Hypergraph-Based Multi-robot Task and Motion Planning

James Motes, Tan Chen, *Member, IEEE*, Timothy Bretl, *Senior Member, IEEE*, Marco Morales Aguirre, and Nancy M. Amato, *Fellow, IEEE*

***Abstract*—In this article, we present a multi-robot task and motion planning method that, when applied to the rearrangement of objects by manipulators, results in solution times up to three orders of magnitude faster than the existing methods and successfully plans for problems with up to 20 objects, more than three times as many objects as comparable methods. We achieve this improvement by decomposing the planning space to consider manipulators alone, objects, and manipulators holding objects. We represent this decomposition with a hypergraph where vertices are decomposed elements of the planning spaces and hyperarcs are transitions between elements. The existing methods use graph-based representations where vertices are full composite spaces and edges are transitions between these. Using the hypergraph reduces the representation size of the planning space for multimanipulator object rearrangement, the number of hypergraph vertices scales linearly with the number of either robots or objects, while the number of hyperarcs scales quadratically with the number of robots and linearly with the number of objects. In contrast, the number of vertices and edges in graph-based representations scales exponentially in the number of robots and objects. We show that similar gains can be achieved for other multi-robot task and motion planning problems.**

***Index Terms*—Cooperating robots, motion and path planning, multi-robot systems, task planning.**

I. Introduction

THE use of autonomous robotic systems is rapidly increasing. This can be seen in warehouse management, manufacturing, household chores, healthcare, etc. In many of these settings, multi-robot systems are leveraged to complete tasks that a single robot cannot do through inter-robot cooperation and increase throughput by working in parallel. These improvements in effectiveness often require complex coordination for both task and motion planning.

Unfortunately, the planning space for these problems is very large. For a single robot, it is intractable in the general case to represent explicitly [1], [2], and it grows exponentially for the composite space of multi-robot systems. Task and motion planning adds additional dimensions to the tasks present.

For simpler problems, decoupling the planning space into individual robot state spaces results in faster planning times. However, decoupled approaches cannot handle high levels of coordination. Thus, the existing multi-robot task and motion planning (MR-TMP) approaches primarily plan in the composite space and cannot efficiently plan for large, complex multi-robot systems.

Some problems, such as payload transportation, model tasks abstractly and often ignore the physical dimensions of the tasks during planning [3], [4], [5]. Other problems, especially those involving object manipulation, must consider physical constraints [6], [7], [8], [9], [10]. Here, the coordination required limits the use of decoupled methods, and the planning space grows too large to apply composite methods.

Hybrid search techniques that balance the strengths and weaknesses of coupled and decoupled approaches have shown promise in recent multi-robot motion planning (MRMP) methods [11], [12]. However, before our work, hybrid search techniques were not used for problems requiring more complex coordination, such as MR-TMP.

In this article, we present the *decomposable state-space hypergraph* (DaSH) method, a general multi-robot planning framework based on a hypergraph representation that can both incorporate the existing composite and decoupled multi-robot methods and enable new hybrid methods. We present a novel hybrid search technique based on the DaSH representation that captures the coordination and complexity in MR-TMP problems while scaling to larger numbers of robots and tasks. Additionally, our method successfully plans for higher ratios of tasks to robots (10:1), which the existing methods, such as [10], struggle with.

Our approach supports varying levels of (de)coupling of the planning space at different problem stages. The degree of coupling should be tailored to the problem and its corresponding planning space and can change to fit different problem stages. We

Manuscript received 13 April 2023; accepted 13 June 2023. Date of publication 24 August 2023; date of current version 4 October 2023. This work was supported in part by Foxconn Interconnect Technology (FIT) and in part by the Center for Networked Intelligent Components and Environments (C-NICE) at UIUC. The work of Marco Morales was supported by Academia Mexicana de Cultura A.C. This paper was recommended for publication by Associate Editor S. L. Smith and Editor W. Burgard 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).

Tan Chen is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA (e-mail: tanchen@mtu.edu).

Timothy Bretl is with the Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Champaign, IL 61820 USA (e-mail: tbretl@illinois.edu).

Marco Morales Aguirre is with the Parasol Lab, Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL 61820 USA, and also with the Department of Computer Science, Instituto Tecnológico Auónomo de México, Mexico City 01080, Mexico (e-mail: moralesa@illinois.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3297011.

Digital Object Identifier 10.1109/TRO.2023.3297011

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

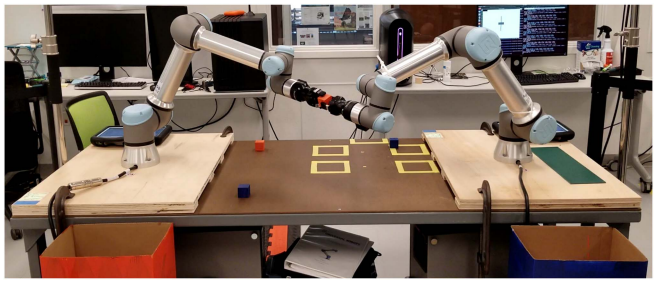
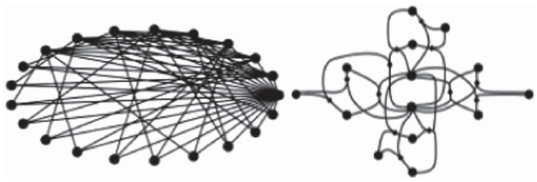


Fig. 1. We demonstrate our proposed approach for the multimanipulator sorting problems with a pair of UR5e manipulators. The robots must sort the blocks into bins of the corresponding color. This figure captures the moment in which the robots are performing a handoff. A full video of the sorting demo can be found at https://youtu.be/MSyeYXu0Xzs.



(a)

Robots	Objects	Representation	Vertices	Transitions
2	4	Hypergraph	14	24
		Graph	21	120
	8	Hypergraph	26	48
		Graph	73	496
4	4	Hypergraph	24	80
		Graph	209	8672
	8	Hypergraph	44	160
		Graph	3393	213184

(b)

Fig. 2. Figures and table provide an illustration of the contrast in representation sizes for composite (graph) and decoupled/hyrbid (hypergraph) search spaces for multimanipulator problems. The details of the problem constraints can be found in Section VII. (a) Composite (graph) representation of the task space for 2 robots and 4 objects contains 21 vertices and 120/60 directed/bidirectional edges. The decoupled (hypergraph) representation for the same problem contains 14 vertices and 24/12 directed/bidirectional hyperarcs. (b) Number of vertices and directed transitions (edges/hyperarcs) are given for increasing problem sizes. The reduction in the size of the hypergraph representation is significant when increasing both robots and objects. The graph representation quickly becomes too large to use efficiently, while the hypergraph remains relatively small.

support this with hybrid composition representations for several planning spaces that capture varying degrees of (de)coupled spaces depending on the coordination required.

Take, for example, a multimanipulator problem involving a handoff between two robots, as depicted in Fig. 1. Different levels of coordination are required at different stages of planning. Robots operate mostly independently in their individual state spaces. The joint space of a single robot and an object is important while picking/placing the object or while carrying the object, and the joint space of both robots and an object is important while planning a handoff between the two robots.

These decoupled and hybrid composition representations often remain sparse as the number of robots and tasks (or objects) increase. In contrast, the size of pure composite representations scales exponentially with both the number of robots and tasks. A small illustration of the size difference is shown in Fig. 2. As a result, existing methods, which plan in the composite space, are forced to use implicit representations of the composite space and engineer heuristics, which make assumptions about the underlying problem structure. Our decoupled and hybrid representations can be explicitly constructed and leveraged into powerful heuristics informed directly by the representation leading to significant performance gains.

Additionally, prior multimanipulator methods, such as [10], struggle as the number of tasks (or objects) increases relative to the number of robots. Our proposed representation flips the scaling paradigm where tasks (or objects) are no longer the limiting factor. As a result, DaSH supports higher ratios of objects to robots, which is appropriate for manipulation tasks where a few robots should handle many objects.

We demonstrate the ability of the DaSH method to generalize too many MR-TMP problems, including MRMP and multimanipulator rearrangement. For the rearrangement planning problem, we illustrate how the number of vertices in the hypergraph scales linearly with both the number of robots and objects, while the number of hyperarcs scales quadratically with the number of robots and linearly with the number of objects (see Fig. 2) and demonstrates improved planning times on both physical (see Fig. 1) and virtual multi-robot systems [see Fig. 13(a)].

In summary, our contribution is given as follows.

1) A generalized definition of the multi-robot planning space for composite, decoupled, and hybrid approaches.
2) A novel hypergraph representation for varying coupled/decoupled spaces in multi-robot planning problems.
3) A general planning algorithm that exploits this hypergraph representation.
4) The application of this approach to the multi-robot motion planning space (MRMP-DaSH) and to the multimanipulator planning space (MM-DaSH).
5) A theoretical analysis of the size of the proposed hypergraph representation compared with the traditional graph representations for multimanipulator rearrangement problems.
6) An experimental evaluation of the planning algorithm applied to the multimanipulator rearrangement problem.

II. Background and Related Work

In this section, we give an overview of the background and related work for MR-TMP.

A. Motion Planning

The *degrees of freedom* (DOFs) of a robot fully parameterize its position. They may contain the robot's pose, orientation, joint angles, etc. A specification of DOF values for a robot defines a *configuration*. Motion planning considers a continuous state space comprised of the set of all robot configurations known as *configuration space* ($\mathcal{C}_{space}$) [13].

The motion planning problem is to find a continuous path from a start location to a goal location through the subset of $\mathcal{C}_{space}$ consisting of valid configurations called *free space*. $\mathcal{C}_{space}$ is usually intractable to represent explicitly [1], [2]. To handle the complexity of planning in $\mathcal{C}_{space}$, sampling-based motion planners, such as the probabilistic roadmap method (PRM) [14],

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

attempt to create a discretized approximation of the connectivity of the $\mathcal{C}_{space}$ known as a roadmap. A roadmap is a graph in which vertices are individual configurations and edges represent the transitions between a pair of configurations. Paths are found by searching over this roadmap.

B. Multi-robot Motion Planning

MRMP considers the composite $\mathcal{C}_{space}$ of the system. This composite space is defined as the Cartesian product of the $\mathcal{C}_{space}$ of each individual robot. This space grows exponentially with the number of robots.

There are three main approaches to MRMP. *Coupled* planners plan directly within the composite space [15], [16]. They struggle to plan for large numbers of robots and are often slow, but they are able to provide probabilistic completeness. *Decoupled* methods consider individual robot $\mathcal{C}_{space}$ to find a set of paths [15], [17]. They are typically faster but usually lack completeness and optimality guarantees. *Hybrid* methods, such as conflict-based search for motion planning (CBS-MP) [11], seek to leverage the strengths of both composite and decoupled planners. They often iteratively plan within individual robot $\mathcal{C}_{spaces}$ (or the composite space of a subgroup of the robots) while reconciling plans against each other. Hybrid methods usually achieve planning times similar to decoupled methods while offering the completeness and optimality guarantees of composite methods.

C. Task and Motion Planning

When we consider planning problems with objects that can be moved by robots, we enter the domain of task and motion planning [18]. If we model the state of the entire world (robots and objects) as a configuration space $\mathcal{W}$, we end up with a highly underactuated system. We only have direct control over the robot DOFs, while object sDOFs can only be changed indirectly by the robot acting upon them. Additionally, the actions the robot takes affect motion feasibility. For example, a valid motion for a robot not holding an object may no longer be valid if the robot is grasping an object, and the current placement of the objects affects the feasibility of motions.

One approach is to treat each combination of the robot and held object as a unique robot and each placement of objects as a new environment. This is sometimes called multimodal motion planning [19], [20], [21], although this language is less intuitive than task and motion planning and can be confused for motion planning with different modes of locomotion.

Instead, we define a *task space* $\mathcal{T}$, where element $T_i \in \mathcal{T}$ defines which object the robot is holding (or none), the corresponding grasp constraints, and valid poses for all ungrasped objects. This defines a new configuration space $\mathcal{W}_i$ for the system. Details of this are provided in Section III.

A system can switch between task space elements by applying an action that changes the constraints of valid motion. For example, performing a pick action on an object removes the constraint that the object is on a stable surface and adds the constraint that the object is in a stable grasp for the robot.

