Multi-Robot Task and Motion Planning With Subtask Dependencies

James Motes, Read Sandström, Hannah Lee, Shawna Thomas, and Nancy M. Amato

***Abstract*—We present a multi-robot integrated task and motion method capable of handling sequential subtask dependencies within multiply decomposable tasks. We map the multi-robot pathfinding method, Conflict Based Search, to task planning and integrate this with motion planning to create TMP-CBS. TMP-CBS couples task decomposition, allocation, and planning to support cases where the optimal solution depends on robot availability and inter-team conflict avoidance. We show improved planning time for simpler task sets and generate optimal solutions w.r.t. the state space representation for a broader range of problems than prior methods.**

***Index Terms*—Task planning, motion and path planning, multi-robot systems.**

I. Introduction

THIS work presents a multi-robot integrated task and motion planning method capable of finding solutions for multiply decomposable tasks with sequential subtask dependencies which are optimal w.r.t. the state space representation.

We focus on transportation-like tasks where the objectives of a task correspond to physical locations in the environment. In the simplest example, a transportation task is defined by the movement of an object between a pick-up and a delivery location. We present an approach that finds a solution to a set of transportation tasks for a heterogeneous multi-robot system (MRS) with varying capabilities while leveraging inter-robot interactions. These interactions allow the task object to be handed from one robot to another.

The option to interact in this way creates multiply decomposable tasks. Optimal multi-robot solutions for multiply decomposable tasks require coupled computation of the task decomposition, subtask allocation, and robot motion plans.

Traditional integrated task and motion planning (TMP) methods are designed for single-robot systems and do not naturally contain multi-robot semantics. They do not easily account for multi-robot collisions during motion planning and would need to treat the entire MRS as a single composite agent to address this, which quickly becomes intractable as the size of the robot team increases.

Method	Pathfinding	Motion Planning	Task Decomposition	Task Allocation	Coupled	Interactions
CBS [1]	X					
CBS-MP [2]		X				
TCBS [3]	X			X	X	
IT Method [4]		X	X			X
TMP-CBS		X	X	X	X	X

Fig. 1. Method Capabilities.

Previous multi-robot work handles parts of this with varying coupled and decoupled natures (Fig. 1). To the best of our knowledge, our approach is the first to couple all three parts to find the best available solution for these multiply decomposable tasks. We demonstrate improved scalability by handling twice as many robots as existing methods in both continuous and discrete environments. We also show the importance of coupling in problems with multiple possible decompositions and sequential subtask dependencies, where a decoupled approach often yields poor solution costs.

II. Related Work

The multi-robot transportation task problems addressed in this work require task decomposition, subtask allocation, and motion planning. Previous methods have addressed various subsets of these features for simpler transportation problems (Fig. 1). Some assume an optimal state space discretization and operate in a grid world [1], [3], while others perform motion planning in a continuous space [2], [4]. Many task allocation works requires a pre-computed decomposition [3], [5]. Our prior work, the Interaction Template (IT) method [4], computes a decomposition, but assumes robot availability for allocation. The method presented in this work combines all three features for optimal multi-robot task and motion planning for transportation problems.

A. Task Planning

A task plan consists of a sequence of actions transitioning from a start state to a desired goal state. Multi-robot task planning often consists of two components: task decomposition and task allocation [6].

Manuscript received September 10, 2019; accepted February 6, 2020. Date of publication February 26, 2020; date of current version March 9, 2020. This work was supported in part by the NSF Awards CCF-1423111 and EFRI-1240483 and in part by the DREU Program. The work of Hannah Lee was supported in part by the NSF funded CRA-WP Distributed Research Experiences for Undergraduates (DREU) Program. This letter was recommended for publication by Associate Editor M. Fabian and Editor J. Yi upon evaluation of the reviewers' comments. *(Corresponding author: James Motes.)*

James Motes and Nancy M. Amato are with the Parasol Lab, Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL 61820 USA (e-mail: jmotes2@illinois.edu; namato@illinois.edu).

Read Sandström and Shawna Thomas are with the Parasol Lab, Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77840 USA (e-mail: readamus@tamu.edu; sthomas@tamu.edu).

Hannah Lee is with the Colorado School of Mines, Golden, CO 80401 USA (e-mail: hannahlee@mymail.mines.edu).

Digital Object Identifier 10.1109/LRA.2020.2976329

1) Task Decomposition: Multi-robot task decomposition considers factors such as environment and robot-types and decomposes abstract tasks into subtasks [7]. In this work, we seek decompositions resulting in the single task, single robot, time extended (ST-SR-TA) classification of task allocation problems. In these decompositions, each subtask can be performed by a single robot and defines the requirements for the robot to complete the subtask. Additionally, robots can only perform one subtask at a time though they can perform multiple subtasks over an extended period of time.

A task decomposition is optimized w.r.t. the utility of the MRS executing the set of subtasks produced by the decomposition. For example, when minimizing time the optimal set of subtasks is the one that results in the shortest completion time of the abstract task. This is often handled by a centralized coordinator which we also employ.

