely, if there are specialist robots that can complete their tasks very much faster than non-specialists, then it will become more efficient for robots to move between workstations. The problem that we are most interested in, is the management of constraints in the navigable space that mean that there is a risk that robots can block each other, leading to significantly increased travel times and, therefore, a critical decision between moving robots to different workstations and the delays possible in robots avoiding one another.

In this paper, we consider six different weld-bolt scenarios. Robots are specialized with the action of their type, but the other action can be carried out by taking more time. We structure the work area to include a passage between the parts containing the different workstations. We vary the number of robots, the passage length, and action cost change. We limit the mobile robot to a holonomic system and ignore geometric and differential constraints, such as velocity, for computational convenience and simplicity.
There are obstacles near the passage entry in each room, and a bolt robot and a welding robot start at the lower side of each room respectively. At the upper side of each room are located the workstations. The sixth scenario shows a scaled-up scenario that increases the number of robots and targets. Figure 1 shows an example of a simple scenario.

Scenario I In the first scenario, the passage connecting the rooms has a short length. This passage can be traversed by only one robot at once, and the cost of unskilled work of robots is not more expensive than passing through the passage. An efficient global plan without collisions is for each robot to perform both actions in the same room without crossing the room.

Scenario II The environment of the second scenario is the same as the first scenario but with a significantly high non-specialist work cost. It is therefore better for the robots to traverse the passage several times rather than perform unskilled work. When the bolt robot is working on a nearby task, the welding robot crosses the passage and navigates through the room to avoid collisions ahead of time. After completing the bolt, the bolting robot moves to another target through the passage, and the welding robot also completes welding and crosses the room again to complete the remaining task.

Scenario III The third scenario has high-cost non-specialist action and a long narrow passage. In this case, it is useful to cross between the rooms, but at most once. It is also important to understand the need to sequence access to the passage, since the robots cannot pass each other in it. This problem highlights the fact that planning the movement of each robot while ignoring the other robot will lead to deadlock over access to the passage. On the other hand, it is impractical to preplan all possible paths for the robots, including cases in which both robots need to access the passage. This shows the need to communicate between task planning and motion planning, to determine the candidate paths of interest to the task planner, but to reflect the timing constraints that this implies for the motion planner to analyse the risk of collision.

Scenario IV Scenario IV has a high non-specialist action cost and a passage wide enough for two robots to pass through simultaneously. Therefore, the robots will complete their work in each room and move across passages to other targets at the same time without colliding. This example contrasts with the last, because it is relatively cheap for both robots to simultaneously traverse the work area simultaneously (although the motion planning problem must still ensure that the robots follow non-colliding paths).

Scenario V In the fifth scenario, two narrow passages bridge the two rooms. Hence, if one robot occupies one passage first, the other robot will be forced to use the other passage if traversing in the same interval.

Scenario VI In the last scenario, there are two welding robots and two bolting robots in one room, and there are two weld-bolt tasks in each room, so a total of four targets are located on the map. This scenario was constructed to check the scalability of the number of robots, and more collisions between robots are expected. At least two or more robots will pass through the passage to complete the mission.

Related Work

For a robot to complete its mission, it must find an action sequence and a trajectory that satisfies the geometric constraints that makes each action executable. Several approaches have been considered in combining Task and Motion Planning (Dornhege et al. 2009; Cambon, Alami, and Gravot 2009; Kaelbling and Lozano-Perez 2010; Kim et al. 2019; Thomason and Knepper 2019), and they have mainly solved the problem by expressing low-level geometric constraints as abstractions at a higher-level. For example, PDDLStream (Garrett, Lozano-Pérez, and Kaelbling 2020), an extension of the existing PDDL, uses stream instances to bridge between the discrete task constraints and continuous motion constraints. A stream instance allows the planner to reason by sampling the constraint and cost from the motion planner.

Scenarios in the real world are not limited to manipulation problems, but include navigation problems such as visiting multiple locations. Therefore, it is necessary to represent and reason about collision-free trajectories generated by a motion planner and then represented in the discrete domain. In UP2TA (Muñoz, R-Moreno, and Barrero 2016), the task planner creates an optimal sequence action plan iteratively, using a heuristic that combines the FF heuristic and Euclidian distances. The authors of PETLON (Lo, Zhang, and Stone 2018) devise searches for the task-level optimal plan for mobile robot navigation more efficiently through the bounds of sample values. In (Thomas, Mastrogiovanni, and Baglietto 2019), the authors introduce an approach that uses a Kalman filter to tackle robot TAMP in a stochastic environment. However, most TAMP approaches for navigation are focused on a single robot, and they cannot take into account