A system can only switch between task space elements at configurations that are valid within the elements it is switching between. In practice, these are often stable grasps of objects that are sitting at stable poses on some surface in the environment or handoffs at the intersection of stable grasp poses for a set of robots [8]. These configurations satisfy the constraints of the task space element defining the robot grasping the object and the task space element where the object is located at that stable pose. Depending on the direction of the switch, these configurations can represent the moment of a pick or place action. Finding these transition configurations is considered the most challenging part of multimodal planning [18].

It is often best to view classic task and motion planning problems as hybrid discrete–continuous search problems [18]. A solution consists of a finite and discrete sequence of task space elements (e.g., which object to grasp) with continuous constraint parameters (poses and grasps of objects), and continuous motion paths within the configuration space W_i of each task state T_i to a configuration in the intersection with the subsequent task space element's configuration space.

D. Multi-robot Task and Motion Planning

In multi-robot systems, a task space element consists of the same types of constraints (object poses and grasp constraints in manipulation problems) as in single-robot systems. However, in this domain, there is a combinatorial expansion in the number of task space elements available to the system as both the number of robots and objects grow [22].

The three classes of MRMP (coupled, decoupled, and hybrid) apply here as well. Most task and motion planning problems require too much coordination for decoupled methods to be useful. Although some of our prior work has explored hybrid search techniques in the MR-TMP domain, the approach does not extend to manipulation problems [3]. The rest of this section discusses coupled approaches.

Previous work in the area has had two main focuses to handle the space complexity: developing representations and developing heuristics. Umar et al. [23] use a shared manipulator workspace to build a shared space graph to reason over multi-robot cooperation and adapt a path planning heuristic for multi-manipulator planning. Assumptions about reachability and the number of robots required to securely grasp an object are used in [8] to build a condensed graph representation. This representation is integrated into later work with an MRMP method [24] to create a full multi-robot multimodal planner [9]. This planner introduces a high-level object-mode graph to represent valid transitions among pick, place, and handoff configurations. This graph is used as a heuristic to guide the multimodal motion planning through sequences of modes that form a valid solution. All three of these methods [8], [9], [23] only plan for a single object.

In [10], the work in [9] is expanded to account for multiple objects. The single-object-mode graph in [9] is replaced with an object-centric-mode graph where vertices correspond to manipulators and stable surfaces. Instead of searching over object (or task state) switches for a single object, a multiagent pathfinding (MAPF) technique is proposed to find nonconflicting individual

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

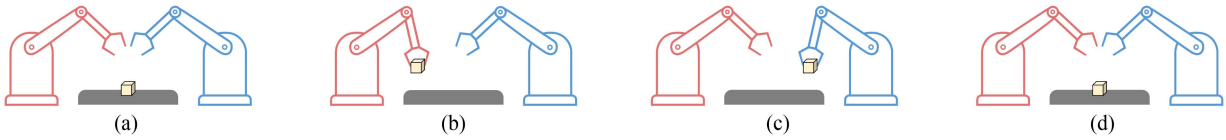


Fig. 3. (a) Figure shows the starting task space element of the system where the object is at rest at a stable pose on the table. (b) *Pick* action results in a switch to the middle left task space element where the red robot is grasping the object. (c) *Handoff* action creates another switch transferring the object from the red robot to the blue robot. (d) Finally, a *place* action returns the system to a stable pose task space element.

object-mode sequences over the available manipulators and stable surfaces. The transfer of multiple objects at once complicates the reasoning over both the sequence of modes and the motion planning. The authors simplify the problem by considering synchronized actions for the set of manipulators in the problem. This allows a single step in the MAPF solution over the object centric-mode graph to both indicate a set of actions for the robots to perform and define a more constrained MRMP problem.

Using MAPF over the object-centric-mode graph as a heuristic, the synchronized multiarm rearrangement (SMART) is able to efficiently plan for up to nine robots [10]. The heuristic is formulated as an MAPF problem over the object-centric-mode graph. The method is biased to greedily move the object-centric-mode state forward along the corresponding MAPF solution. When the MAPF solution is simple and the paths for individual objects are not likely to use the same robots at the same time, this greedy heuristic performs exceptionally well. However, when the paths for objects are likely to conflict, or the number of objects increases relative to the number of robots (increasing the density of the MAPF problem), this heuristic becomes limiting. This is reflected in [10] as no result is shown for more than four objects. We directly compare our method against SMART [10] in later sections.

Recent work, such as the work of Pan et al. [25], has addressed more complicated manipulation tasks, such as tower stacking and obstructing obstacles, although they still struggle to scale the number of objects relative to the number of robots with the highest ratio coming with six objects for two robots. The evaluation of the method's ability to handle increasing numbers of robots only considers very simple task problems where each robot has a very clear correct role [25]. We show the ability to solve similar problems in Section VIII and obtain higher ratios of objects to robots in our more general experiments.

All of these methods focus on a single short-term task involving a small set of obstacles. In contrast, Hartmann et al. [26] present a long-horizon MR-TMP construction method, which decomposes problems with up to 113 objects into subproblems each focused around a single object. Methods, such as the one presented here and [25], can be integrated into the long-horizon framework of [26] to address more complicated subtasks within long-horizon problems. We do not address this integration in this work.

E. Directed Hypergraphs and Hyperpaths

A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a generalization of a graph, where $\mathcal{V} = \{v_0, v_1, \ldots, v_n\}$ is the set of vertices, and

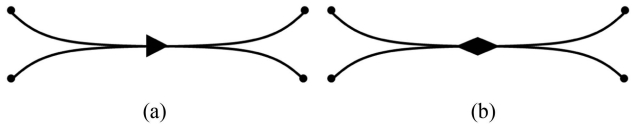


Fig. 4. (a) Directed hyperarc represents a transition from the vertices in the *tail set* on the left to the vertices in the *head set* on the right. (b) Bidirectional hyperarc indicates that the transition can happen in either direction. It is essentially a pair of directed hyperarcs with opposite tail and head sets.

$\mathcal{E} = \{E_0, E_1, \ldots, E_m\}$, with $E_i \subset \mathcal{V}$ for $i \in [0, m]$, is the set of hyperedges. Unlike edges in a regular graph, a hyperedge $E \in \mathcal{E}$ is not restricted to pairs of vertices. In this work, we consider directed hypergraphs, as described in [27].

1) Directed Hypergraph: A directed hypergraph has directed hyperedges, or *hyperarcs*, so that a hyperarc E_i has both a head set $\texttt{Head}(E_i) \subseteq \mathcal{V}$ and a tail set of vertices $\texttt{Tail}(E_i) \subseteq \mathcal{V}$. A visual depiction of directed hyperarcs is given in Fig. 4, and a depiction of a directed hypergraph is given in Fig. 5(a). In this article, all hypergraphs discussed will be directed hypergraphs.

2) Directed Hyperpath: We use directed hyperpaths, as defined in [27]. A *path* in a hypergraph is a sequence of vertices and hyperarcs $v_0, E_0, v_1, E_1, \ldots, v_{\text{final}}$ such that $v_i \in \texttt{Tail}(E_i)$ and $v_{i+1} \in \texttt{Head}(E_i)$ for all vertices and edges in the path [see Fig. 5(b)]. A path is *simple* if all hyperarcs are used at most once.

A *hyperpath* Π_{st} is a minimal hypergraph $\mathcal{H}_\Pi = (\mathcal{V}_\Pi, \mathcal{E}_\Pi)$ such that

$$\mathcal{E}_\Pi \subseteq \mathcal{E} \tag{1}$$

$$s, t \in \mathcal{V}_\Pi = \bigcup_{E_i \in \mathcal{E}_\Pi} E_i \subseteq \mathcal{V} \tag{2}$$

$$x \in \mathcal{V}_\Pi \Rightarrow x \text{ is connected to } s \text{ in } \Pi_{st} \text{ through a cycle-free simple path.} \tag{3}$$

The set of hyperarcs $\mathcal{E}_\Pi$ in the hyperpath Π_{st} must be in the original hypergraph $\mathcal{H}$ (1). The set of vertices $\mathcal{V}_\Pi$ consists of all vertices incident to a hyperarc $E_i \in \mathcal{E}_\Pi$, and the source s and target t must be included in $\mathcal{V}_\Pi$ (2). Every vertex must be connected to the source s in the hyperpath through a cycle-free simple path (3). Thus, the hyperarc from $\{v_0, v_1\}$ to $\{v_2\}$ in Fig. 5 cannot be included in a hyperpath from source v_0 as v_1 is not connected to v_0 via a cycle-free simple path.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

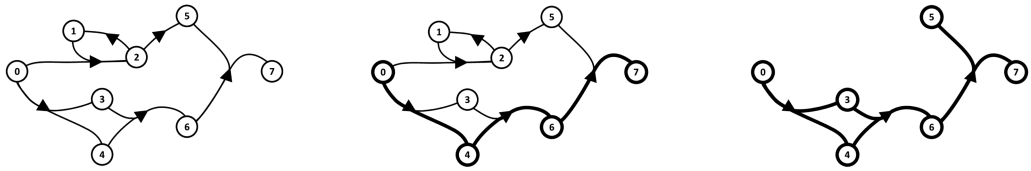


Fig. 5. (a) Directed hypergraph consists of vertices and directed hyperarcs. (b) Simple path in a directed hypergraph from v_0 to v_7. No hyperarc is used more than once. (c) Directed hyperpath from v_0 to v_7. Every vertex incident a hyperarc in the hyperpath is included. All vertices are connected to the source v_0 by a cycle-free simple path.

III. Problem Definition

In this section, we define the MR-TMP problem and the corresponding planning space for a set of moveable bodies $\mathcal{B}$. Our definition generalizes the coupled, decoupled, and hybrid representations used in multi-robot planning. In Sections V and VI, we show how previous MRMP and MR-TMP methods fit into this definition, and how it enables the hybrid MR-TMP approach presented in Section IV. In the following paragraphs, we explain this definition in the context of MRMP and multimanipulator rearrangement planning.

A. Planning Space

We consider a task space $\mathcal{T}$ consisting of elements $T_i = (B_i, \mathcal{W}_i, C_i)$, where $B_i \subseteq \mathcal{B}$ is a subset of the moveable bodies, $\mathcal{W}_i$ is the $\mathcal{C}_{space}$ for B_i, and C_i is a set of constraints, which defines validity in $\mathcal{W}_i$. Constraints in C_i may only apply to the moveable bodies in B_i; thus, the validity of a configuration $w \in \mathcal{W}_i$ cannot depend on a moveable body $b \notin B_i$.

Task space elements can be further decomposed into additional sets of elements or combined with other elements. We denote the decomposition of a task element $T_i = (B_i, \mathcal{W}_i, C_i)$ into a set of task space elements as $D(T_i) \subseteq \mathcal{T}$. A task space element $T_j = (B_j, \mathcal{W}_j, C_j)$ belongs to $D(T_i)$ if $B_j \subseteq B_i$, $\mathcal{W}_j$ is a subspace of $\mathcal{W}_i$, and $C_j \subseteq C_i$. This resembles a power set, although $\mathcal{W}_j$ is a subspace of $\mathcal{W}_i$ and the constraints in C_j only apply to robots in B_j, etc.

We denote the combination of task space elements $T_j = (B_j, \mathcal{W}_j, C_j)$ and $T_k = (B_k, \mathcal{W}_k, C_k)$ to create $T_i = (B_i, \mathcal{W}_i, C_i)$ as $T_j + T_k = T_i$. In the resulting element T_i, $B_i = B_j \cup B_k$, and W_i is the corresponding $\mathcal{C}_{space}$. We assume that no moveable bodies may ever be in collision with each other, so $C_i = C_j \cup C_k \cup \{\text{no collision between } b_p, b_q \in B_i\}$.

Each MR-TMP problem has a set of *admissible* task space elements $\mathcal{T}^* \subseteq \mathcal{T}$ such that, for any $T_i = (B_i, \mathcal{W}_i, C_i) \in \mathcal{T}^*$, the moveable bodies $B_i = \mathcal{B}$ and constraints C_i define an obtainable set of configurations in $\mathcal{W}_i$ for the system (i.e., the system designer considers these acceptable configurations).