2) Task Allocation: Multi-robot task allocation seeks a feasible assignment of a set of tasks to a set of robots. This problem has been studied extensively and has been shown to be NP-hard [8]. The authors in [8] provide a taxonomy of the varying task allocation problems based on factors including the number of robots required for a subtask and the capacity of a robot to perform subtasks simultaneously. This taxonomy is extended in [9] to address interrelated utilities and constraints of tasks.

The highest level of the dependencies presented in [9] is *complex dependencies*. These problems consist of the allocation for tasks with multiple possible decompositions [10], and the effective utility of a robot for a subtask depends on the schedules of other robots. Optimal solutions then require a coupled decomposition and allocation approach deciding *which* set of subtasks should be assigned to *who* and *when*.

Most work on ST-SR-TA problems is focused on lesser dependency classifications laid out in [9]. These have often been modeled as the Multiple Traveling Salesmen and Vehicle Routing problems with additional complexity introduced when adding precedence or synchronization constraints.

Some work has been done on multiply decomposable tasks with complex dependencies. The authors in [11] address tasks with "intra-path" constraints when considering subtasks consisting of clearing possible paths that are used to complete other subtasks. In our previous work, the Interaction Template method (IT) [4], we present a multi-robot integrated TMP approach for payload transportation that has a partial coupling of task decomposition and allocation.

B. Motion Planning

Motion planning considers a state space that is continuous and usually intractable to represent explicitly. This state space is comprised of the set of all possible robot configurations and is known as the *configuration space* (C_{space}) [12]. The problem of motion planning is finding a continuous path from a start to a goal through the subset of C_{space} consisting of valid configurations called *free space* ($\mathcal{C}_{free}$). To handle the complexity of motion planning, sampling-based methods such as the Probabilistic Roadmap Method [13] seek to construct a discrete graph approximation of the C_{space} known as a roadmap. Paths are found by searching over this roadmap.

The Conflict Based Search algorithm [1] provides an optimal multi-robot pathfinding solution given a discretization of the state space. CBS uses a two level search; with a high level tree resolving inter-robot conflicts and a low level search finding individual robot paths to provide optimal team solutions. The authors demonstrate the method on a grid roadmap. The work in [2], presents a continuous space sampling-based motion planning adaptation of this, CBS-MP, that provides optimal multi-robot motion planning solution w.r.t. the individual robot roadmaps.

C. Integrated Task and Motion Planning

The integration of task planning and motion planning has been well studied for single robot systems [14]–[18]. These methods integrate high-level decisions of task planning with the low level constraints of path planning [19], [20]. Traditionally this has been done by taking a *task planning first* approach and using a motion planner to verify actions considered by a task planning method. These single robot designed methods are lacking multi-robot semantics and need to treat the entire MRS as a single composite agent when validating motion actions to account for these collisions. As this approach becomes intractable, we do not compare to these methods in this work.

In our previous work, the IT method [4], we take a *motion planning first* approach, by integrating the task semantics into a *combined roadmap* for a robot team to create a unified representation which demonstrated improved scalability over generalizing the task planning first approaches to TMP. We build off this motion planning first approach in this work.

In [4], the IT method considers tasks for which the system can choose to perform inter-robot interactions, handing the payload off from one robot to another, thus ending one subtask and beginning another. By searching the unified representation of task and motion semantics in the combined roadmap, the IT method is able to generate a task decomposition that accounts for individual subtask execution cost by searching this. However, it assumes that there will be a robot available to hand the payload over whenever the robot currently transporting the payload arrives. For single task problems, this occasionally leads to small inefficiencies, but when considering multiple tasks at once, the assumption of robot availability is often extremely costly.

In recent years, multi-robot research has looked at the combined target-assignment and pathfinding (TAPF) problem [21]. This problem first plans the task allocation and then collision-free paths agent-task allocations. The separation of the two stages leads to inconsistencies in the costs used during task allocation and the cost resulting from the feasible collision-free path found for that task allocation.

Two recent works have extended the CBS algorithm to provide coupled TAPF solutions [3], [5] and show how this outperforms the decoupled approaches. CBS-TA (task allocation) provides a method that solves the *anonymous* multi-agent pathfinding problem [5]. This problem is a variation of the standard or *labeled* pathfinding problem where instead of each robot being preassigned a goal location, the allocation of goal locations

to robots is included in the problem. CBS-TA takes an approach similar to M* where it generates additional complete task allocations on demand as they are needed. Each generated task allocation is a new root in the high-level portion of the CBS algorithm resulting in a conflict forest instead of the traditional tree.

Task CBS (TCBS) solves multi-robot transportation problems [3]. It takes a different approach than CBS-TA and only assigns a single subtask at a time. Additional tasks are only assigned once all path conflicts have been resolved. Whenever a new task assignment is added to the plan, a new CBS node is generated for every remaining task-robot allocation combination. This can lead to a high branching factor as the number of tasks and agents scale.

D. IT+TCBS-MP

Both CBS task allocation extensions [3], [5] rely on pre-computed optimal task decompositions. In contrast, we present a task *planning* method that computes both the task *decomposition* and *allocation*. As a result, our approach is able to handle multiply decomposable tasks where the optimal decomposition is reliant upon the robot availability.

Existing methods solve components of the problem we address and can be combined to provide a solution in a decoupled manner. As the method presented here provides a coupled approach to the problem, we also study a new combination of existing methods to provide a decoupled alternative that we compare against in our experiments.