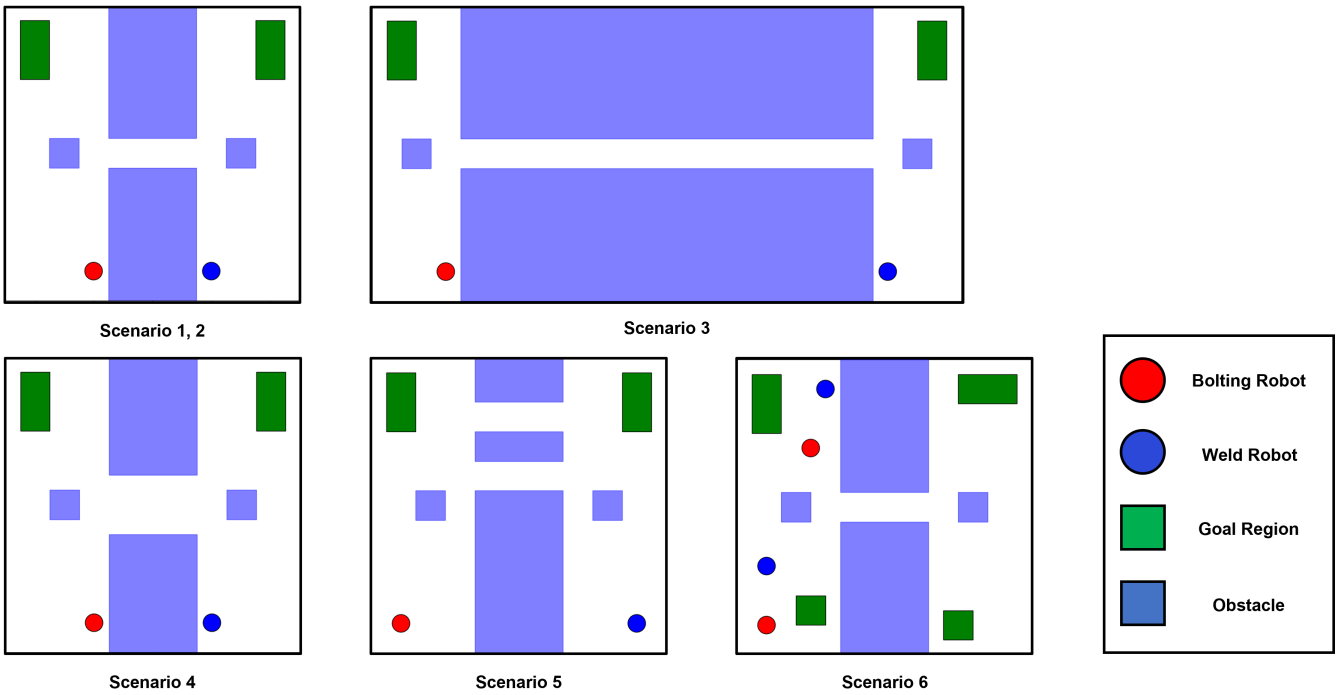

Figure 1: An example scenario for the CBTAMP Framework. The objects shown are: bolting robot (red circle), welding robot (blue circle), work stations (green rectangles), and obstacles (blue rectangular regions). A robot can work when at a workstation and each goal requires two tasks: weld and bolt. The robots' mission is to complete the tasks at all the workstations in the map. Short clips of scenario solution by CBTAMP can be found at https://youtube.com/playlist?list=PLxww1CRMAaMfeF92F0k_m2--CqroZZwPW.

potential collisions between robots arising from shared work spaces such as those presented in the previous section.

On the other hand, a sampling-based planning algorithm has been used in TAMP to account for continuous space and as an abstraction for task planning. A rapidly-exploring random tree (RRT) (LaValle and Kuffner Jr 2001) is presented in (Burfoot, Pineau, and Dudek 2006) with a STRIPS symbolic task planner. A probabilistic roadmap (PRM) (Kavraki et al. 1996) can reuse the generated roadmap for multiple different source-destination paths, so provides a better method than RRT for multiple queries. For example, the author of (Le and Plaku 2017) uses the probabilistic roadmap for extending a motion tree with geometric and kinematic constraints, and (Keren, Canal, and Cashmore 2021) presents a method to use waypoints sampled through a PRM to populate the discrete domain. This waypoint-based method increases the solution quality by sampling the waypoints required for the task plan in a specific area through a combined score function. However, it cannot account for the conflicts of multiple robots. The work differs from the work we present in that it is necessary to predefine specific areas of the environment (e.g., doorway) rather than to rely on discovering them by sampling.

We propose a novel approach that utilizes the advantages of sampling-based motion planning to describe the environment model and solves a multi-robot multi-goal problem. Influenced by CBS-MAPF (Sharon et al. 2015), this method iteratively finds an efficient collision-free action sequence in which the collisions between paths generated from the motion planner are updated in a waypoint-based discrete domain, and the task planner resolves the discrete choices implied by the need to avoid these collisions.

Conflict-Based Task and Motion Planning (CBTAMP)

In this section, we introduce a novel system that integrates a sampling-based motion planning method with a PDDL-based temporal task planner (Coles et al. 2010). The architecture of the framework can be seen in figure 2.

Problem Definition

A world $\mathcal{W}$ consists of obstacles $\mathcal{O} = \{\mathcal{O}_1, ..., \mathcal{O}_k\}$, goals $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_m\}$, and robots $\mathcal{M} = \{\mathcal{M}_1, ..., \mathcal{M}_n\}$, and each robot model is expressed as $\mathcal{M}_i = \{D_i, \mathcal{Q}_i, \mathcal{S}_i, q_i^{init}, s_i^{init}\}$ which records its radius $\mathcal{D}_i$, configuration space $\mathcal{Q}_i$, state space $\mathcal{S}_i$, initial configuration $q_i^{init} \in \mathcal{Q}_i$, and initial state $s_i^{init} \in \mathcal{S}_i$. The objective is to solve the planning problem $\Pi = (Q, S, M, A, S_I, G)$ where $Q \subseteq \mathbb{R}^n$ is the n-dimensional configuration space of the robots, S is the symbolic state space, M is a set of robots, A is a set of actions, $S_I \in S$ is the initial symbolic state, and $G \subseteq S$ defines the goal states. The solution of a multi-robot multi-goal problem Π is a plan π leading