The MRMP problem for a group of robots $\mathcal{B}$ is often defined to consider the composite configuration space $\mathcal{C}_0 \times \cdots \times \mathcal{C}_{|\mathcal{B}|-1}$ such that $\mathcal{C}_i$ is the configuration space for robot $b_i \in \mathcal{B}$. A configuration is valid if no robot is in collision with any obstacle in the environment and no two robots are in collision with each other. An MRMP solution consists of a continuous valid path from some start point $w_{\texttt{start}}$ to some goal point $w_{\texttt{goal}}$, where $w_{\texttt{start}}, w_{\texttt{goal}} \in \mathcal{C}_0 \times \cdots \times \mathcal{C}_{|\mathcal{B}|-1}$.

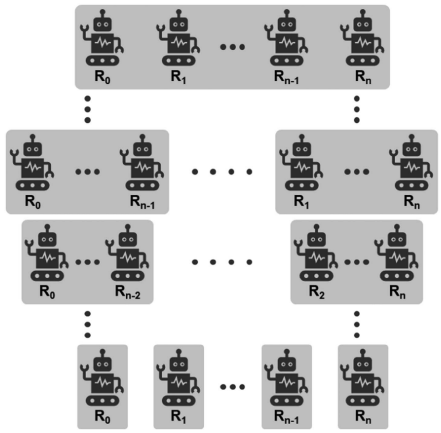


Fig. 6. Figure depicts the task space for MRMP. The top row indicates the admissible element containing all robots and collision constraints. This is the space composite methods plan. The last row represents a set of fully decoupled elements, each containing a single robot. All other combinations of robots with their collision constraints also exist in the task space (in the intermediate rows).

The task space $\mathcal{T}_{\texttt{MRMP}}$ for the MRMP problem contains a single *admissible* element. In the MRMP problem, $\mathcal{T}^* = \{T_{\texttt{COMPLETE}} = (B_{\texttt{COMPLETE}}, \mathcal{W}_{\texttt{COMPLETE}}, C_{\texttt{COMPLETE}})\}$ such that $C_{\texttt{COMPLETE}}$ contains all the MRMP constraints, $B_{\texttt{COMPLETE}} = \mathcal{B}$, and $\mathcal{W}_{\texttt{COMPLETE}} = \mathcal{C}_0 \times \cdots \times \mathcal{C}_{|\mathcal{B}|-1}$. Composite MRMP methods plan motions directly within $T_{\texttt{COMPLETE}}$.

The full set of elements in $\mathcal{T}_{\texttt{MRMP}}$ can be obtained from the decomposition of $T_{\texttt{COMPLETE}}$ such that $\mathcal{T}_{\texttt{MRMP}} = D(T_{\texttt{COMPLETE}})$, as shown in Fig. 6. Fully decoupled MRMP methods, which plan motion separately for each robot $b_j \in \mathcal{B}$, plan within task elements $T_j = (B_j, \mathcal{W}_j, C_j)$, where $B_j = \{b_j\}$, $\mathcal{W}_j = \mathcal{C}_j$, and C_j defines validity for b_j independent of other robots. Hybrid MRMP methods move between different levels of task space element decomposition and combination. Both decoupled and hybrid MRMP methods construct a solution in $T_{\texttt{COMPLETE}}$ by combining their solutions with other elements. This often requires the reenforcement of the inter-robot collision constraints relaxed when decomposing task space elements.

More generally, MR-TMP problems may have several admissible task space elements. For example, an object manipulation problem may have an admissible element for every combination of robot–object grasps. Furthermore, the start and goal of a problem may belong to different task space elements. A valid motion path from the start to the goal must then *transition* between task space elements.

A path may transition between a pair of elements T_i and T_j at a transition configuration w, which is valid in $T_i + T_j$ so long

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

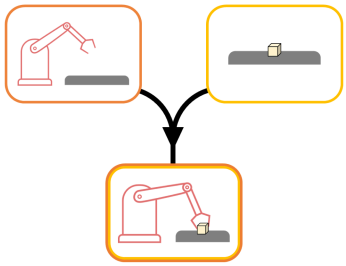


Fig. 7. Hyperarc captures the grasp of an object. The transition requires both the task space element for the robot not holding an object and the object not holding and resting on a stable surface. These together form the tail set. The transition moves the robot and object into a new task space element where the robot is now grasping the object.

as T_i and T_j contain the same moveable bodies $B_i = B_j$. An MR-TMP solution, thus, consists of both a sequence of elements in $\mathcal{T}^*$ and a valid continuous motion path through them.

In object manipulation problems, $\mathcal{T}^* \subseteq \mathcal{T}_{\texttt{MANIP}}$ contains elements, which denote which robot is grasping which object. These often resemble the task space elements, as depicted in Fig. 3. Most existing multimanipulator planning methods are composite approaches that plan only in task space elements in $\mathcal{T}^*$. Transitions between task space elements (e.g., mode switches in [10] and [18]) correspond to pick, place, and handoff actions. Solutions consist of a sequence of these transitions between task space elements in $\mathcal{T}^*$ and valid paths within each task space element between transition configurations.

True decoupled approaches, which plan with elements in the decompositions of admissible elements, are likely to fail to find a plan for manipulation problems due to the transition requirement of two elements T_i and T_j containing the same moveable bodies $B_i = B_j$. For example, elements $T_i = (B_i, \mathcal{W}_i, C_i)$ and $T_j = (B_j, \mathcal{W}_j, C_j)$, where B_i contains only robot r_i and B_j contains only object o_j, cannot transition to an element $T_k = (B_k, \mathcal{W}_k, C_k)$, where $B_k = \{r_i, o_j\}$ in a decoupled approach; thus, r_i cannot grasp o_j. Considering an element, which contains both r_i and o_j before the grasp constraint, is applied instead of T_i and T_j address this issue; however, when r_i needs to manipulate another object or o_j needs to be manipulated by another robot, the same issues arise. This quickly forces decoupled approaches to only consider the admissible elements to find a feasible solution, which is equivalent to composite approaches.

However, hybrid methods can effectively plan with decoupled nonadmissible elements. These approaches, which can move between different levels of task space element decomposition and combination, can transition between $T_i + T_j = T_l$ and T_k so long as $B_i \cup B_j = B_k$ and there exists a valid configuration in both $\mathcal{W}_l$ and $\mathcal{W}_k$ (see Fig. 7). Thus, transitions can occur between sets of task space elements so long as there exists a transition between the combination of elements on either side of the transition. Just as in MRMP, hybrid methods must ensure that a solution constructed in elements decomposed from elements in $\mathcal{T}^*$ is a valid solution in the elements in $\mathcal{T}^*$. This article presents a hybrid approach to MR-TMP.

Algorithm 1: High-Level Approach.

```
1: Procedure APPROACH(Task and Motion Planning Problem)
2:   S_optimisitic, S_best ← ∅
3:   while not converged do
4:     H ← ExpandRepresentation(H)
5:     while S_best not optimal for H do
6:       S_optimisitic ← ComputeTaskPlan(H)
7:       S_complete ← ResolveConflicts(H, S_optimistic)
8:       if S_complete.cost < S_best.cost then
9:         S_best ← S_complete
10:      if earlyTermination and S_best ≠ ∅ then
11:        return S_best
12:  return S_best
```

IV. Method

We propose the DaSH method, a general MR-TMP method based on the problem formulation, as presented in Section III. We first present an overview of the method, followed by a detailed description of the representation and construction process. We then define the task search and motion conflict resolution stage. Finally, we discuss several variants of the method and their theoretical properties.

A. Overview

Our approach consists of three stages: representation construction, task planning, and conflict resolution.

The DaSH method first builds the representation layers and then queries them to generate task plans. We introduce a hierarchy of hypergraph-based representation layers, which capture increasing levels of information. The highest layer contains the task space and transitions between task space elements. The middle layer encodes motion feasibility within the constrained configuration spaces in the task space elements. The final layer represents the search itself and includes the transition history of potential solutions.

Task plans are computed by finding a hyperpath through this final representation layer. This includes continuous motion paths within different task space elements. Thus, a final conflict resolution stage ensures that there are no interpath collisions between different task space elements. A solution then consists of a sequence of admissible task space elements and a valid motion path through them.

We include two variants in Algorithm 1. An `earlyTermination` option can be used to return the first found solution (line 11). This does not offer any optimality guarantees, although we will show later that it can be probabilistically complete.

For an asymptotically optimal solution, we first continuously expand the representation on line 4. Then, we use a hybrid approach inspired by the work in [4] to find the optimal solution for the current representation (lines 5–11) by iteratively computing and validating task plans. Additionally, at anytime, behavior can be gained by returning a valid $\mathcal{S}_{\texttt{best}}$ before convergence. Details

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

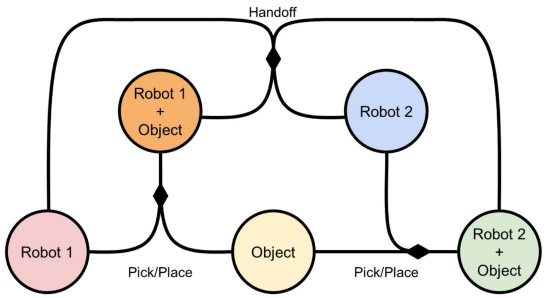


Fig. 8. This image depicts the task space hypergraph for a multimanipulator problem with two robots and one-object system. The vertices represent the set of task space elements obtainable in the system. The hyperarcs encode the set of allowed transitions available by applying *pick, place,* and *handoff* actions. The set of admissible task space elements that can combine to create is captured in Fig. 3, and the colors generally correlate. The red vertex represents Robot 1 not holding any object. The blue vertex represents Robot 2 not holding any object. The yellow vertex represents the object in a stable pose. The combination of these three vertices creates the admissible task space element, as depicted in Fig. 3(a). The hyperarc from the red and yellow vertices to the orange vertex denotes a pick (or place) action and includes only the moveable bodies involved. The hyperarc from the orange and blue vertices to the red and green vertices encodes a handoff action (as the one depicted in Fig. 7), and the hyperarc from the green vertex to the yellow and blue vertices encodes a place (or pick) action. This sequence is the same as the sequence of admissible task space elements, as depicted in Fig. 3, but only the moveable bodies that experience change are involved in the hypergraph transitions.

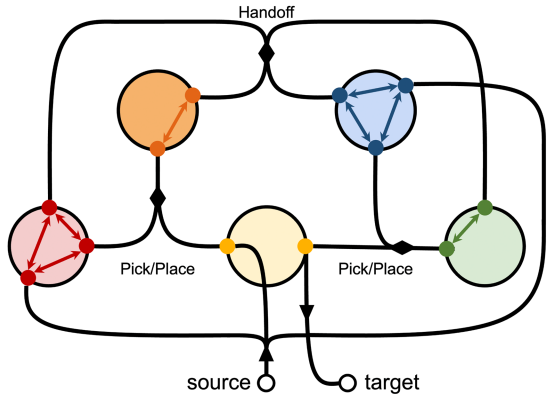


Fig. 9. This motion hypergraph is an augmentation of the multimanipulator task space hypergraph in Fig. 8. Transition hyperarc paths are planned for each transition hyperarc in the task space hypergraph. The start and end configurations of the transition paths are represented as colored vertices within the corresponding task space elements. Move paths are planned between transition start/end configurations within the same task space element and represented as (colored) edges to encode movement within that task space element. Note that there are no internal edges within the yellow object task space element as it is not actuated.

of these variations are discussed in Section IV-F. Probabilistic completeness and asymptotic optimality are both dependent upon the underlying construction and search methods used in each stage.

B. Task Space Hypergraph

As defined in Section III, task planning can be represented by a *task space* $\mathcal{T}$ where each task space element $T_i = (B_i, \mathcal{W}_i, C_i) \in \mathcal{T}$ includes a set of moveable bodies $B_i \subseteq \mathcal{B}$, the configuration space $\mathcal{W}_i$ for B_i, and a set of constraints C_i defining validity in $\mathcal{W}_i$. Task space elements can be decomposed or combined, as defined in Section III. Additionally, transitions can occur between sets of task space elements so long as all moveable bodies present on one side of the transition are present on the other side of the transition.

We propose a *task space hypergraph* $\mathcal{H}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ to model the set of task space elements and their relationships (see Fig. 8). Each vertex $v_\mathcal{T} =< T_i >\in \mathcal{V}_\mathcal{T}$ encodes a task space element $T_i \in \mathcal{T}$. Hyperarcs $E_\mathcal{T} =< \texttt{Tail}, \texttt{Head} >\in \mathcal{E}_\mathcal{T}$ encode either a *composition* or *transition* relationship from a tail set of task space elements to a head set of task space elements (see Fig. 7).