TCBS can be adapted to handle a pre-computed decomposition in which each task is comprised of an ordered set of sequentially dependent subtasks like the IT method output by modifying its allocation branching function. When considering a new set of allocation possibilities a subtask can only be considered for allocation if its preceding subtask has been allocated in the current solution in the high-level search, and a robot can only be considered for allocation if its most recent allocation does not have an unallocated subsequent task in the current solution in the high-level search. To generate these decompositions with sequential dependencies we use the IT method [4] and replace the auction allocation with TCBS to create the IT+TCBS approach. This computes the decomposition decoupled from the allocation and path planning components. Additionally, as the task allocation branching function of TCBS is not reliant upon the grid roadmap structure used in CBS [1], it can be extended to continuous space motion planning by merging the task branching into CBS-MP [2]. Altogether we label this IT+TCBS-MP.

This combined approach is provides solutions for multiply decomposable tasks with the sequential subtask dependencies but with no guarantee of optimality. In this letter, we present an approach that uses a unified representation of the motion, task, and *availability* semantics, to simultaneously consider the task decomposition, allocation, and motions required to find an optimal solution for a set of tasks.

III. Problem Definition

We wish to find an optimal solution to a set of decomposable transportation tasks T. A transportation task $t \in T$ is defined by a pair of pick-up and delivery locations. Planning methods are given a team of robots $R = R_1 \cup R_2 \cup \ldots \cup R_K$, where R_i is the set of robots of type i, and a set of possible interactions I. A valid team solution consists of individual robot motions and possibly interactions $I_{ij} \in I$ between robots of type i and j that transition all payloads from pick-up to delivery locations. We consider interactions which model the handing over a task from a robot $r_1 \in R_i$ to another $r_2 \in R_j$. A set of placements P for these possible interactions is either given or computed.

An optimality metric is required. Two common multi-robot options are the sum of all costs or the maximum cost of any task cost. This cost can be time, energy, etc..

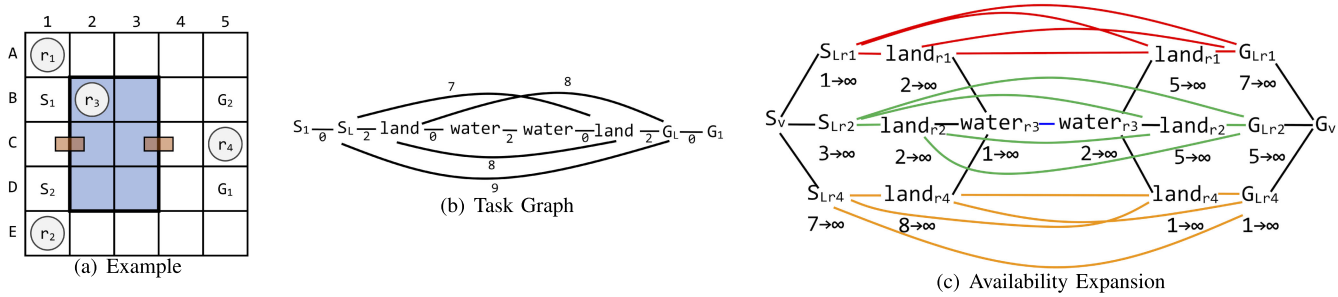
Decomposition: With these interaction placements P, transportation task $t \in T$ can be decomposed into subtasks S_t also defined by pick-up and delivery location pairs. The subtasks in S_t have a sequential ordering, and the delivery location of a subtask $s_i \in S_t$ coincides with the pick-up location of the subsequent subtask $s_{i+1} \in S_t$.

An interaction between the robots allocated to s_i and s_{i+1} marks the end of s_i and the beginning of s_{i+1} as the payload is handed over between the robot executing s_i and the robot allocated to s_{i+1}. All robots involved must be present for the payload to be exchanged thus creating a dependency on the cost of robots performing subtasks on either side of an interaction. Thus a valid transportation task decomposition consists of ordered feasible subtasks and interactions that move the payload between the designated pick-up and delivery locations, and a valid decomposition for a set of transportation tasks is a set of valid feasible subtask and interaction sequences.

Allocation: We assume each robot is only able to perform one subtask at a time, and that each subtask only requires a single robot. With the time extended assignment of subtasks, this results in decompositions of the tasks into single robot, single task, time extended allocation problems (SR-ST-TA) [8]. A valid allocation thus consists of one robot assigned to each subtask such that each robot is capable of performing its assigned subtasks. The dependency of allocation costs between subtasks comprising the multiply decomposable transportation tasks falls under the complex dependency classification described by [9]. As such the decomposition and allocation of the set of transportation tasks cannot be solved optimally in a decoupled manner. A complete multi-robot TMP method which can produce optimal solutions for multiply decomposable tasks with sequential subtask dependencies thus requires a coupled choice of decomposition and allocation.

Example: In the lake scenario depicted in Fig. 2(a), the interaction placements are dock locations. These allow land and water robots to exchange a payload when executing a transportation task. There now exists multiple decompositions for the transportation tasks in Fig. 2(a). Both can be decomposed either into a single *land* subtask moving the payload around the lake or into a sequence of subtasks: *land, water, land* with the subtasks splitting at the dock locations.