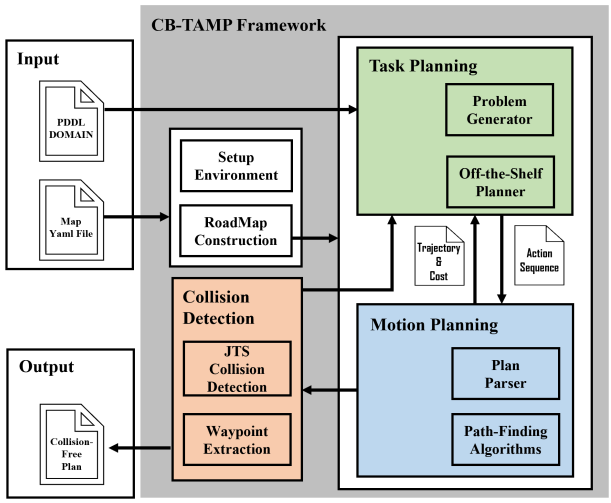

Figure 2: General Architecture of CBTAMP framework.

from the initial symbolic state S_I to a symbolic goal state $g \in G$, which is described by $S_I \xrightarrow{\pi} g$. The overall plan π can be projected onto separate plans of each robot π_i by restricting to actions that include M_i, and the plan of each robot π_i defines the action sequence it executes and is also associated with a trajectory $\zeta_i : [0, \infty) \rightarrow \mathcal{S}_i$, defining its position as a function of time. A valid solution ensures that there is no time, t, and pair of robots, M_i and M_j, such that $\zeta_i(t)$ and $\zeta_j(t)$ are within $max(D_i, D_j)$ (a collision).

In this framework, we place the following limitations to reduce computation and focus on our aim. Robot orientations are ignored, so that each robot has a 2-D Euclidean configuration space, $Q \subseteq \mathbb{R}^2$. Each robot's speed is constant, and its acceleration is neglected. Therefore, each robot can move or stop immediately, and the navigation time is proportional to the distance between locations. All robots have the same size and shape, $\mathcal{D}_i = d$ for all M_i, so one roadmap can be shared for all robots, using the same high-level representation for task planning aspects. The trajectory of actions other than navigation is assumed to be a fixed interval of constant location. Since this paper focuses on an efficient global plan in which a collision between robots or obstacles is avoided and only navigation action trajectories are computed.

In other multi-robot systems, one objective function that has been considered is the sum of costs of each robot's plan, where cost might be measured in fuel, risk or monetary terms, while an alternative is makespan, the time it takes all robots to complete a mission (Stern et al. 2019). This study prioritizes makespan for exploiting as many robots as possible and based on the characteristics of the task planner.

Constructing a Roadmap

The framework starts by instantiating a map file that describes the robot's state $\mathcal{M}$, goals $\mathcal{G}$, and obstacles $\mathcal{O}$ for the world $\mathcal{O}$ and building a roadmap $\mathcal{R} = (V, E)$ in the corresponding configuration space $\mathcal{Q}$. Note that all robots share the same configuration space and roadmap as they have the same shape and size. The purpose of constructing a roadmap is to provide not only a collision-free path network in continuous space but also a geometric abstraction of the environment to the task planner. Two well-known sampling-based methods can be considered. Since a PRM can process multiple queries efficiently, it is more suitable for a multi-robot system than RRTs and has the advantage of reusing the generated roadmap and samples. Also, certain samples in the roadmap can be utilized for geometric abstraction; therefore, it is suitable for our proposed framework. CBTAMP uses the simplest version of PRM in this paper for versatility. Several variants such as Lazy PRM and Obstacle-based PRM can be deployed (Nissoux, Siméon, and Laumond 1999; Bohlin and Kavraki 2000; Amato et al. 1998).

Algorithm 1: CBTAMP-PRM algorithm

Input: Configuration Space $\mathcal{Q}$
Parameter: n: number of samples to put in roadmap
k: number of closest neighbors
sd: samling distance
cd: casting distance
Output: A *roadmap* $\mathcal{R} = (V, E)$

1: $V \leftarrow \emptyset$
2: $E \leftarrow \emptyset$
3: **while** $|V| < n$ **do**
4: $q_{new} \leftarrow$ a bridge or random configuration in $\mathcal{Q}$
5: **if** $dist(q_{new}, q_i) > sd$ for all $q_i \in V_i$ **then**
6: $V \leftarrow V \cup \{q_{new}\}$
7: **for all** $q \in V$ **do**
8: $N_q \leftarrow$ the k closest neighbors of q chosen from V according to dist
9: **for all** $q' \in N_q$ **do**
10: **if** $(q, q') \notin E$ **and** $\Delta(q, q') \neq$ NIL **and** $dist(q, q') < cd$ **then**
11: $E \leftarrow E \cup \{(q, q')\}$
12: **return** $\mathcal{R} = (V, E)$

The generated roadmap $\mathcal{R}$ is used in Task Planning and Motion Planning, respectively. The algorithm can be found in Algorithm 1. A mixture of two methods, uniform sampling, and bridge-test sampling (Hsu et al. 2003), is used for CBTAMP-PRM sampling. Bridge test sampling increases the probability of finding a sample in a narrow region by checking between two obstacles (line 4). For well-distributed samples and a sophisticated roadmap, the sampling distance and casting distance parameters proportional to the robot's size are used (line 5, line 10). Before Task Planner uses the roadmap, the framework needs to check that the roadmap has enough samples to approach the goal or the robot's initial position. If there is no path to access $q_{\mathcal{G}_k} \in \mathcal{Q}$ from each goal $\mathcal{G}_k \in \mathcal{G}$ or $q_n^{init} \in \mathcal{Q}$ from each robot $q_n^{init} \in \mathcal{M}$, the algorithm is repeated again, increasing parameter n.