Composition hyperarcs $E_\mathcal{T}^{\texttt{comp}} \in \mathcal{E}_\mathcal{T}$ represent a change in the compositions of moveable bodies, their configuration spaces, and constraints. These may correspond to a decomposition, combination, or a mix of the two.

Transition hyperarcs $E_\mathcal{T}^{\texttt{trans}}$ represent a transition between the configuration spaces and/or constraints for a set of moveable bodies. For both types, the set of elements in the tail must be independent (i.e., no overlapping moveable bodies). Similarly, the set of elements in the head set must be independent.

A set of *composition rules* may be used to restrict the allowable decompositions and combinations of task space elements to reduce the size of the task space. For example, a common choice in MRMP is to only consider task space elements that contain a single robot. A multimanipulator planning algorithm may consider only task space elements that contain only robots and objects involved in direct manipulation with each other (e.g., a robot grasping an object). Transitions then occur between sets of task space elements.

Similarly, constraints on what transitions are feasible between sets of task space elements can restrict arbitrary transitions between configuration spaces or constraint sets for a set of moveable bodies. These constraints along with the composition rules are packaged as *allowed transitions* A and implicitly define the set of obtainable vertices and hyperarcs within a task space hypergraph.

Algorithm 2 defines how the task space hypergraph $\mathcal{H}_\mathcal{T}$ can be constructed from the initial task space element $T_{\texttt{init}}$ and the set of allowed transitions A. The direction of expansion is implemented in the `ExpandHypergraph` function and may be configured to a breadth first search (BFS), depth first search (DFS), or heuristic-guided search. An `earlyQuit` option may be included if a single task solution is sufficient.

C. Motion Hypergraph

The task space hypergraph considers only task space conditions and does not consider the motion feasibility of transitions. We build a *motion hypergraph* $\mathcal{H}_\mathcal{M} = (\mathcal{V}_\mathcal{M}, \mathcal{E}_\mathcal{M})$ to expand the task space information in the task space hypergraph to include motion feasibility. This motion hypergraph is a sampled representation of possible transitions through the MR-TMP space.

1) Motion Vertices: The vertices in the task space hypergraph contain task space constraints, but they do not have any DOF values to represent an explicit configuration within the configuration space $\mathcal{W}_i$ of the corresponding task space element T_i. A motion vertex $v_\mathcal{M} =< v_\mathcal{T}, q >\in \mathcal{V}_\mathcal{M}$ contains both a task space hypergraph vertex $v_\mathcal{T} \in \mathcal{V}_\mathcal{T}$ (corresponding to a task space

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

Algorithm 2: Task Space Hypergraph Construction.

1: **Procedure** BUILD $\mathcal{H}_\mathcal{T}(T_{\texttt{init}}$, Allowed Transitions A, `earlyQuit`)
2: $D(T_{\texttt{init}}) \leftarrow \texttt{Decompose}(T_{\texttt{init}}, A)$
3: $\mathcal{H}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T}) \leftarrow \texttt{Initialize}(D(T_{\texttt{init}}), A)$
4: **while** `true` **do**
5: $\mathcal{V}_{\texttt{new}}, \mathcal{E}_{\texttt{new}} \leftarrow \texttt{ExpandHypergraph}(\mathcal{H}_\mathcal{T}, A)$
6: **if** $\mathcal{E}_{\texttt{new}} == \emptyset$ **then**
7: break
8: $\mathcal{V}_\mathcal{T} \leftarrow \mathcal{V}_\mathcal{T} \cup \mathcal{V}_{\texttt{new}}; \mathcal{E}_\mathcal{T} \leftarrow \mathcal{E}_\mathcal{T} \cup \mathcal{E}_{\texttt{new}}$
9: **if** `earlyQuit` and $\texttt{ContainsSolution}(\mathcal{H}_\mathcal{T})$ **then**
10: break
11: **return** $\mathcal{H}_\mathcal{T}$

element $T_i \in \mathcal{T}$) and an explicit configuration $q \in \mathcal{W}_i$. As such, each $v_\mathcal{T} \in \mathcal{V}_\mathcal{T}$ may map to many motion vertices in $\mathcal{V}_\mathcal{M}$.

2) Motion Hyperarcs: There are three kinds of hyperarcs in the motion hypergraph: *composition, transition, and move.*

A *composition hyperarc* simply indicates that the planning space composition has changed.

A *transition hyperarc* $E^{\text{trans}}_\mathcal{M}$ encodes a set of feasible motions for performing a task space transition. The tail and head sets of $E^{\text{trans}}_\mathcal{M}$ may lie in different task space elements (or combinations of task space elements).

A *move hyperarc* $E^{\text{move}}_\mathcal{M}$ encodes a motion path between two motion vertices within the same task space element. The tail and head sets each contains only a single vertex, $v^{\texttt{tail}}_\mathcal{M}$ and $v^{\texttt{head}}_\mathcal{M}$, respectively, and each has the same corresponding task space vertex $v^{\texttt{tail}}_\mathcal{M}.v_\mathcal{T} = v^{\texttt{head}}_\mathcal{M}.v_\mathcal{T}$. They do not correspond to a transition in the task space hypergraph.

Each motion hyperarc $E_\mathcal{M} =< \texttt{Tail}, \texttt{Head}, E_\mathcal{T}, \pi >$ consists of the standard tail and head vertex sets, in addition to a corresponding task space hyperarc $E_\mathcal{T} \in \mathcal{E}_\mathcal{T}$, and a path π from the motion vertices $v_\mathcal{M} \in E_\mathcal{M}.\texttt{Tail}$ to the motion vertices $v_\mathcal{M} \in E_\mathcal{M}.\texttt{Head}$. The task space hyperarc $E_\mathcal{T} = \texttt{NULL}$ for motion hyperarcs. The path for composition and transition motion hyperarcs $E^{\text{comp}}_\mathcal{M}.\pi$ or $E^{\text{trans}}_\mathcal{M}.\pi$ must contain a configuration valid in both the constrained configuration spaces produced by combining all task space elements in the tail set $E_\mathcal{M}.\texttt{Tail}$ and the head set $E_\mathcal{M}.\texttt{Head}$. The path should be valid in the tail set $\mathcal{C}_{space}$ before the switch and valid in the head set $\mathcal{C}_{space}$ after. In the case of composition hyperarcs, the path is often a single configuration.

3) Composition Hyperarc Planning: A motion composition hyperarc $E^{\text{comp}}_\mathcal{M} \in \mathcal{E}_\mathcal{M}$ captures the composition change encoded in the task space composition hyperarc $E^{\text{comp}}_\mathcal{T} = E^{\text{comp}}_\mathcal{M}.E_\mathcal{T} \in \mathcal{E}_\mathcal{T}$. As the constraints do not change (outside the relaxation of collision avoidance between moveable bodies in different task space elements), $E^{\text{comp}}_\mathcal{M}.\pi$ can be any path in the constrained $\mathcal{C}_{space}$ induced by the combination of the elements in the tail (or head) set of $E^{\text{comp}}_\mathcal{T} \in \mathcal{E}_\mathcal{T}$. These paths may be a single configuration, which may be sampled, or encode a longer motion. Paths longer than a single configuration can be planned in the same manner as move hyperarcs operating in the combination of task space elements in the tail (or head) set of $E^{\text{comp}}_\mathcal{T} \in \mathcal{E}_\mathcal{T}$.

The end points of the path are decomposed along the task space elements present in the tail and head vertices in the task space hyperarc $E^{\text{comp}}_\mathcal{M}.E_\mathcal{T}$. These comprise the tail and head sets of $E^{\text{comp}}_\mathcal{M}$ and are included in $\mathcal{V}_\mathcal{M}$. $E^{\text{comp}}_\mathcal{M}$ is included in $\mathcal{E}_\mathcal{M}$, encoding a feasible path for performing the composition change.

4) Transition Hyperarc Planning: Generating transition hyperarcs often requires a specialized planner to capture feasible motions transitioning from the constraints of one set of task space elements to another. For example, pick, place, and handoff transitions can be computed using an inverse kinematics (IK)-based planner.

The $E^{\text{trans}}_\mathcal{M}.\pi$ plan can consist of a single configuration or a path. The start and end points of the path are decomposed along the task space elements present in the tail and head vertices in the task space hyperarc $E^{\text{trans}}_\mathcal{M}.E_\mathcal{T}$ in the same manner as the composition hyperarcs. The tail and head sets of $E^{\text{trans}}_\mathcal{M}$ are included in $\mathcal{V}_\mathcal{M}$. $E^{\text{trans}}_\mathcal{M}$ is included in $\mathcal{E}_\mathcal{M}$, encoding a feasible path for performing a task space transition.

5) Move Hyperarc Planning: Move hyperarcs encode motion feasibility within task space element configuration spaces $\mathcal{W}_{\texttt{move}}$ between the start and end points of transition hyperarcs. While any motion planner may be used to generate the path $E^{\texttt{move}}_\mathcal{M}.\pi$ between a pair of motion vertices v_i and v_j, within the same task space elements $v_i.v_\mathcal{T} = v_j.v_\mathcal{T}$, we chose to utilize a sampling-based approach. Within a particular task space element's configuration space $\mathcal{W}_{\texttt{move}}$, we grow a roadmap and connect the start and end points of transition hyperarcs within $\mathcal{W}_{\texttt{move}}$. Paths between these transition start and end points are found by querying the roadmap. Each $E^{\text{move}}_\mathcal{M} \in \mathcal{E}_\mathcal{M}$, thus, encodes motion feasibility between planned transitions in and out of a particular task substate.

6) Start and Goal: The initial configuration of the system $q \in \mathcal{W}_{\texttt{init}}$ can be decomposed into a set of motion vertices following the initial task state decomposition $D(T^{\texttt{init}}, A)$, where A is the set of allowed transitions. These motion start vertices can be connected to transition start points via move hyperarc planning. Goal constraints can be directly sampled to create motion goal vertices. These can also be connected to transition end points via move hyperarc planning.

A *virtual source* vertex $v^{\texttt{source}}_\mathcal{M}$ along with a virtual hyperarc $E_\mathcal{M} =< \{v^{\texttt{source}}_\mathcal{M}\}, \{\text{all start vertices}\}, \texttt{NULL}, \emptyset >$ connecting $v^{\texttt{source}}_\mathcal{M}$ to the set of start vertices, is used to capture the system start state. If multiple decompositions of the start state are included, then each complete set of start vertices would form the head of a hyperarc originating from the virtual start vertex.

A *virtual target* vertex $v^{\texttt{target}}_\mathcal{M}$ along with virtual hyperarcs of the form $E_\mathcal{M} =< \{\text{satisfying set of vertices}\}, \{v^{\texttt{target}}_\mathcal{M}\}, \texttt{NULL}, \emptyset >$ connecting unique sets of satisfying motion vertices to $v^{\texttt{source}}_\mathcal{M}$, is used to capture the necessary conditions to complete the task.

7) Construction: The construction of the motion hypergraph $\mathcal{H}_\mathcal{M}$ (see Algorithm 3) begins by initializing $\mathcal{H}_\mathcal{M}$ with the virtual source and target vertices $v^{\texttt{source}}_\mathcal{M}$ and $v^{\texttt{target}}_\mathcal{M}$ (line 3). The initial set of motion vertices $\mathcal{V}_{\texttt{init}}$ is computed by decomposing the initial configuration $q^{\texttt{init}}$ according to the decomposition provided in $D(T_{\texttt{init}})$ (line 4). These are added to $\mathcal{H}_\mathcal{M}$ and connected to $v^{\texttt{source}}_\mathcal{M}$ (lines 5 and 6). The same process is repeated for all valid goal configurations provided in Q^{goal} connecting

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

them to $v_{\mathcal{M}}^{\texttt{target}}$ (lines 7–10). A *vertex map* M_v tracks all motion instances of task space vertices (lines 11 and 13).

The main loop iterates first sampling transitions (and compositions), and then moves until the hypergraph contains a solution (line 14). Transitions are sampled by iterating over all task space hyperarcs $E_{\mathcal{T}} \in \mathcal{E}_{\mathcal{T}}$ (line 17), and, for some desired number of samples, calling the transition planner to find a feasible path to perform the transition (line 18).

Motion composition hyperarcs may also be sampled here, but due to the number of possibilities, it is often best to lazily discover the need for changing compositions. Section IV-D5 discusses this as a means of resolving interpath conflicts.