Individually, the shortest decomposition for each of the transportation tasks is the *land, water, land* decomposition. When considered together, the water robot can only perform one subtask at a time forcing the other task to wait on the water robot to become available again. When considering the waiting time, the



(a) Example (b) Task Graph (c) Availability Expansion

Fig. 2. (a) The team of four robots are assigned a pair of transport tasks from S_i to G_i respectively. Robot R_3 resides in a lake (blue), and a dock (brown) exists on either side where the robots have the ability exchange the task payload. (b) The Task Graph extracts the potential subtask start/end points from Task 1 in Fig. 2(a) into vertices with edges corresponding to paths between the locations in the underlying roadmap (prior to consideration to motion conflicts). (c) The vertices in the Task Graph are crossed with robots and available intervals to create the task planning search space. S_{r_i} corresponds to the interval for each of the land robots that can reach S_L. The beginning of the interval corresponds to the time each robot would be able to reach and start performing a subtask at the location. The edge weights are path-dependent and thus not shown in the figure.

land, water, land decomposition solution results in a greater cost than the single *land* solution for whichever task is allocated the water robot second. The optimal solution for the set of tasks is one task decomposed into *land, water, land* and one decomposed into just *land* with each subtask allocated to a single robot $\{T_1(R_1, R_3, R_4), T_2(R_2)\}$. Discovering this solution requires coupled task decomposition and allocation planning to account for sequential dependencies within the decomposition options. Tasks requiring this coupled framework make up the complex dependency classification in [9].

IV. Method

We present integrated Task and Motion Planning Conflict-Based Search (TMP-CBS) as the first multi-robot task and motion planner for multiply decomposable tasks with sequentially dependent subtasks which can extract team plans that are optimal with respect to the current individual robot roadmaps. We first outline the CBS [1] algorithm. We then map the CBS approach to task planning. Finally, we integrate the task planner with CBS and CBS-MP [2]. The task component is the same for both discrete and continuous problems, so our explanation and example (Fig. 2(a)) will use CBS and a discrete grid roadmap for clarity of exposition.

A. Conflict-Based Search (CBS)

CBS is a MRPF algorithm that finds the optimal set of conflict free paths for a set of robots [1]. It first plans for all robot paths individually, and then iteratively finds and resolves conflicts between these paths where a conflict is defined by two robots occupying the same vertex or edge in the roadmap at the same time. The method uses a low-level pathfinding method to find individual robot paths and a high-level *Conflict Tree* (CT) in order to manage conflicts.

Nodes in the CT contain a solution to the MRPF problem consisting of the set of individual robot paths, a solution cost, and a set of path constraints for each robot. The root node is the initial path found for all robots with no constraints. The method selects the cheapest unprocessed node in the tree and checks for conflicts between paths. If no conflicts are found, the goal node is returned containing the optimal solution.

When a conflict is discovered within a CT node, the conflict is resolved by splitting the node into two children, one for each robot. Both children inherit the set of path constraints for each robot from the parent node. Each child is given an additional constraint for one of the conflicting robots at the time and location of the conflict, and the path for that robot is replanned with the new constraint. Thus a node exists for both possibilities where either robot is given precedence for that location at that time. CBS guarantees optimality by examining both possibilities as a node is split into two children for all conflicts [1].

B. Task Planning

In this work, we propose to adapt the CBS algorithm to the problem of solving a set of tasks given a set of robots. A node in the adapted CT contains a solution and a set of constraints (Fig. 3). The solution consists of a decomposition, allocation, and corresponding subtask paths for each individual task. The constraint set is comprised of two subsets, motion constraints and allocation constraints. A motion constraint is the same as in CBS (robot r, vertex v, time x) with an additional task component indicating that the motion constraint only affects the robot r when it is planning for a subtask $s \in$ task t for a constraint notion of (t, r, v, x).

An allocation constraint indicates a time interval with a start/end location between which a robot is *unavailable* for the constrained task. A task plan satisfying an allocation constraint ensures that the robot is able to reach the start location of the constraint at the beginning of the constraint time interval and does not require the robot to arrive at any location at a time earlier than the robot can reach following the end of the allocation constraint. Allocation constraints usually correspond to other subtasks assigned to the robot in question and the time intervals and start/end locations associated with those subtasks. We denote an allocation constraint (t, r, s) with task t, robot r, and subtask allocation s containing the time interval and start/end locations.