Listing 1: The `entry_to_col_navigate` action of the weld-bolt scenario

```
(:durative-action entry_to_col_navigate
    :parameters (?r - robot ?from - entry_way ?to -
        col_way ?e - entry)
    :duration (= ?duration (/ (distance ?from ?to) (
        velocity ?r)))
    :condition (and (at start (at ?r ?from))
                (at start (> (distance ?from ?to) 0))
                (at start (entry_located ?e ?from))
                (at start (> (entry_key ?e) 0))
                (at start (reserved ?to))
                (over all (vertex_free ?to))
                (over all (connected ?from ?to)))
    :effect (and (at start (not (at ?r ?from)))
            (at start (vertex_free ?from))
            (at end (not (vertex_free ?to)))
            (at start (not (reserved ?to)))
            (at end (reserved ?to))
            (at start (decrease (entry_key ?e) 1))
            (at end (at ?r ?to))))
```

Listing 2: The output plan of the `weld-bolt` scenario I

```
0.000: (normal_navigate robot0000 wp0 wp10) [8.910]
0.000: (normal_to_entry_navigate robot0001 wp1 wp14 e0)
    [4.150]
4.151: (entry_to_col_navigate robot0001 wp14 wp13 e0) [0.890]
5.042: (col_navigate robot0001 wp13 wp11) [1.770]
6.813: (col_navigate robot0001 wp11 wp12) [0.890]
7.704: (col_to_normal_navigate robot0001 wp12 wp7) [5.000]
8.911: (bolt robot0000 wp10 goal0001) [5.000]
13.911: (normal_navigate robot0000 wp10 wp6) [0.940]
13.912: (weld robot0001 wp7 goal0001) [5.000]
14.852: (normal_to_col_navigate robot0000 wp6 wp12) [4.840]
18.912: (normal_navigate robot0001 wp7 wp8) [1.730]
19.693: (col_navigate robot0000 wp12 wp11) [0.890]
20.584: (col_navigate robot0000 wp11 wp13) [1.770]
20.643: (normal_to_col_navigate robot0001 wp8 wp12) [4.330]
22.355: (col_to_entry_navigate robot0000 wp13 wp14 e0) [0.890]
23.246: (entry_to_normal_navigate robot0000 wp14 wp2 e0)
    [4.710]
24.974: (col_navigate robot0001 wp12 wp11) [0.890]
25.865: (col_navigate robot0001 wp11 wp13) [1.770]
27.636: (col_to_entry_navigate robot0001 wp13 wp15 e0) [0.980]
27.956: (bolt robot0000 wp2 goal0000) [5.000]
29.217: (entry_to_normal_navigate robot0001 wp15 wp3 e0)
    [3.740]
32.957: (weld robot0001 wp3 goal0000) [5.000]
```

Task Planning

The task planning phase begins with the generation of a PDDL problem instance, based on information from the roadmap $\mathcal{R}$ and collision waypoints from the collision detection module. Then, we use OPTIC (Benton, Coles, and Coles 2012) (a variant of POPF (Coles et al. 2010), that performs branch-and-bound search to optimise the solution within the same search space) to create a temporal plan. The temporal structure allows concurrency between actions of different robots and properly respects the timing constraints of accessibility to areas of the work space, in order to avoid collisions. The planner resolves conflicts based on collision information from the motion planner using a *semaphore* token in the domain model to restrict access to areas of the workspace. The waypoints at entry and exit around the restricted areas are where the token is taken or released. One important point is that robots entering a restricted area from the same direction and passing through it in the same direction will not cause a conflict (in general – if the robots travel at different speeds, conflicts might still arise).

The Problem Generator module creates robot, goal, waypoint, and entry instances with the generated roadmap and collision information. The instance contains types of robots or actions declared in the PDDL domain. (e.g., weld_robot in our scenario). Workstation access constraints prevent multiple robots from working on one target simultaneously and determine the goal operations required. This information is passed from the Map instance. Waypoints are divided into normal, collision, and entry waypoints. The normal waypoints include the robot's starting and target position, and other waypoints are collision-related waypoints created in the previous iteration. An entry instance has two entry waypoints and is mainly created near the entrance of the passage, and it not only manages other robots so that they can pass through the passage without collision but also becomes a means to connect normal waypoints and collision waypoints.

The well-constructed connection between waypoints is essential for navigation action in a given environment. For example, if there is a waypoint in the middle of a narrow passage connecting two rooms, the waypoints in both rooms must be connected via a passage waypoint. Therefore, the task planner calls the motion planner to calculate the connection and the distance. The detailed procedure is described in the motion planning section.

We tested two alternative variants of the domain, based on the release point of semaphore tokens for the constrained areas. The Late-release domain (releasing the token at the end of the durative navigation action) guarantees more stability than the less conservative Early-release domain, by delaying the release time of waypoints, resulting in a higher Makespan on average. Example scenario domains, weld and bolt, and output plans can be viewed partially in listing 1 and listing 2. Note that every action in this domain is durative: reasoning about temporal properties of actions and supporting true concurrency is critical in solving the problems we are considering. The generated plan is then passed to the motion planner.