The start and end points of successful transitions (or compositions) are added as motion vertices to the hypergraph, and the transition (or compositions) path as a whole is added as a hyperarc (lines 24 and 25). The vertex map is updated with the new vertices on line 27.

Move hyperarcs are computed by iterating over all of the vertices in the task space hypergraph $v_{\mathcal{T}} \in \mathcal{V}_{\mathcal{T}}$ and attempting to connect motion vertices from the same task vertex (lines 29 and 30). A `Candidates` function on line 31 allows for a heuristic selection of which vertices to attempt to connect (the default is all other $v_{\mathcal{M}} \in M_v[v_{\mathcal{T}}]$). On line 32, the move planner $P_{\texttt{move}}$ attempts to compute a feasible motion plan between the two motion vertices in the corresponding subconfiguration space. Successful motion plans are saved as move hyperarcs on line 38.

The `singleShot` option may be used if only the task space hyperarcs for a specific solution are provided. This indicates that any failure to sample a hyperarc will prevent a solution from being contained within $\mathcal{H}_{\mathcal{M}}$, and lines 20 and 34 allow for early termination upon failure. If all hyperarc samples are successful, then the `singleShot` options allow for early return of motion hypergraph without mandating that it contain a solution (line 39).

D. Transition-Extended Hypergraph

Because both transition and composition hyperarcs in the motion hypergraph $\mathcal{H}_{\mathcal{M}}$ move in and out of different space compositions, planning directly over $\mathcal{H}_{\mathcal{M}}$ can result in a moveable body existing in multiple states simultaneously. In order to query a valid task plan, we must consider a *transition-extended* search space, which we represent with a *transition-extended hypergraph* $\mathcal{H}_{TE} = (\mathcal{V}_{TE}, \mathcal{E}_{TE})$ (see Fig. 10).

1) Transition-Extended Vertices: Each vertex $v_{\texttt{TE}} =< v_{\mathcal{M}}, \Pi_{v_{\texttt{TE}}^{\texttt{source}}, v_{\texttt{TE}}} > \in \mathcal{V}_{TE}$ is defined by a motion vertex $v_{\mathcal{M}} \in \mathcal{V}_{\mathcal{M}}$ and a *transition history* $\Pi_{v_{\texttt{TE}}^{\texttt{source}}, v_{\texttt{TE}}}$.

$\mathcal{H}_{TE}$ has a virtual source vertex $v_{\texttt{TE}}^{\texttt{source}} =< v_{\mathcal{M}}^{\texttt{source}}, \Pi_{v_{\texttt{TE}}}^{\texttt{source}}, v_{\texttt{TE}}^{\texttt{source}} >$ which contains a trivial transition history. The transition history of every other vertex $v_{\texttt{TE}} \in \mathcal{V}_{TE}/\{v_{\texttt{TE}}^{\texttt{source}}\}$ consists of all of the moves and transitions taken to reach $v_{\texttt{TE}}$ from $v_{\texttt{TE}}^{\texttt{source}}$. This corresponds to a hyperpath $\Pi_{v_{\texttt{TE}}^{\texttt{source}}, v_{\texttt{TE}}}$ in $\mathcal{H}_{TE}$ from $v_{\texttt{TE}}^{\texttt{source}}$ to $v_{\texttt{TE}}$. As such, each vertex $v_{\texttt{TE}}$ has a single incoming hyperarc as this incoming hyperarc is a part of its transition history.

A transition history $\Pi_{v_{\texttt{TE}}^{\texttt{source}}, v_{\texttt{TE}}}$ is valid if every vertex $v_{\texttt{TE}} \in \Pi_{v_{\texttt{TE}}^{\texttt{source}}, v_{\texttt{TE}}}$ contains at most one outgoing hyperarc. This indicates that the moveable bodies at a particular configuration are denoted by the vertices in the tail set of a hyperarc transition to at most one additional configuration at a time. Multiple outgoing hyperarcs from a single transition-extended vertex $v_{\texttt{TE}}$ indicate that the moveable bodies corresponding to $v_{\texttt{TE}}$ perform simultaneous motions and exist in more than one place at once. This is an invalid transition history.

Algorithm 3: Motion Hypergraph Construction.

1: **Procedure** BUILD $\mathcal{H}_{\mathcal{M}}$(Task space hypergraph $\mathcal{H}_{\mathcal{T}} = (\mathcal{V}_{\mathcal{T}}, \mathcal{E}_{\mathcal{T}})$, composition planner P, transition planner $P_{\texttt{trans}}$, move planner $P_{\texttt{move}}$, initial task state decomposition $D(T_{\texttt{init}})$, initial configuration $q_{\texttt{init}}$, goal task state decomposition $D(T_{\texttt{goal}})$, goal configurations $Q_{\texttt{goal}}$, `singleShot`)
2: Initialize motion hypergraph with start and goal configurations
3: $\mathcal{H}_{\mathcal{M}} \leftarrow (\mathcal{V}_{\mathcal{M}}, \mathcal{E}_{\mathcal{M}}) = (\{v_{\mathcal{M}}^{\texttt{source}}, v_{\mathcal{M}}^{\texttt{target}}\}, \emptyset)$
4: $\mathcal{V}_{\texttt{init}} \leftarrow \texttt{DecomposeCfg}(q^{\texttt{init}}, D(T_{\texttt{init}}))$
5: $\mathcal{V}_{\mathcal{M}} \leftarrow \mathcal{V}_{\mathcal{M}} \cup \mathcal{V}_{\texttt{init}}$
6: $\mathcal{E}_{\mathcal{M}} \leftarrow \mathcal{E}_{\mathcal{M}} \cup \{< \{v_{\mathcal{M}}^{\texttt{source}}\}, \mathcal{V}_{\texttt{init}}, \texttt{NULL}, \emptyset >\}$
7: **for all** $q^{\texttt{goal}} \in Q^{\texttt{goal}}$ **do**
8: $\mathcal{V}_{\texttt{goal}} \leftarrow \texttt{DecomposeCfg}(q^{\texttt{goal}}, D(T_{\texttt{goal}}))$
9: $\mathcal{V}_{\mathcal{M}} \leftarrow \mathcal{V}_{\mathcal{M}} \cup \mathcal{V}_{\texttt{goal}}$
10: $\mathcal{E}_{\mathcal{M}} \leftarrow \mathcal{E}_{\mathcal{M}} \cup \{\mathcal{V}_{\texttt{goal}}, < \{v_{\mathcal{M}}^{\texttt{target}}\}, \texttt{NULL}, \emptyset >\}$
11: vertex map $M_v \leftarrow \emptyset$
12: **for all** $v_{\mathcal{M}} \in \mathcal{V}_{\mathcal{M}}$ **do**
13: $M_v[v_{\mathcal{M}}.v_{\mathcal{T}}] \leftarrow M_v[v_{\mathcal{M}}.v_{\mathcal{T}}] \cup \{v_{\mathcal{M}}\}$
14: **while** does not $\texttt{ContainsSolution}(\mathcal{H}_{\mathcal{M}})$ **do**
15: Sample transitions (and compositions)
16: **for all** $E_{\mathcal{T}} \in \mathcal{E}_{\mathcal{T}}$ **do**
17: **for all** desired number of samples **do**
18: $E_{\mathcal{M}} \leftarrow P_{\texttt{trans}}(E_{\mathcal{T}})$ (or $P_{\texttt{comp}}(E_{\mathcal{T}})$)
19: **if** $E_{\mathcal{M}} == \emptyset$ **then**
20: **if** `singleShot` **then**
21: **return** $\mathcal{H}_{\mathcal{M}}$
22: **else**
23: continue
24: $\mathcal{V}_{\mathcal{M}} \leftarrow \mathcal{V}_{\mathcal{M}} \cup E_{\mathcal{M}}.\texttt{Tail} \cup E_{\mathcal{M}}.\texttt{Head}$
25: $\mathcal{E}_{\mathcal{M}} \leftarrow \mathcal{E}_{\mathcal{M}} \cup \{E_{\mathcal{M}}\}$
26: **for all** $v_{\mathcal{M}} \in E_{\mathcal{M}}.\texttt{Tail} \cup E_{\mathcal{M}}.\texttt{Head}$ **do**
27: $M_v[v_{\mathcal{M}}.v_{\mathcal{T}}] \leftarrow M_v[v_{\mathcal{M}}.v_{\mathcal{T}}] \cup \{v_{\mathcal{M}}\}$
28: Sample moves
29: **for all** $v_{\mathcal{T}} \in \mathcal{V}_{\mathcal{T}}$ **do**
30: **for all** $v_{g,i} \in M_v[v_{\mathcal{T}}]$ **do**
31: **for all** $v_{g,j} \in \texttt{Candidates}(v_{g,i}, M_v[v_{\mathcal{T}}])$ **do**
32: $E_{\mathcal{M}} \leftarrow P_{\texttt{move}}(v_{g,i}, v_{g,j})$
33: **if** $E_{\mathcal{M}} == \emptyset$ **then**
34: **if** `singleShot` **then**
35: **return** $\mathcal{H}_{\mathcal{M}}$
36: **else**
37: continue
38: $\mathcal{E}_{\mathcal{M}} \leftarrow \mathcal{E}_{\mathcal{M}} \cup \{E_{\mathcal{M}}\}$
39: **if** `singleShot` **then**
40: **return** $\mathcal{H}_{\mathcal{M}}$
41: **return** $\mathcal{H}_{\mathcal{M}}$

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

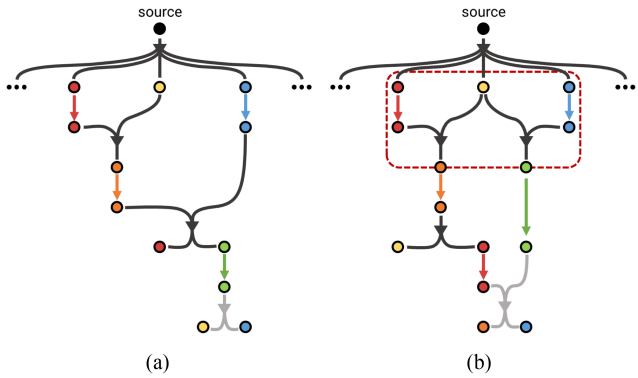


Fig. 10. Both figures (a and b) consider a transition-extended version of the multimanipulator space depicted in the task space hypergraph (see Fig. 8) and motion hypergraph (see Fig. 9) where robot 2 can now reach the object's starting position and there may be more robots and objects involved outside the vertices and hyperarcs depicted. The colors of the vertices correspond to the previous hypergraphs (yellow-object, red-robot 1, and blue-robot 2). In either figure, we consider the transition history for the tail set of the light gray hyperarc. (a) Depicts a valid transition history where no vertex has more than one outgoing hyperarc. (b) Depicts an invalid transition history. The conflict in the transition history is in the dotted red box. By including the pair of outgoing hyperarcs of the yellow object vertex in the same transition history, the object must exist in two conflicting states at once. (a) Valid history. (b) Invalid history.

In addition to ensuring that each moveable body performs at most one motion at once, the transition history also encodes the time each transition or move is expected to occur and the "last known position" for each moveable body according to the set of transitions and moves in the transition history.

2) Transition-Extended Hyperarcs: Each hyperarc $E_{\text{TE}} =< \texttt{Tail}, \texttt{Head}, E_{\mathcal{M}} >$ corresponds to either a composition, transition, or move hyperarc $E_{\mathcal{M}} \in \mathcal{E}_{\mathcal{M}}$. The head set consists of vertices, which map to the motion vertices in the motion head set of $E_{\mathcal{M}}$. The tail set consists of the union of vertices, which map to the motion tail set of $E_{\mathcal{M}}$ and a set of *scheduling constraint* vertices.

A set of transition-extended vertices can form a tail set if the union of their transition histories is valid. This is necessary for the head set vertices of E_{TE} to have valid transition histories.

3) Goal Vertices: Any transition-extended vertex v_{TE} such that $v_{\text{TE}}.v_{\mathcal{M}} == v_{\mathcal{M}}^{\texttt{target}}$ is considered a valid goal vertex. As each $v_{\text{TE}} \in \mathcal{V}_{TE}$ contains a valid and unique transition history, the transition history of the goal vertex corresponds to a unique *optimistic schedule* of compositions, transitions, and moves to complete the task. This schedule is optimistic as, while motions are valid within their own subconfiguration spaces, they are not guaranteed to be collision free with respect to the paths of other moveable bodies occurring simultaneously. This requires the use of the conflict resolution stage (discussed in Section IV-E), which seeks to adjust individual motion plans to resolve conflicts.