1) Low-Level Search: In the initial pathfinding CBS algorithm [1], the goal is to find a set of feasible collision-free robot paths. These individual paths are computed with a low-level pathfinding algorithm [1]. The authors use A* over a two

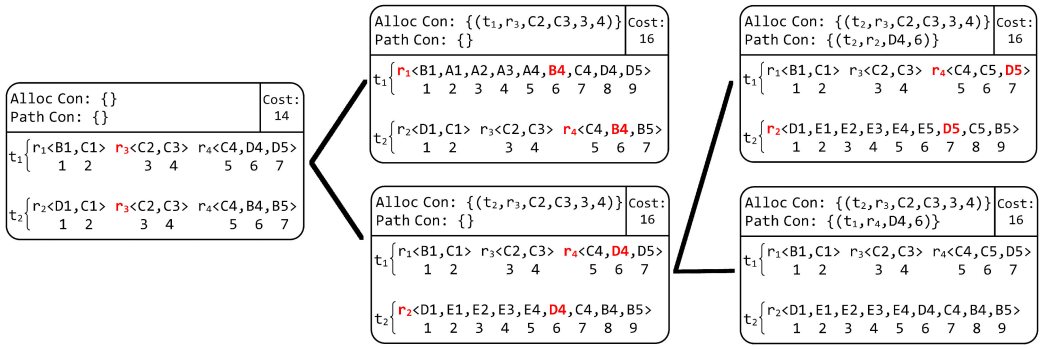


Fig. 3. The Conflict Tree (CT) for the example in Fig. 2(a). Each node contains allocation and motion constraints (upper left), a solution cost (upper right), and a plan for each task (lower). The root node contains an allocation conflict (red), and the first lower child node contains a motion conflict (red). The lower right node contains a valid solution.

dimension `location x time` search space in the original work for this. We map this concept to task space by creating a unified representation for task, motion, and robot availability semantics so that searching over it produces the optimal task decomposition, allocation, and robot motions.

In our previous work [4], we developed a *combined roadmap* that integrates task semantics within a set of roadmaps for all robot-types in the system. This is used to capture inter-robot interactions exchanging the payload in transportation tasks. Virtual nodes are used to represent the initial pick-up and delivery locations with 0 weight edges connecting the virtual nodes to valid robot configurations at those locations. Plans across the combined roadmap produce task decompositions and preliminary motion plans for the individual subtasks. The costs associated with the subtasks in the decomposition assume that a robot is immediately available to perform each subtask and thus does not account for time taken for robots to travel to the subtask start location. Additionally, it relies upon reactive collision avoidance behavior and does not include the cost of collision avoidance in the cost of subtasks when computing the decomposition either. Here we expand the combined roadmap representation to capture robot availability and use the CBS structure to account for motion collisions.

The CBS conflict tree approach results in repeated searching of the space, so we construct the *task graph*, a condensed representation of the combined roadmap. Vertices in the task graph correspond to subtask start/end locations which in our combined roadmap correspond to interaction locations where robots can exchange a payload and the virtual nodes for the task's initial pick-up and final delivery locations. Edges between these vertices correspond to paths in the combined roadmap between these locations.

For example, in the lake scenario depicted in Fig. 2(a) the docks located at $(C1, C2)$ and $(C3, C4)$, are interaction locations and thus correspond to possible subtask start/end vertices in the task graph (the two *land-water* vertex pairs in Fig. 2(b)). The task graph in Fig. 2(b), corresponds to task t_1 defined by pick-up/delivery locations S_1, G_1 in Fig. 2(a). S_1 in the task graph is the virtual start node, and S_L is the land robot subtask start at $B1$. The edge between S_L and the left *land* vertex corresponds to the path (B_1, C_1) in Fig. 2(a).

A plan across the task graph produces a decomposition for a task without considering robot availability. This produces solutions similar to the IT method which assumes robot availability. As discussed in Section II.A, this can lead to suboptimal decompositions as it does not account for the time taken for a robot to reach a subtask start or for allocations of a robot to subtasks outside of the current task.

For example, when considering the scenario in Fig. 2(a), both tasks would compute a decomposition of three subtasks (`land`, `water`, `land`) with an interaction between each subtask. The middle, `water`, subtask for both tasks must be assigned to r_3, thus forcing one of the task plans to have an extended first `land` subtask, as r_1 or r_2 waits on r_3 to return to perform its second subtask. While this is a valid solution, it is suboptimal as the decomposition could instead consist of a single subtask directly transporting the payload from the pick-up to the delivery location arriving earlier than waiting for r_3 to become available again.

To address this, we add another dimension to our representation denoting robot availability. As considering every time step each robot is available will lead to a drastic increase in the search space, we condense contiguous intervals of robot availability for each vertex in the task graph. This represents the first instance a robot is able to start a subtask and the last instance the robot is able to complete a subtask at the location corresponding to the task graph vertex before violating an allocation constraint (usually this indicates leaving for another allocation). A search over this space produces an optimal individual task plan (decomposition and allocation) w.r.t. the existing allocation constraints of the robots.

This is modeled after the Safe Interval Path Planning technique [22], which condenses the time dimension of a search space with constraints from dynamic obstacles or other robots. In Fig. 2(c) the expansion of the land start node S_L by robot availability and the connections to the virtual start node are shown. S_{r_4} has an availability interval of $7 \rightarrow \infty$ as robot r_4 can reach the start location at time 7.

2) High-Level Search: The high-level search (Alg. 1) builds a CT tree to iteratively build an optimal plan for a set of transportation tasks and a set a robots. CT nodes contain a solution for the set of tasks consisting of decompositions, allocations, and corresponding motion plans.