Motion Planning

The motion planner is called within the framework to provide the information needed to generate a problem file and parses the action sequence from the OPTIC planner to construct the trajectories needed for navigation. The basic algorithm for the motion planner uses Dijkstra's algorithm from the generated roadmap. Each module has the following procedure.

For *Problem Generator:*

1. The motion planner, which obtains the waypoint instances from the task planning problem generator, splits the collision waypoints from the other waypoints. Colli-

sion waypoints are connected across restricted areas, between themselves and only allow access to the associated collision region through specific entry waypoints.

2. A collision roadmap is created to check whether collision waypoints in the same collision region are connected. For example, suppose that there are n collision waypoints in one collision region and m collision samples. In order to check the connection of the two collision waypoints way_0 and way_1, the remaining collision waypoints $way_2, way_3, ..., way_{n-1}$ and their surrounding samples (k) are excluded from the total sample m; then, a new roadmap with $m - k$ remaining samples is checked to ensure a path is available between the two waypoints. If there is another collision waypoint between the two waypoints then the path will not exist. If the planner finds a path, the connection and distance information is returned to the problem generator. This procedure is executed iteratively to verify the connection of all collision waypoints in the collision area.
3. A collision-free roadmap is created that excludes all collision samples, in order to find connections between normal waypoints. If the Dijkstra algorithm returns a path between normal waypoints, the connection and distance are returned to the problem generator.

The roadmaps and connections from the above procedure allow the task planner to generate sophisticated plans.

For *Parser and Navigation:*

1. The action sequence created from OPTIC is parsed and used together with the roadmap to create the trajectory of robots along with the dispatched time. For example, for the output plan shown in listing 2, $robot0000$ constructs the first two actions in the order: ($normal\_navigate$ $robot0000$ wp_0 wp_{10}, 0.000) ($bolt$ $robot0000$ wp_{10} $goal0001$, 8.910).
2. The robot finds the configuration trajectory in the sequence of its actions using Dijkstra's algorithm. It is noteworthy that the path of the robot includes prediction time information. In the case of a $navigate$ action, the time can be predicted through the distance between configurations, and the cost or dispatch time of other actions can be used to predict how long the robot should wait at a location. The generated path is passed to the collision detection module for collision checks.

Collision Detection and Update

The Collision Detection module provides the necessary abstraction of the work area required for the discrete model, in a waypoint-based format, through collision information between robots in a continuous environment. Collision regions $\mathcal{C}$ are divided into two region types: narrow collision and non-narrow collision. For narrow collision areas, collisions can occur because of interactions with obstacles. Since the robot's approach to the collision region has to be dealt with more precisely in the dense area of obstacles, it creates an entry point for a traffic circle. In the case of non-narrow collision areas, the source of risk is multiple robots entering the same open area. The computation can be reduced by simplifying the procedure. It detects the collision of the trajectory of two or more robots, extracts the collision region, and selects collision waypoints that serve as the *entrance* of the region from among the samples in the region. An entry point, which is a location where the robot can wait, is created for each collision waypoint depending on the type of region. In order to reduce computational effort, collision waypoints in the same region are reused from previous iterations. The detailed pesudocode is shown in Alg. 2

A collision detector checks collisions on the trajectories of all robots (line 2). In this implementation, we use the JTS package as a collision detector for efficient implementation. The collision region which contains multiple samples is returned. The type of each collision is then determined (line 3). Collisions are classified into narrow and non-narrow collisions based on the distance between the samples in the collision region and the surrounding obstacles. One area is randomly selected to handle (line 4), with the narrow area taking precedence over non-narrow collisions. We next determine whether the selected region can be unified with the previous region (line 6). If there are overlapping samples, the selected region is merged with the previous region to create

Algorithm 2: CBTAMP-Collision Detection algorithm

Input: World $\mathcal{W}$, Roadmap $\mathcal{R}$, trajectory ζ
Previous Collision Regions $\mathcal{C}_{old}$
Output: New Collision Regions $\mathcal{C}_{new}$

1: **for** two combination $(\zeta_{\pi_1}, \zeta_{\pi_2}, ...) \in \zeta$ **do**
2: collision_samples ←COL_DETECT($\zeta_{\pi_i}, \zeta_{\pi_j}$)
3: **if** collision_samples is $\emptyset$ **then**
4: **return** None
5: collisions ←COL_CLASS(collision_samples, $\mathcal{W}$)
6: collision ←randomly pick one collision in collisions
7: $\mathcal{C}_{cand}$ ←COL_UNIFY($\mathcal{C}_{old}$, $collision$)
8: **if** $\mathcal{C}_{cand}$ is narrow collision **then**
9: $\mathcal{C}_{new}$ ←COL_WAY_EXTRACT($\mathcal{C}_{cand}$, $\mathcal{R}$)
10: $\mathcal{C}_{new}$ ←ENTRY_EXTRACT($\mathcal{C}_{new}$, $\mathcal{R}$)
11: **else if** $\mathcal{C}_{cand}$ is non-narrow collision **then**
12: $\mathcal{C}_{new}$ ←COL_WAY_SIMPLE_EXTRACT($\mathcal{C}_{cand}$, $\mathcal{R}$)
13: **return** $\mathcal{C}_{new}$