4) Scheduling Constraints: Some motion conflicts will not be resolvable by the conflict resolution layer. This occurs when the path or position of one moveable body cannot occur simultaneously with the path or position of another moveable body. These are motion-based *scheduling conflicts*.

Scheduling conflicts can occur either between a pair of motion vertices and a pair of motion hyperarcs, or a vertex and a hyperarc. *Vertex–vertex conflicts* indicate that once one vertex has been reached, the other cannot be reached until the first one has been left. This occurs when the configurations for the two motion vertices are in collision.

Hyperarc–hyperarc conflicts indicate that the corresponding paths cannot occur simultaneously, and one must be concluded before the other can begin. This occurs when the motions within separate subconfiguration spaces are found to be incompatible with each other.

Vertex–hyperarc conflicts indicate that the configuration of a motion vertex completely impedes the path of the hyperarc. For example, this may occur if a stationary object is blocking the motion of robot.

The conflict resolution stage (see Section IV-E) may discover these scheduling conflicts and pass the information back to the representation layers as a *scheduling constraint* associated with a motion vertex or hyperarc. Scheduling constraints indicate that the associated motion vertex/hyperarc pair cannot be included in the hyperpath simultaneously. Transition histories must satisfy the scheduling constraints of the included motion vertices/hyperarcs to be valid. Scheduling vertices can be included in a tail set for a transition-extended hyperarc E_{TE} to augment the transition history with additional transitions and motions that satisfy the scheduling constraint. In the case of an object blocking the path of a robot, this encodes that the object must not be at the blocking location during the execution of that path of the robot.

5) Identifying Composition Hyperarcs: Outside of introducing scheduling constraints, some motion conflicts can be addressed by adding a composition hyperarc to the task space hypergraph $\mathcal{H}_{\mathcal{T}}$ that provides access to the composite space of the conflicting sets of moveable bodies. This requires the update of the motion and transition-extended hypergraph layers as well to fully encode the composite shift and allow the conflict to be resolved in the composite space. This resembles the approach taken by M* [28] upon discovering conflicts.

6) Implicit Construction and Search: As the purpose of this graph is to find an optimistic schedule in the form of a goal vertex, and $\mathcal{H}_{TE}$ is potentially infinitely large, we search an implicit representation of $\mathcal{H}_{TE}$ and construct the necessary vertices and hyperarcs as a part of the search process. This search, as described in Algorithm 4, is an adaptation of the optimal shortest hyperpath query algorithm, as presented in [4], to account for the implicit representation and leverage the particular constraints of the transition-extended hypergraph (e.g., each vertex has only a single incoming hyperarc).

The search starts by initializing the source vertex $v_{\text{TE}}^{\texttt{source}}$ and the explicit $\mathcal{H}_{TE}$ on lines 2 and 3 along with the vertex weight function W, the frontier queue Q, and the parent map from vertices to incoming hyperarcs P_v on lines 4 and 5.

A set of *partial hyperarcs* U is initialized on line 6. Due to the implicit (and potentially infinite) nature of the hypergraph, the *forward star* (FS), or set of all outgoing hyperarcs, of a transition-extended vertex $v_{\texttt{te}}$ cannot always be computed. This occurs when the forward star of the corresponding motion vertex

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

Algorithm 4: Transition-Extended Hypergraph Search.

1: **Procedure** Search $\mathcal{H}_{TE}$Motion hypergraph $\mathcal{H}_{\mathcal{M}}$, $v_{\mathcal{M}}^{\texttt{source}}$, $v_{\mathcal{M}}^{\texttt{target}}$
2: $v_{\texttt{TE}}^{\texttt{source}} \leftarrow < v_{\mathcal{M}}^{\texttt{source}}, \Pi_{v_{\texttt{TE}}^{\texttt{source}}, v_{\texttt{TE}}^{\texttt{source}}} >$
3: $\mathcal{H}_{TE} \leftarrow (\{v_{\texttt{TE}}^{\texttt{source}}\}, \emptyset)$
4: Weight function $W(v_{\texttt{TE}}^{\texttt{source}}) \leftarrow 0$
5: Queue $Q \leftarrow \{v_{\texttt{TE}}^{\texttt{source}}\}$; Parent map $P_v \leftarrow \emptyset$
6: Partial hyperarcs $U \leftarrow \emptyset$
7: **while** $Q \neq \emptyset$ **do**
8: $u \leftarrow Q.\texttt{pop}()$
9: **for all** $E_{\texttt{TE}} \in \texttt{FS}_{\text{complete}}(u, U)$ **do**
10: $\mathcal{E}_{TE} \leftarrow \mathcal{E}_{TE} \cup \{E_{\texttt{TE}}\}$
11: $f \leftarrow \texttt{max}_{v_{\texttt{TE}} \in E_{\texttt{TE}}.\texttt{Tail}} W(v_{\texttt{TE}})$
12: **for all** $y \in E_{\texttt{TE}}.\texttt{Head}$ **do**
13: $\mathcal{V}_{TE} \leftarrow \mathcal{V}_{TE} \cup \{y\}$
14: $Q \leftarrow Q \cup \{y\}$
15: $W(y) \leftarrow f + w(E_{\texttt{TE}})$ ▷$w(E_{\texttt{TE}})$ is the weight of $E_{\texttt{TE}}$
16: $P_v[y] = E_{\texttt{TE}}$
17: **if** $y.v_{\mathcal{M}} == v_{\mathcal{M}}^{\texttt{target}}$ **then**
18: **return** y
19: $U \leftarrow \texttt{UpdatePartialHyperarcs}(y)$
20: **return** NULL

$\texttt{FS}(v_{\texttt{te}}.v_{\mathcal{M}})$ includes hyperarcs with tail sets containing more than one vertex as any set of transition extensions of the motion tail set with nonconflicting transition histories may form a tail set for a transition-extended hyperarc. Instead, we track partial hyperarcs in U, where we dynamically construct valid tail sets as new transition-extended vertices are discovered.

When the search attempts to expand from a vertex u, all completed hyperarcs in the forward star of u are expanded (line 9). This maintains the search properties in the original hyperpath query in [27], which only expands a hyperarc once all of the vertices in its tail set have been expanded. The set of partial (and completed) hyperarcs U is updated when a new vertex is added to the frontier queue (line 19).

For each complete hyperarc $E_{\texttt{TE}}$ in the forward star of y, the search adds $E_{\texttt{TE}}$ to the hypergraph on line 10 and computes the maximum weight of the vertices in the tail set on line 11. In the context of planning, this naturally captures any waiting time required by the moveable bodies in one tail vertex on the moveable bodies in the other tail vertices to arrive before beginning the transition.

Next, each of the vertices in the head set of $E_{\texttt{TE}}$ is added to the hypergraph and the frontier queue (lines 13 and 14) before setting the weight and parentage of each vertex (lines 15 and 16). Finally, a check is performed to see if any of the head set vertices correspond to the motion target vertex $v_{\mathcal{M}}^{\texttt{target}}$, indicating that an optimistic schedule has been found (line 17). If the queue is empty, and no schedule has been found, a NULL solution is returned.

E. Conflict Resolution

The optimistic scheduled produced by searching $\mathcal{H}_{TE}$ corresponds to scheduled motions, which are only guaranteed to be feasible within their own configuration spaces. Thus, there is no guarantee that these motions are conflict free with each other. We address this with a conflict resolution layer.

Each motion within the schedule is defined by a start and a goal for the corresponding moveable bodies along with any precedence constraints defined by the schedule. This resembles a scheduled variant of the MRMP problem, as discussed in Section II-B. As such, scheduled variations of MRMP methods may be adapted to perform conflict resolution with different theoretical properties.

For fast, single-shot planning, priority-based decoupled methods, such as priority-based search [29], can provide quick solutions. When an optimal search over the current representation is needed, hybrid methods, such as CBS-MP [11], can be adapted to account for scheduling constraints in a manner similar to the grid-world scheduled CBS variant in [4].

F. Method Variants

The various components of the DaSH method can be configured to balance a tradeoff in planning times versus solution quality. On the two extremes, we have the option to return the first valid solution found or to continue searching until an asymptotically optimal solution is found.

1) Faster Planning Times Versus Higher Solution Quality: The high-level algorithm described in Algorithm 1 can be configured to return the first valid solution by setting `earlyTermination=True` from the beginning. Alternatively, permanently leaving `earlyTermination=False` will allow the algorithm to converge on the optimal solution so long as the `ExpandRepresentation` function maintains asymptotically optimal properties and the loop in lines 3–11 returns the optimal solution for the current representation. The algorithm can be configured with anytime properties if `earlyTermination` is instead a function, which signals that time is up and the current best solution is returned.

2) Representation Construction: As the representation construction is a hierarchical construction process, the design choices at each level can impact the properties of the algorithm as a whole. The task space hypergraph $\mathcal{H}_{\mathcal{T}}$ construction can be modified by changing the behavior of the functions in Algorithm 2. If `earlyQuit=True`, then choosing a greedy `ExpandHypergraph` function can lead to a small, concise $\mathcal{H}_{\mathcal{T}}$ containing the minimum number of task space elements and transitions to move from the start element to the goal element. This can be beneficial when the task space is very expansive, and there is not a set of allowable transitions that can keep the size down. Alternatively, if the problem does have an effective set of allowable transitions, then a BFS style `ExpandHypergraph` function and `earlyQuit=False` can quickly build a complete $\mathcal{H}_{\mathcal{T}}$. So long as `ExpandHypergraph` is configured to continue expanding $\mathcal{H}_{\mathcal{T}}$ everytime is called until $\mathcal{H}_{\mathcal{T}}$ contains a complete representation of $\mathcal{T}$ (i.e., all admissible task space elements can be formed by combining elements represented in $\mathcal{H}_{\mathcal{T}}$), probabilistic completeness, asymptotic optimality, and anytime properties are still obtainable.

The size of the $\mathcal{H}_{\mathcal{T}}$ directly affects the effort of Algorithm 3 as each hyperarc in the $\mathcal{H}_{\mathcal{T}}$ may be sampled many times (lines 18 and 32). The theoretical properties of the anytime hypergraph

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

$\mathcal{H}_\mathcal{M}$ are determined by the $\mathcal{H}_\mathcal{T}$ Algorithm 3 given and the transition, composition, and move planners. If the move planner is probabilistically complete and asymptotically optimal within each of the task space elements (e.g., PRM* [30]), then the transition and composition planners continue to sample new hyperarcs, and the motion hypergraph will converge to the optimal set of motions for the given $\mathcal{H}_\mathcal{T}$.

3) Next-Best Search: Asymptotic optimally requires an optimal query of the current representation in lines 5–11 of Algorithm 1. We use the next-best search algorithm (see Algorithm 5) to iteratively build and validate task plans until it converges to the optimal solution for the current representation.

Each task plan represents an *optimistic schedule* of independent motions not guaranteed to be collision free of each other. The cost of this optimistic schedule serves as a lower bound for the solution cost as the solution cost can only increase from resolving conflicts. The algorithm requires each successive call to `FindTaskPlan` (line 6) to return the *next-best* task plan in terms of cost (i.e., each optimistic schedule cost must be at least as much as the previous schedule cost). This allows each successive iteration to update the lower bound of the solution.

Each optimistic schedule must be converted into a valid solution by resolving any potential conflicts. The `ConflictResolution` function (line 8) uses the roadmaps in the motion hypergraph and the schedule to construct a set of collision-free paths following the scheduled generated by `FindTaskPlan`. If the cost of this valid solution is lower than the previous upper bound, we update the upper bound to reflect this solution and save it as the best solution (lines 9 and 10). If no valid solution can be found in the current representation, the cost is treated as infinite, and the upper bound is unchanged. Schedules are saved in $\mathcal{S}$ (line 11) and given to the task planner to ensure that it returns the next-best solution (line 6).

The algorithm iterates until an optimistic schedule is produced with a cost greater than the current upper bound (line 5). At this point, the current best solution is determined to be the optimal solution with respect to the current representation, and this solution is returned (line 12). This iteration follows the strategy proposed by Brown et al. [4]. If no valid plan is found, it returns to the representation construction and expands the representation.