Algorithm 1: TMP-CBS.

```
1: procedure Mainset of tasks T, set of robots R
2:     Init N with low-level plans for the individual tasks
3:     Insert N into OPEN
4:     while OPEN not empty do
5:         N ← best node from OPEN // lowest-cost solution
6:         Find first subtask allocation conflict in task plans
7:         if N has no task allocation conflicts then
8:             Find first motion conflict in robot paths
9:             if N has no motion conflicts then
10:                return N.solution // N is a goal node
11:            M ← (t_i, r_i, t_j, r_j, v, x) // motion conflict
12:            for all task t_i in M do
13:                Q ← GenerateMotionChild
14:            continue
15:        C ← (t_i, t_j, r, s_i, s_j) // allocation conflict
16:        for all task t_i in C do
17:            Q ← GenerateAllocChild
18:            Insert Q into OPEN
```

We build upon CBS conflict detection and resolution, by adding an *allocation conflict*. An allocation conflict between two tasks t_1, t_2 is defined by the allocation of the same robot r to a subtask in both tasks $s_i \in t_1$, $s_j \in t_2$ such that assignment time of r to s_i and s_j is overlapping, or the minimum time r can reach the start location of s_j after completing s_i is greater than the assigned time of r to s_j.

Upon detection of an allocation conflict $C = (t_1, t_2, r, s_i, s_j)$, two child nodes are created, one for each task involved in the conflict. Both child nodes receive the parent node's solution and constraint set, and each child node is given one additional allocation constraint for its corresponding task, (t_1, r, s_j) and (t_2, r, s_i) as shown on Alg. 1 line 17. A new low-level plan is then computed for the corresponding task in each node that *recomputes* the decomposition, allocation, and motion plan by searching the unified representation. This allows limitations on robot availability to affect the task decomposition.

This process can be seen in Fig. 3 for the lake scenario in Fig. 2(a). Both initial individual task plans use r_3 during overlapping intervals creating an allocation conflict. This is resolved by creating a child node for each task with a new constraint corresponding to the other task's allocation of r_3. The updated solution for each child is shown in Fig. 3.

C. Integration of Task and Motion Planning

If we only considered allocation conflicts when evaluating a CT node, we will produce solutions similar to the allocation and pathfinding methods [21] where the separation of task allocation and collision-free path planning leads to inconsistencies in the costs used for allocation and the costs of the collision-free paths (the distinction still standing that we additionally simultaneously compute the decomposition for a complete task planning approach instead of just task allocation). The cost of a robot executing a subtask in the task graph is dependent upon the path it takes between subtask start/end locations as well as the time spent waiting on the next robot to reach the exchange the point. When motion conflicts are considered the cost of this path can be altered by a longer path being required either for the transition of a robot from the end of one subtask to the start of another or the path taken during execution of the subtask.

Instead, we want an integrated TMP method to account for possible motion conflicts. As mentioned previously, the task components are the same for both discrete CBS and continuous CBS-MP. We will explain the integration w.r.t. to CBS for clarity of the exposition and examples. A motion conflict $M = (t_i, r_i, t_j, r_j, v, x)$ is defined by a pair of robots r_i, r_j, the tasks they are currently executing a subtask of t_i, t_j, the vertex v the robots occupy in the plan, and a time x at which the robots occupy v in the plan (motion conflicts can also occur along edges, but we point the reader to [1] and [2] for details as we handle them in the same fashion).

The method first checks for allocation conflicts, and if none are found checks for motion conflicts. The edges in the unified task graph-availability space correspond to paths between subtask start/end locations in the underlying combined roadmap. To check for motion conflicts, the paths of the current task plans are extracted and validated.

The complete path for each robot across all of its allocations is compiled. These paths now mirror the paths checked for conflicts in CBS or CBS-MP, and conflicts are found in the same manner. The only distinction is tracking which task each robot is currently allocated to throughout robot paths.

Upon discovering a conflict, two child nodes are created, one for each task-robot $(t_i, r_i), (t_j, r_j)$ pair. Both child nodes are given the solution and constraint set of the parent, and each is given an additional motion constraint for the task-robot pair, $M_i = (t_i, r_i, v, x)$ and $M_j = (t_j, r_j, v, x)$ respectively. Given the new motion constraint, the transition costs for the robot in the unified task graph-availability space are updated in the respective node. The low-level search is invoked to update the individual task plan (decomposition, allocation, and corresponding motion plans). This allows collision avoidance to affect the decomposition and allocation. The method continues to evaluate the next best node in the CT tree until the first conflict-free node is discovered. This node contains the optimal solution complete with the decomposition, allocation, and motion plans.

Considering these motion constraints results in changes to the available intervals as well as the cost of traversing edges in the task search space. As the cost of a path between two locations depends upon the start time of that path, the cost of traversing an edge in the unified representation space must be computed by a search in the underlying roadmap w.r.t. the motion constraints for that robot during search time. These costs are then validated with the availability intervals to find optimal paths in task space w.r.t. both the task allocation constraints and the motion constraints. The two-level search framework discovers and resolves both types of conflicts to produce the optimal multi-robot task plan.