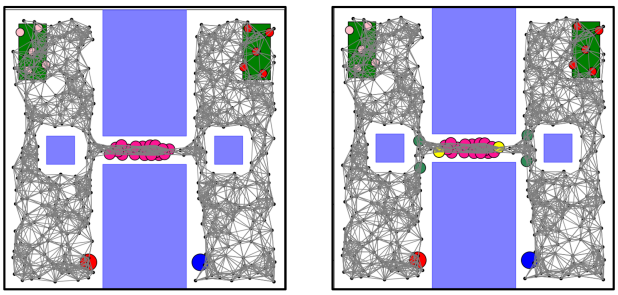

Figure 3: Predicted collision samples (pink) from collision detection module (left), generated collision waypoints (yellow) and entry waypoints (green) based on collision region (right).

Algorithm 3: CBTAMP-Overall Approach

Input: PDDL Domain **D**, Map **M**
Parameter: n: number of iteration
Output: Collision-Free Plan π

1: env ←ENV_INTSTANCE(**M**)
2: $\mathcal{R}$ ←CBTAMP_PRM(env)
3: $\mathcal{C} \leftarrow \emptyset$
4: $best\_plan \leftarrow \emptyset$, $min\_cost \leftarrow \infty$
5: **while** ¬ collision-free plan **or** iteration $< n$ **do**
6: **P** ←PROBLEM_GENERATOR($\mathcal{R}$,$\mathcal{C}$)
7: $plan$ ←TASK_PLAN(**D**, **P**)
8: ζ ←MOTION_PLAN($plan$)
9: $\mathcal{C}$ ←CBTAMP_COLLISION_DETECTION(env, $\mathcal{R}$, ζ, $\mathcal{C}$)
10: **if** $\mathcal{C}$ is None **and** $cost_\zeta < min\_cost$ **then**
11: $best\_plan$ ←($plan$, ζ), min_cost ←$cost_\zeta$
12: **return** $best\_plan$

a candidate region. If the candidate region is a narrow collision, the collision waypoints in the candidate region are sampled (line 7). N samples are selected randomly from the collision region. After generating the PRM excluding the existing collision waypoint in the same region, if there are paths between n-samples and any normal waypoints, the sample in the collision region closest to the normal waypoint becomes the collision waypoint. This procedure is repeated until there is no path to the normal waypoint. The previously collision waypoint is reused if the candidate region is merged. If the candidate region is a non-narrow collision, the two samples with the longest distance of the region become collision waypoints (line 10). Entry points for the collision waypoints are created (line 9) for narrow collision areas, chosen from samples of the roadmap excluding all collision samples. Up to two entry waypoints can be created if possible, but it is possible that an area might not have one. Entry waypoints can only be accessed from the other collision waypoints. Figure 3 shows an example of the collision regions found and the collision waypoints and entry waypoints created.

Iterating Framework

The overall framework repeats plan generation from the task planner, trajectory creation from the motion planner, and collision detection. The Task Planner uses both the early release domain and the late release domain, to improve search quality. When the created plan has no conflicts, the plan with the smallest makespan is saved as the best plan. The iteration terminates when both domains produce collision-free plan paths or more than a specific number of times. Algorithm 3 represents the CBTAMP overall approach when using one domain.

Evaluation

Various benchmarks for multi-robot pathfinding have been introduced in (Stern et al. 2019); however, these are mainly run in the discrete domain and do not address multi-goal problems. The purpose of the experiment is to confirm the change of the global plan of multi-robots according to the environment and parameters and allow collision-free trajectories. In that sense, the motivating scenario where robots have a mission to work on multiple goals in a narrow space with different action costs is optimal to account for our purpose. In order to increase the efficiency of computation, the planning time of the planner, OPTIC, is limited to 10 seconds, which means that more than 20 seconds are consumed in the task planning module since two domains are used per iteration. Experiments report the runtime, makespan, sum of cost, success rate, and scenario accuracy rate. The runtime includes the time from reading the map file to plan out, and the success rate is the result of finding a collision-free plan so that it is considered a failure if the planner cannot produce an output plan from the problem instance or find it within a predefined number of iterations. Experiments were carried out on an Intel Core i7 (3.00GHz) with single-threaded implementation.

Scenario Evaluation The six scenarios in figure 1 were tested through the framework, the robot is a circular model with a radius of 0.3, and the map size is the same as 10x10 except for the third scenario (20x10). The number of initial roadmap samples is set to 500 or 600 depending on the size, and the task planner uses only total time, that is, makespan, as an optimized metric. In the case of prediction accuracy, the result is derived by comparing the output plan from our framework and the desired plan noted in the motivating scenario section above. See *Motivating Scenario* for detailed predicted plans.

We performed 50 executions for each scenario and set an iteration limit of 15 times, and if it exceeds the limitation, it is considered a failure. The results are described in figure 4. A collision-free plan was successfully derived in all scenarios with high probability, and the desired plan was created with a prediction accuracy of over 70% in the environment except the second scenario. Note that the second scenario has a similar cost to the plan in which one robot crosses the passage first while the other robot is working, and the plan in which the robots start work first in each room and one robot waits while the other robot crosses. In the case of runtime, the average value was within 90 seconds in all environments except for the fourth and sixth scenarios where four robots exist. This means that the plan is generated within three iterations on average, and an environment with many robots requires multiple iterations with a high probability of collision. Figure 4 shows the scenarios where the passage is wide or has multiple passages have less makespan and sum of cost since robots can cross the passage simultaneously.