4) Task Planning: In order for Algorithm 5 to return the optimal solution for the current representation, the `FindTaskPlan` function must always return the *next-best* solution. This requires it to return the optimal solution the first time it is called, and each solution cost serves as a lower bound for the next call. Fortunately, the transition-extended hypergraph $\mathcal{H}_{TE}$ makes this simple, as each vertex contains the unique transition history to itself, a satisfying goal vertex represents a unique solution. If Algorithm 4 finds an optimal solution, then the *next-best* solution can be obtained by removing the previous solution from $\mathcal{H}_{TE}$ and calling the search again. This also reduces the cost of computing subsequent solutions if the search frontier is maintained. We will discuss two optimal variants of Algorithm 4 and a greedy alternative.

Algorithm 5: Next-Best Search.

1: **Procedure** NBSMotion hypergraph $\mathcal{H}_\mathcal{M}$
2: $(S^*, T^*) \leftarrow (\emptyset, \infty)$ ▷Best solution
3: $\underline{T} \leftarrow 0$ ▷best lower bound
4: $\mathcal{S} \leftarrow \emptyset$ ▷Set of discovered solutions
5: **while** $T^* > \underline{T}$ **do**
6: $(P, \underline{T}) \leftarrow$ `FindTaskPlan`$(\mathcal{H}_\mathcal{M}, \mathcal{S})$
7: **if** $T^* > \underline{T}$ **then**
8: $(S, T) \leftarrow$ `ConflictResolution`$(\mathcal{H}_\mathcal{M}, P)$
9: **if** $T < T^*$ **then**
10: $(S^*, T^*) \leftarrow (S, T)$
11: $\mathcal{S} \leftarrow \mathcal{S} \cup \{P\}$
12: **return** $S*$

Dijkstra-like Hyperpath Query: The first option resembles a Dijkstra shortest path search over $\mathcal{H}_{TE}$ and follows Algorithm 4 explicitly. The frontier queue Q is ordered on the weight $W(y)$ of vertices (line 15).

A-like Hyperpath Query:* An A*-like search can be configured by adding a lower bound cost-to-go heuristic value to each vertex v_{TE} in the frontier queue Q and ordering the queue by the sum of the weight and cost-to-go heuristic. We consider a cost-to-go heuristic derived from path (not hyperpath) distances, as described in Section II-E2, on the motion hypergraph from $v_{\text{TE}}.v_\mathcal{M}$ to $v_\mathcal{M}^{\texttt{target}}$. This path must only consist of vertices for which the corresponding moveable bodies $v_{\text{TE}}.v_\mathcal{M}.v_\mathcal{T}.T_i.B_i$ are specified in the goal constraints of the problem. The heuristic value of vertices without goal specified moveable bodies is 0, and they cannot be included in the heuristic path. In object rearrangement problems, this is equivalent to computing the path distance for an object to its goal state from a given motion hypergraph vertex while ignoring any robot-only vertices and hyperarcs. This provides a lower bound by assuming that all other dependencies happen in parallel before this goal constraint is satisfied (e.g., all robots are immediately available to perform actions and all other objects have already reached their goal positions).

Greedy Hyperpath Query: If the optimality is not required and a fast solution is preferred, then a greedy search can be used in place of Algorithm 4. The simplest greedy option iteratively constructs a valid transition history one hyperarc at a time. This approach benefits from a heuristic for choosing the next hyperarc to add, and it may be forced to backtrack if the choices made do not allow for a feasible solution to be reached. If this search is complete, then probabilistic completeness may be maintained if it is supported by the other choices.

5) Conflict Resolution: The variants of conflict resolution are discussed in Section IV-E. An optimal option, such as the scheduled variant of CBS-MP [11], is required for Algorithm 5 to return an optimal solution. Otherwise, the fastest option may be used, although the completeness affects the probabilistic completeness property of the configuration.

Section VIII evaluates an asymptotically optimal configuration using NBS with the A* variant of Algorithm 4 and a fast, greedy approach, which uses the greedy hypergraph query to generate a single task plan for which it attempts to resolve

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

conflicts before expanding the representation. Both methods use the scheduled variant of CBS-MP [11] to resolve conflicts.

V. Application to MRMP

We examine the application of the DaSH method to the multi-robot motion planning problem (MRMP-DaSH) to illustrate the concepts discussed in the previous sections and show how the proposed framework generalizes many existing MRMP methods. We first formally define the problem: Given a set of robots $\mathcal{B}$ and an environment, find continuous collision-free paths for each robot $r_i \in \mathcal{B}$ from a given start s_i to a given goal g_i in the environment.

The task space $\mathcal{T}_{\texttt{MRMP}}$ consists of elements $T_i = (B_i, \mathcal{W}_i, C_i)$, where $B_i \subseteq \mathcal{B}$, $\mathcal{W}_i$ is the configuration space for B_i, and C_i constrains each $b_j \in B_i$ to not be in collision with any obstacle or any $b_k \in B_k, b_k \neq b_j$. There is a single admissible task space element $\mathcal{T}^* = \{T_{\texttt{complete}} = (B_{\texttt{COMPLETE}}, \mathcal{W}_{\texttt{COMPLETE}}, C_{\texttt{COMPLETE}})\}$, where $B_{\texttt{COMPLETE}} = \mathcal{B}$ and $C_{\texttt{COMPLETE}}$ contain all collision avoidance constraints.

The hypergraph representation layers defined in Section IV-B–D can be configured by adjusting the allowed transitions A to create the different planning spaces commonly considered by MRMP methods.

A. Composite Representation

MRMP-DaSH can be reduced to composite MRMP methods, such as the work of [15] and [16], by considering allowed transitions A, which constrain the task space hypergraph $\mathcal{H}_{\mathcal{T}_{\texttt{MRMP}}} = (\mathcal{V}_{\mathcal{T}}, \mathcal{E}_{\mathcal{T}})$ such that a vertex $v_i \in \mathcal{V}_{\mathcal{T}}$ iff $v_i.T_i = T_{\texttt{COMPLETE}}$. This results in a $\mathcal{H}_{\mathcal{T}_{\texttt{MRMP}}}$ with a single vertex and no hyperarcs. The motion hypergraph $\mathcal{H}_{\mathcal{M}} = (\mathcal{V}_{\mathcal{M}}, \mathcal{E}_{\mathcal{M}})$ contains a pair of vertices, one for the start and one for the goal configuration in $\mathcal{W}_i$, along with the virtual source and target vertices. A single move hyperarc $E^{\texttt{move}}_{\mathcal{M}}$ connects the start and goal configurations, and virtual hyperarcs connect these to the virtual source and target vertices. As there are no meaningful transitions, the transition-extended hypergraph mirrors the motion hypergraph. The computation of the path $E^{\texttt{move}}_{\mathcal{M}}.\pi$ provides a solution to the MRMP problem directly in the composite configuration space; thus, no conflict resolution stage is required.

B. Decoupled Representation

MRMP-DaSH can be reduced to MRMP methods that utilize decoupled representations, such as the work of [11] and [15], by considering allowed transitions A, which constrains $\mathcal{H}_{\mathcal{T}_{\texttt{MRMP}}}$ such that a vertex $v_i \in \mathcal{V}_{\mathcal{T}}$ iff $|v_i.T_i.B_i| < |B|$ and $\mathcal{E}_{\mathcal{T}} = \emptyset$. It is often further constrained such that each task space element represented in $\mathcal{V}_{\mathcal{T}}$ contains only a single robot. The corresponding motion hypergraph contains a start and goal vertex within each task space element represented by a vertex in $\mathcal{V}_{\mathcal{T}}$ and a move hyperarc between them (see Fig. 11). Additional hyperarcs connect the set of valid start and goal vertices to the virtual source and target vertices. Fig. 11 depicts $\mathcal{H}_{\mathcal{M}}$ for a fully decoupled representation.

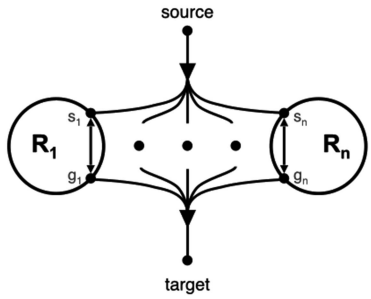


Fig. 11. Motion hypergraph for a decoupled representation of the MRMP problem contains a single hyperarc within each individual robot subspace in addition to the source and target hyperarcs. The hyperarc within the task space element for each individual robot is an abstraction of the path across the corresponding roadmap within that element. All robots must be included in the tail set of the target hyperarc to capture the requirement that all robots reach their goal.

There are no meaningful transitions in $\mathcal{H}_{\mathcal{M}}$, so the transition-extended hypergraph $\mathcal{H}_{TE}$ resembles $\mathcal{H}_{\mathcal{M}}$. The independent paths in the move hyperarcs are not guaranteed to be collision free with each other, so the conflict resolution stage is required to produce a solution valid in $T_{\texttt{COMPLETE}}$. The algorithm design choices for constructing the independent paths and resolving conflicts produce different MRMP algorithms with varying properties. For example, using asymptotically optimal sampling-based approaches to iteratively construct and improve the individual paths can enable hybrid search methods, such as CBS-MP [11], to provide asymptotically optimal solutions. Alternatively, priority-based methods, such as decoupled PRM [15] or PBS [29], applied to roadmaps will provide fast, suboptimal solutions. The properties of the grid-world MAPF algorithms [29], [31] depend on the resolution of their grid representation.

C. Hybrid Representation

Hybrid representations for MRMP-DaSH include composition hyperarcs in $\mathcal{H}_{\mathcal{T}_{\texttt{MRMP}}}$, allowing the subsequent representation layers to transition between $\mathcal{C}_{space}$ compositions. This requires a decision to be made on when to change compositions. Methods, such as M* [28], change compositions as a means of resolving conflicts between decoupled paths. This results in an iterative construction and search approach, which lazily adds composition hyperarcs to the motion hypergraph $\mathcal{H}_{\mathcal{M}}$ around conflict locations. The transition-extended hypergraph $\mathcal{H}_{TE}$ ensures that all solutions contain a single continuous path for each robot in the problem.

VI. Application to Multimanipulator Rearrangement

We consider multimanipulator rearrangement problems to illustrate our proposed approach and its benefits. This application is referred to as MM-DaSH. We first define the problem: Given a set of moveable bodies $\mathcal{B} = \mathcal{R} \cup \mathcal{O}$ consisting of robot manipulators $\mathcal{R}$ and manipulable objects $\mathcal{O}$, an action space expressed as allowed transitions $\mathcal{A}$, and an environment, the planner must find a sequence of actions and motions for the

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

robots to take such that the manipulable objects move from a start pose $p_{\texttt{start}}$ to a satisfactory goal pose $p_{\texttt{goal}}$.

The task space $\mathcal{T}_{\texttt{manip}}$ with elements $T_i = (B_i, \mathcal{W}_i, C_i)$, where B_i contains some set of robots and objects, $\mathcal{W}_i$ denotes their $\mathcal{C}_{space}$, and C_i may constrain an object $o_j \in B_i$ to be at rest on a stable surface or in a stable grasp of some robot $r_k \in B_i$ in addition to the standard collision avoidance constraints. The set of admissible task space elements $\mathcal{T}^*$ consists exclusively of elements T_i, where $B_i = \mathcal{B}$ and C_i must constrain every object to either a stable surface or grasp in additional to the full set of collision avoidance constraints. The constraint set $C_{\texttt{COMPLETE}}$ in $T_{\texttt{complete}}$ contains all allowed grasps and stable surfaces for the objects and robots in $\mathcal{B}$.

A. Composite Representation

MM-DaSH can be configured to existing multi-robot rearrangement methods, such as the work of Shome and Bekris [10], by considering a set of allowable transitions A, which constrains the task space hypergraph $\mathcal{H}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ such that a vertex $v_i \in \mathcal{V}_\mathcal{T}$ iff $v_i.T_i \in \mathcal{T}^*$. As all task space elements represented as vertices in $\mathcal{V}_\mathcal{T}$ contain all $r, o \in \mathcal{B}$, there are no composition hyperarcs in $\mathcal{H}_\mathcal{T}$. The action space available to the robots determines the transition hyperarcs $E^{\texttt{trans}}_\mathcal{T}$ included in $\mathcal{H}_\mathcal{T}$. A hyperarc $E^{\texttt{trans}}_\mathcal{T} \in \mathcal{E}_\mathcal{T}$ may include a set of simultaneous, independent actions transitioning between task space elements in $\mathcal{T}^*$, so long as each robot and object is only involved in single action (e.g., Robot 1 picks object A, while Robot 2 places object B). As all these transition hyperarcs have a head and tail set of size one, $\mathcal{H}_\mathcal{T}$ is a graph.