When considering the first lower node in Fig. 3 for the lake scenario, although there are no task conflicts, there is a path conflict at timestep 6 with R_4 and R_2 both being at D4 while performing T_1 and T_2 respectively. The path conflict is converted

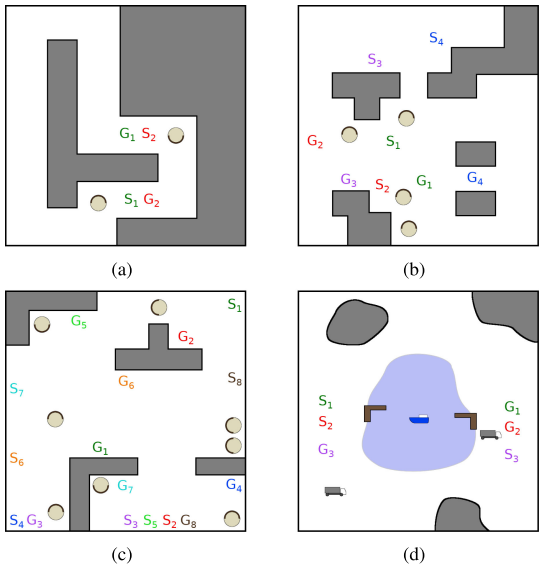


Fig. 4. The experiment environments. Obstacles are shown in gray. Robots are illustrated as tan disks, trucks, and boats. The i^{th} task start/goal is shown as S_i/G_i in matching colors. (a–c) Disc robots must solve a set of tasks in several environments. (d) Two trucks and a boat must move three payloads to opposite sides of a lake (blue), possibly leveraging interactions at the docks (brown).

to constraints and the node branched the same as CBS/CBS-MP with the addition of the constraint being unique to the individual task being planned (Fig. 3).

V. Experiments

We demonstrate the performance and expanded planning capability of TMP-CBS by performing experiments for sets of non-decomposable simple transportation tasks and sets of multiply decomposable transportation tasks in discrete grid worlds and continuous state spaces. The continuous space experiments demonstrate the full capability of the method, while the grid world state space representation experiments highlight the task planning without the complexity resulting from the continuous space multi-robot motion planning. Each experiments was run with a set of 20 random seeds.

We optimize for the sum-of-costs by taking the sum of the end time of each task, and we evaluate our performance against TCBS [3], a recent task allocation and pathfinding algorithm that also extended CBS [1] and performed favorably against existing methods. For the multiply decomposable tasks, we use the IT method to generate the decomposition and handle the dependencies with the modifications described in Section II.D (IT+TCBS) and use CBS-MP for the motion planning component of TCBS in continuous space experiments (IT+TCBS-MP).

A. Non-Decomposable Simple Tasks

We tested in three environments for non-decomposable simple transportation tasks with a scaled number of robots and tasks. We use the 2×2 (robots x tasks, Fig. 4(a)) and 4×4 (Fig. 4(b)) experiments used in the original TCBS work [3] as well as the 8×8 problem (Fig. 4(c)). We evaluate the 4 and 8 task scenarios with smaller robot teams as well. For the 2×2 and 4×4 experiments, we evaluated both methods in a continuous state space using the sampling-based motion planning approaches describes in earlier sections (Fig. 5(b)). Additionally, all three environments were evaluated with a pre-computed grid world state space representation (Fig. 5(a)).

TMP-CBS shows an increased scalability as the number of tasks and robots increase. This is due to the allocation branching of the TCBS high-level search. For every remaining (sub)task it creates a child node in the CT for every robot that can be assigned to the task ($\mathcal{O}(|R||\text{remaining } T|)$ children from every allocation branching). By instead mapping the CBS planning process to the task space instead, we consider the best known option for each (sub)task allocation and resolve conflicts between these allocations.

Additionally, as TCBS builds up possible solutions one (sub)task allocation at a time, the selection of the current best node considers incomplete solutions which often results in extensive effort and CT nodes spent resolving motion conflicts for poor allocation combinations. For example, it will often be the case that the cost of a solution containing two allocations will be less than a solution containing four allocation regardless of how poor a choice the two allocations of the first solution are. The high branching factor in TCBS that results from these factors limited it from completing the 8x8 environment as it would overflow the memory and crash the computer running the experiment. As TMP-CBS builds complete solutions, we avoid this issue. TMP-CBS does have limits due to memory usage as the CT grows. This can happen due to a high number of tasks and robots in the 12×12 scenario or the natural increase in the number of allocation conflicts when more tasks share fewer robots (Fig. 5(a)).

The continuous space problems predictably require more extensive planning than the grid world counterparts. Interestingly, TMP-CBS is able to solve the 4×4 problem faster than the 2×2 problem. This indicates the extreme effect that tight spaces can have on the planning time when resolving continuous space motion conflicts as the 4×4 environment is more open. TCBS is unable to complete the 4×4 problem in the continuous space version. This is due to attempting to resolve poor allocation combinations that introduce high levels of motion conflicts by evaluating CT node solution cost with incomplete solutions resulting in several hours of planning time for a single seed or overflowing the memory.

B. Multiply Decomposable Tasks

We demonstrate the increased planning capability of TMP-CBS by considering a set of multiply decomposable transportation tasks where the decomposition options come from the robot team's ability to interaction and handover a payload from one robot to another. We use a lake environment where the land and water robots are able to exchange a payload at dock locations (Fig. 4(d)). Each task has three possible decompositions as it can either be completed with single subtask moving around the lake, or three sequentially dependent subtasks either passing through the left or right dock first. This creates 27 possible decomposition combinations to consider when planning with all allocation possibilities for each of these and corresponding motion plans.