Scalability The framework's scalability with respect to the number of robots is considered. Experiments are carried out in the same environment as the first scenario, and the number of goals is set equal to the number of robots to increase the probability that all robots work from the initial time. In order to increase the congestion that makes the problem challenging, all the robots are started in one room. Goal locations are equally divided in each room. At least two robots will use the narrow passage with this setup, and collisions are likely to

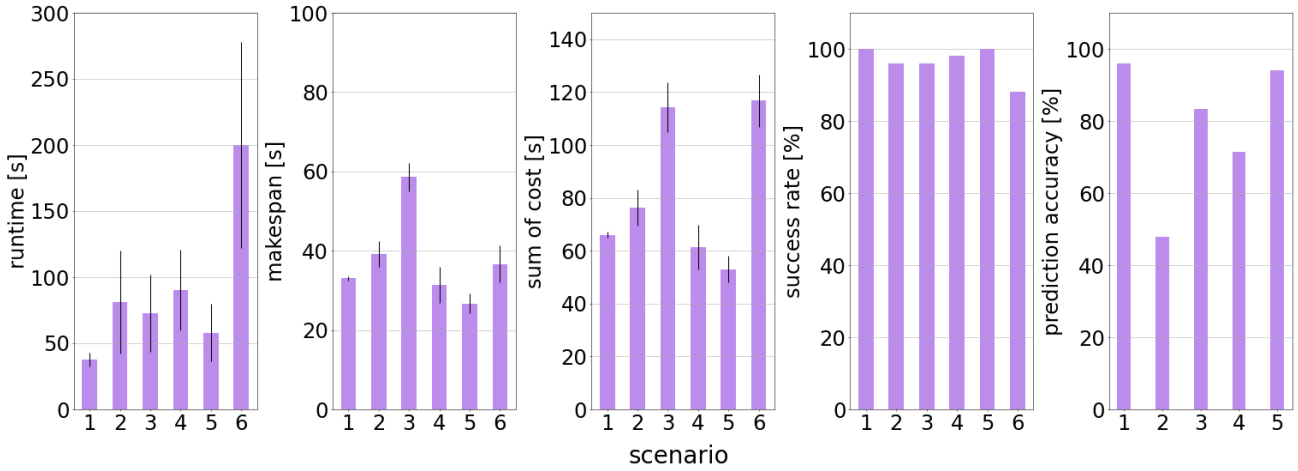

Figure 4: Scenario analysis (from left to right) for runtime, makespan, sum of cost, success rate, and prediction accuracy.

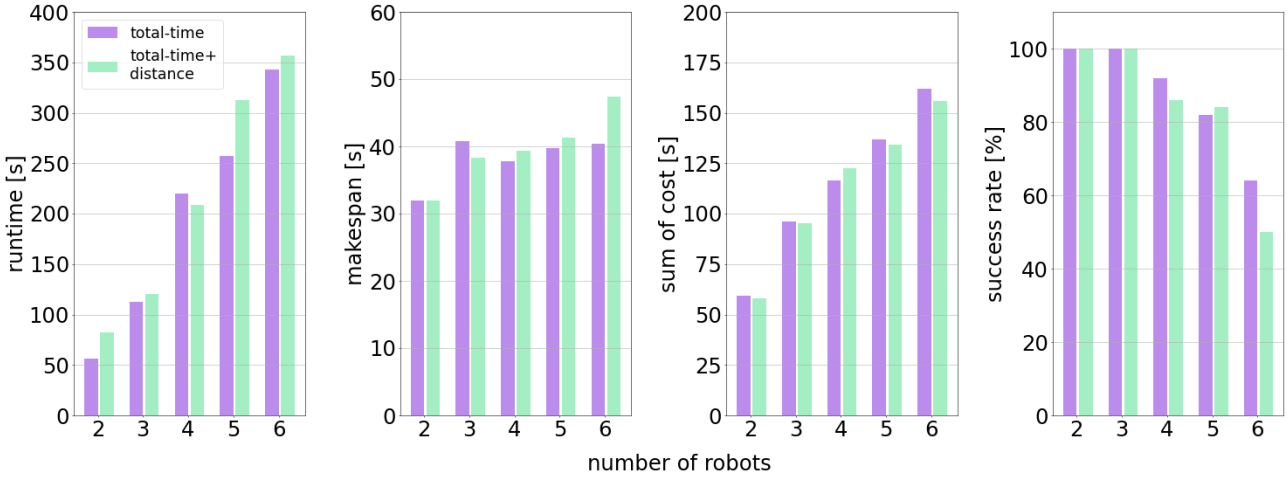

Figure 5: Scalability analysis according to the number of robots (2 to 6) and optimization metric (total-time and mixture of total-time and robot's distance).

occur. The number of robots increased from 2 to 6, and total time optimization and a mixture of total time and distance of each robot optimization were tested. In order to confirm the generation of an efficient plan, the unskilled action cost was set to high, and if the robot completed the mission through the unskilled action, it was considered a failure.

The experimental results can be seen in figure 5. For 2-3 robots, the framework always creates a collision-free plan. As the number of robots increases, runtime and soc increase significantly, whereas makespan shows a slight increase. This proves that in the case of multiple robots, all robots work without much waiting or detours through efficient planning. When the optimization function is mixed with distance, it has a slightly better soc than when total-time is used alone, but the runtime increases, and the success rate is low. It means that optimization metric with mixture makes it difficult for task planners to find plans with many predicates.