Discrete Grid World

Method	Agents x Tasks	Path Cost	Planning Time		CT Nodes
			Avg.	Std. Dev.	
Non-Decomposable Tasks					
TCBS	2x2	34	0.02	0.01	20
	2x4	98	0.37	0.01	40
	4x4	78	1.33	0.31	1232
	4x8	-	-	-	-
	8x8	-	-	-	-
TMP-CBS	2x2	34	0.03	0.01	10
	2x4	98	0.16	0.01	40
	4x4	78	0.12	0.03	24
	2x8	-	-	-	-
	4x8	384	75.5	0.46	7165
	8x8	270	0.53	0.12	20
	12x12	-	-	-	-
Multiply-Decomposable Tasks					
IT-TCBS	3x3	114	1.47	0.35	500
TMP-CBS	3x3	94	0.15	0.01	12

(a)

Continuous Space

Method	Agents x Tasks	Path Cost		Planning Time		CT Nodes	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Non-Decomposable Tasks							
TCBS-MP	2x2	1262	104.7	110.8	185.7	47.11	34.48
	4x4	-	-	-	-	-	-
TMP-CBS	2x2	1262	104.7	93.88	149.9	87.10	140.4
	4x4	2949	56.20	11.79	8.340	12.32	10.15
Multiply-Decomposable Tasks							
IT-TCBS-MP	3x3	-	-	-	-	-	-
TMP-CBS	3x3	3955	90.29	6.590	2.34	51.42	14.96

(b)

Fig. 5. Experiment results. Path cost is in timesteps. Planning time is in seconds. The edges in the continuous-space problems averaged 50 timesteps.

TMP-CBS is able to find optimal solutions as opposed to the decoupled IT+TCBS/TCBS-MP. This is due to the coupled planning on unified representation of the decomposition, robot availability, and motion plan used in TMP-CBS. Additionally, the high branching factor in TCBS for the reasons described before are again compounded when considering continuous space motion planning resulting in the inability of IT-TCBS-MP to find a solution.

VI. Conclusion

We present the first multi-robot integrated task and motion planning method capable of handling multiply decomposable tasks with sequentially dependent subtasks. We show that our method, which couples the task decomposition and allocation components that are typically treated separately, achieves improved performance on sets of non-decomposable simple tasks. Future work will seek to extend TMP-CBS to handle broader task abstractions and decompositions and will consider scheduling constraints such as deadlines and release times as well as synchronization, total, and partial ordering between tasks.

References

[1] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.

[2] I. Solis, R. Sandström, J. Motes, and N. M. Amato, "Representation-optimal multi-robot motion planning using conflict-based search," 2019, *arXiv:1909.13352*.

[3] C. Henkel, J. Abbenseth, and M. Toussaint, "An optimal algorithm to solve the combined task allocation and path finding problem," in *Proc. IEEE Int. Conf. Intel. Rob. Syst.*, 2019.

[4] J. Motes, R. Sandström, W. Adams, T. Ogunyale, S. Thomas, and N. M. Amato, "Interaction templates for multi-robot systems," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2926–2933, Jul. 2019.

[5] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Conflict-based search with optimal task assignment," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 757–765.

[6] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *Int. J. Adv. Robot. Syst.*, vol. 10, pp. 399–416, Nov. 2013.

[7] J. Chen, Y. Yang, and L. Wei, "Research on the approach of task decomposition in soccer robot system," in *Proc. Int. Conf. Digit. Manuf. Autom.*, Dec. 2010, pp. 284–289.

[8] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[9] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.

[10] R. Zlot and A. Stentz, "Complex task allocation for multiple robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 1515–1522.

[11] E. G. Jones, M. B. Dias, and A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints," *Auton. Robots*, vol. 30, no. 1, pp. 41–56, 2011.

[12] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-Free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, pp. 560–570, Oct. 1979.

[13] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[14] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 5002–5008.

[15] C. R. Garrett, T. Lozano-Prez, and L. P. Kaelbling, "FFRob: An efficient heuristic for task and motion planning," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2014, pp. 179–195.

[16] A. Hertle, C. Dornhege, T. Keller, and B. Nebel, "Planning with semantic attachments: An object-oriented view," in *Proc. Eur. Conf. Artif. Intell.*, 2010, vol. 242, pp. 402–407.

[17] E. Erdem, E. Aker, and V. Patoglu, "Answer set programming for collaborative housekeeping robotics: Representation, reasoning, and execution," *Intell. Service Robot.*, vol. 5, no. 4, pp. 275–291, 2012.

[18] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 4575–4581.

[19] N. Shah and S. Srivastava, "Anytime integrated task and motion policies for stochastic environments," in *Proc. 7th ICAPS Workshop Planning Robot.*, Feb. 2018.

[20] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *Int. J. Robot. Res.*, vol. 28, no. 1, pp. 104–126, Jan. 2009.

[21] H. Ma and S. Koenig, "Optimal target assignment and path finding for teams of agents," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2016, pp. 1144–1152.

[22] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 5628–5635.