Runtime Distribution Figure 6 shows the runtime distribution as the number of robots changes. The task planning module takes a significant part of the runtime, and a large amount of time is in the plan generation of the task planner. Timeout of the task planner can be lower but results in a less efficient plan or failure to generate it. As the number of robots increases, the computation time of the collision detection module increases. This indicates that the framework spends considerable time figuring out collision waypoints and creating entry points with collisions.

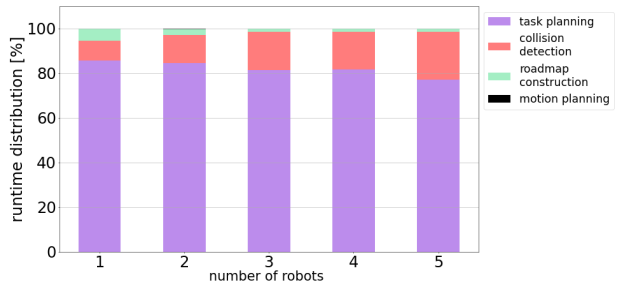

Figure 6: Runtime distribution for task planning, collision detection, roadmap construction, and motion planning of CBTAMP framework.

Conclusion

In this paper, we have proposed a conflict-based task and motion planning (CBTAMP) framework that combines task and motion planning to solve the multi-goal multi-robot problem. CBTAMP is different from other approaches in that it iteratively solves problems through continual information exchange between the discrete level task planner and the continuous environment motion planner. In addition, the novelty of our approach is that the conflict can be resolved in the task planner rather than the motion planner by utilizing information abstracted in the form of collision waypoints. Our experiments on six motivating scenarios show that CBTAMP generates a more efficient conflict-free plan according to environment and parameter changes, and challenging experiments with many robots in a narrow space show that the framework has scalability.

In future work, we intend to extend our framework by representing geometric constraints such as robot's orientation and acceleration in the high-level domain. In addition, more sophisticated hierarchical framework structures could be considered, which might combine not only the geometric reasoning for robots, but also the behavioral states of humans, about which there is always uncertainty (Faroni et al. 2020). This could allow the planner to reason in a more realistic environment, and to be used for case studies other than the scenarios presented in the paper.

References

Amato, N. M.; Bayazit, O. B.; Dale, L. K.; Jones, C.; and Vallejo, D. 1998. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 155–168.

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *International Conference on Automated Planning and Scheduling*.

Bohlin, R.; and Kavraki, L. E. 2000. Path planning using lazy PRM. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, 521–528. IEEE.

Burfoot, D.; Pineau, J.; and Dudek, G. 2006. RRT-Plan: A Randomized Algorithm for STRIPS Planning. In *ICAPS*, 362–365.

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.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proc. International Conf. on Automated Planning and Scheduling*.

Dornhege, C.; Gissler, M.; Teschner, M.; and Nebel, B. 2009. Integrating symbolic and geometric planning for mobile manipulation. In *2009 IEEE International Workshop on Safety, Security & Rescue Robotics (SSRR 2009)*, 1–6. IEEE.

Faroni, M.; Beschi, M.; Ghidini, S.; Pedrocchi, N.; Umbrico, A.; Orlandini, A.; and Cesta, A. 2020. A layered control approach to human-aware task and motion planning for human-robot collaboration. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 1204–1210. IEEE.

Fox, M.; and Long, D. 2003. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*, 20: 61–124.

Garrett, C. R.; Lozano-Pérez, T.; and Kaelbling, L. P. 2020. PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 440–448.

Hsu, D.; Jiang, T.; Reif, J.; and Sun, Z. 2003. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, volume 3, 4420–4426. IEEE.

Kaelbling, L.; and Lozano-Perez, T. 2010. Hierarchical task and motion planning inthe now. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; 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.

Keren, S.; Canal, G.; and Cashmore, M. 2021. Task-aware waypoint sampling for robotic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, 643–651.

Kim, B.; Wang, Z.; Kaelbling, L. P.; and Lozano-Pérez, T. 2019. Learning to guide task and motion planning using score-space representation. *The International Journal of Robotics Research*, 38(7): 793–812.

LaValle, S. M.; and Kuffner Jr, J. J. 2001. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5): 378–400.

Le, D.; and Plaku, E. 2017. Cooperative multi-robot sampling-based motion planning with dynamics. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, 513–521.

Lo, S.-Y.; Zhang, S.; and Stone, P. 2018. PETLON: planning efficiently for task-level-optimal navigation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 220–228.

Muñoz, P.; R-Moreno, M. D.; and Barrero, D. F. 2016. Unified framework for path-planning and task-planning for autonomous robots. *Robotics and Autonomous Systems*, 82: 1–14.

Nissoux, C.; Siméon, T.; and Laumond, J.-P. 1999. Visibility based probabilistic roadmaps. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, volume 3, 1316–1321. IEEE.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219: 40–66.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. S.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Twelfth Annual Symposium on Combinatorial Search*.

Thomas, A.; Mastrogiovanni, F.; and Baglietto, M. 2019. Task-motion planning for navigation in belief space. *arXiv preprint arXiv:1910.11683*.

Thomason, W.; and Knepper, R. A. 2019. A unified sampling-based approach to integrated task and motion planning. In *International Symposium on Robotics Research (ISRR)*.