The motion hypergraph encodes paths for allowed transitions and motions within individual task space elements between transitions. The transition-extended hypergraph encodes possible sequences of simultaneous actions and motions for the system to take. As any path computed through a sequence of task space elements always considers all moveable bodies and collision constraints, no conflict resolution is needed.

Existing rearrangement methods, such as SMART [10], can be modeled in this framework and often perform simultaneous construction and search. They build an oracle for planning move hyperarcs $E^{\texttt{move}}_\mathcal{M} \in \mathcal{H}_\mathcal{M}.\mathcal{E}_\mathcal{M}$ (e.g., precompute PRM* for each robot) and then rely on heuristics to decide which hyperarc $E^{\texttt{trans}}_\mathcal{T}$ to attempt to sample and add to the motion hypergraph $\mathcal{H}_\mathcal{M}$. They then consider taking the equivalent hyperarcs in the transition-extended hypergraph $\mathcal{H}_{TE}$ until $\mathcal{H}_{TE}$ contains a hyperpath corresponding to a sequence of moves and transitions that solve the task.

B. Decoupled Representation

MM-DaSH can be configured to consider smaller task space elements with allowable transitions A, which constrain $\mathcal{H}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{H}_\mathcal{T})$ such that $v_i \in \mathcal{V}_\mathcal{T}$ iff the $B_i \in v_i.T_i$ cannot be partitioned without relaxing a constraint in C_i other than interbody collision avoidance. For example, grasping constraints result in task space elements $T_i = (B_i, \mathcal{W}_i, C_i), B_i = \{o_j, r_k\}$, where robot r_k is grasping object o_j according to a constraint in C_i. Task space elements with no grasping constraints contain either only a single object or robot.

A decoupled representation requires that no composition hyperarcs exist in $\mathcal{H}_\mathcal{T}$. Transition hyperarcs must be between sets of vertices in $\mathcal{V}_\mathcal{T}$ and follow the rules described in Section IV-B. These correspond to independent actions in the action space (e.g., pick, place, and handoff), as depicted in Fig. 8.

The variations of representation construction and search, as presented in Section IV, produce decoupled or hybrid approaches with varying theoretical properties over this decoupled task space representation. Hyperpaths in the transition-extended hypergraph $\mathcal{H}_{TE}$ correspond to invalidated solutions in the set of admissible task space elements, so a conflict resolution stage is required. Options for this stage are discussed in Section IV-E. We provide the decision choices for two decoupled representation options with hybrid search techniques in Section VIII, which are included in our empirical evaluations.

C. Hybrid Representation

The hybrid representation for MM-DaSH applies the same modifications to the decoupled representation as in MRMP-DaSH. Composition hyperarcs are allowed in $\mathcal{H}_\mathcal{T}$ and the subsequent representation layers. The decision remains on where to include composition changes. Further investigation on this question is outside the scope of this work.

VII. Representation Analysis

Having an explicit representation of the planning space enables an incredibly informed generic heuristic based on the representation itself. The composite graph-based representation provides more complete information than the decoupled hypergraph-based representation; however, in many applications, the graph representation of the task space derived from only considering the admissible task space elements is too large to ever use explicitly. Existing methods, which consider only the admissible composite elements instead, use heuristics tailored to their problem to greedily search an implicit representation of the graph [10]. In contrast, if a hypergraph representation of decomposed task space elements is small enough to build, it allows the use of more generic heuristics computed from the various hypergraph layers, such as the cost-to-go heuristic discussed in Section IV-F4.

In this section, we compare representing the task space with the traditional composite graph representation to the proposed decoupled hypergraph representation for the multimanipulator problems, as discussed in Section VI. We show that the size of the proposed representation scales well with the problem size and, thus, is able to be constructed and used in heuristics.

We consider the pick, place, and handoff action space, and assume that all robots can reach each other to perform handoffs and all objects to perform pick/place actions. This is an overestimate as it ignores reachability constraints but serves as an informative upper bound on the space complexity as a planning method must reason over reachability before eliminating potential transitions. Each robot can hold at most one object. Each object can be held

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

by at most one robot. For simplicity, we ignore specific grasp constraints' parameters or object poses and just consider the allocation of objects to robots.

A. Graph Representation

We first consider a graph-based composite representation of the task space. The moveable bodies of a task space element contain the full set of n robots and m objects.

1) Total Number of Vertices: Let $P_{\max} = \texttt{min}(m, n)$ be the maximum number of objects that can be held in a scene with n robots and m objects. Let $p \in \{0, \ldots, P_{\max}\}$ be the number of objects currently held in a given task space element's constraints A^{nxm}. Then, the number of unique sets of p objects that are held in a given task space element is $\binom{m}{p}$. The number of unique sets of p robots holding those objects in a given task space element is $\binom{n}{p}$. Considering all possible assignments of the held objects to the robots and the different values of p, the total number of elements $\mathcal{S}$ is given as follows:

$$\mathcal{S}(m,n) = \sum_{p=0}^{P_{\max}} \binom{m}{p}\binom{n}{p} p!. \tag{4}$$

2) Total Number of Edges: We first consider the set of available place actions. Let $l \in \{0,...,p\}$ be the number of objects placed in a given transition. Then, the number of unique sets of objects placed in a given transition is $\binom{p}{l}$. The total number of transitions just performing a place action is given by (5), where 1 is subtracted for the case $l = 0$ and no transition occurs

$$-1 + \sum_{l=0}^{p} \binom{p}{l}. \tag{5}$$

$n - p$ is the number of free robots in a given task space element. $H_{\max} = \texttt{min}(p - l, n - p)\}$ denotes the maximum number of objects handed off in a given transition. Let $h \in \{0, \ldots, H_{\max}\}$ be the number of objects handed off in a given transition. Then, the number of unique sets of objects handed off in a given transition is $\binom{p-l}{h}$, and the number of unique robots receiving the handoffs is $\binom{n-p}{h}$. Considering all possible assignments, the total number of place and handoff transitions is given by (6), where 1 is subtracted for the case $l = 0$ and $h = 0$ and no transition occurs

$$-1 + \sum_{l=0}^{p} \binom{p}{l} \sum_{h=0}^{H_{\max}} \binom{p-l}{h}\binom{n-p}{h} h!. \tag{6}$$

$m - p$ is the number of free objects in a given task space element. $n - p - h$ denotes the number of robots not holding an object in an element and not receiving an object via a handoff in a given transition. $G_{\max} = \texttt{min}(m - p, n - p - h)$ is the maximum number of objects picked in a given transition. Let $g \in \{0, G_{\max}\}$ be the number of objects picked in a given transition. Then, the number of unique sets of objects being picked is $\binom{m-p}{g}$, and the number of unique sets of robots picking is $\binom{n-p-h}{g}$. Considering all possible assignments, the total number of transitions from a given task space element performing a set of handoff actions is given by (7), where 1 is subtracted for the case $l = 0, h = 0,$ and $g = 0$, and no transition occurs

$$\mathcal{G}(m,n,p,h) = -1 + \sum_{g=0}^{G_{\max}} \binom{m-p}{g}\binom{n-p-h}{g} g!. \tag{7}$$

Let $\mathcal{H}$ be the total number of handoff and grasp action combinations possible, given $m, n, p,$ and l, and let $\mathcal{L}$ be the total number of transitions possible consisting of a combination of place, handoff, and pick actions, given $m, n,$ and p

$$\mathcal{H}(m,n,p,l) = \sum_{h=0}^{H_{\max}} \binom{p-l}{h}\binom{n-p}{h} h!\mathcal{G}(m,n,p,h) \tag{8}$$

$$\mathcal{L}(m,n,p) = \sum_{l=0}^{p} \binom{p}{l} \mathcal{H}(m,n,p,l). \tag{9}$$

The total number of edges $\mathcal{T}$ in the graph is given by the total number of edges from each of the vertices, as given in the following equation:

$$\mathcal{T}(m,n) = \sum_{p=0}^{P_{\max}} \binom{m}{p}\binom{n}{p} p!\mathcal{L}(m,n,p). \tag{10}$$

B. Hypergraph Representation

The hypergraph-based representation of the decoupled task space considers decomposed task space elements containing every pairing of a robot grasping an object, each robot not holding an object, and each object not being grasped for a total of $mn + m + n$ vertices.

Each pick action is a transition from a robot-only vertex and an object-only vertex to the vertex for the task space element where the robot grasps that object. A place action is this transition in reverse. This results in $2mn$ transition hyperarcs for all pick/place actions.

Each handoff action is a transition from a pair of vertices, where robot 1 is grasping the object and robot 2 is free, to a pair of vertices, where robot 1 is free and robot 2 is grasping the object. Each of the mn grasping vertices can be paired with any of the $n - 1$ free robot vertices for a total of $mn^2 - mn$ handoff hyperarcs. This results in a total of $mn^2 + mn$ hyperarcs to capture all possible transitions.

C. Comparison

From (4), the graph-based representation maintains linear growth in the number of vertices when considering either a single robot or a single object but exhibits exponential growth as soon there are both multiple robots and multiple objects.

In contrast, in the hypergraph representation, the number of vertices scales linearly with the number of robots while holding the number of objects constant, regardless of the number of objects. The same is true if the number of objects changes and the number of robots is held constant.

The number of transitions follows a similar trend. If there is either a single object or a single robot, then the number of transitions is equal for graph- and hypergraph-based representations when the other quantity (robots or objects) increases. As soon

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

as a second object or a second robot is introduced, the growth of the number of task space transitions (edges) in the graph-based representation becomes exponential. In the hypergraph-based representation, the number of hyperarcs scales quadratically with the number of robots and linearly with the number of objects. This follows directly from the total number of hyperarcs, as discussed in Section VII-B.

Despite containing more information, the exponential growth in the size of the composite graph-based representation prevents it from being utilized in planning. The more concise hypergraph-based representation can be both constructed and leveraged to inform heuristics that are not problem specific. We present one instance of this in Section IV-F4, where heuristic values come from simple paths in the motion hypergraph.

To provide an intuitive visualization of the difference in representation size, we depict the graph and hypergraph representation for two robots and four objects along with a table showing the immediate size difference in smaller problems (see Fig. 2).

D. Alternative Problem Spaces

To examine the generality of the DaSH framework, we consider the size of the hypergraph-based representation for alternative problem spaces.

1) Two-Robot Grasping: We first consider a problem space where each object requires two robots to be simultaneously grasping an object to move it. We assume that each robot involved in a grasp has a specific role (i.e., robot 1 and robot 2). We maintain the $m + n$ vertices for each object alone and each manipulator alone task space element. Instead of the mn vertices corresponding to task space elements for object–robot pairs, we have $mn^2 - mn$ elements for each permutation of two robots and one object for a total of $mn^2 + m + n$ vertices in the task space hypergraph.

In this problem, there are $2mn^2 - 2mn$ pick/place transitions for every combination of robot pairs and objects. Additionally, there are $mn(n-1)(n-2)(n-3)$ handoff transitions as each of the m objects can be handed off between every ordered pair of robots. Thus, the number of transitions in the hypergraph-based representation still scales linearly with the number of objects and quartically with the number of robots.

The number of vertices in the task space hypergraph can be modeled for larger sets of k robots grasping an object by considering $m\frac{n!}{(n-k)!}$ task space elements, where k robots grasp each object with specific roles in the grasp (plus the isolated robot and object element vertices). Pick/place transitions are always twice this number for moving in either direction. Meanwhile, there are $m\frac{n!}{(n-k)!}\frac{(n-k)!}{(n-2k)!}$ handoff transitions.

2) Stacking: We consider a problem space where a set of m objects must be stacked in a particular order and can only be stacked in that order. Each object can be grasped by at most one robot at a time. From our original pick, place, and handoff action space, we add an additional m task space elements corresponding to every partial stack of $i \in [1, m]$ objects each of which corresponds to an additional vertex in the task space hypergraph. The addition of each of the m objects to the stack may be done by any of the n robots for an additional mn transition. Thus, the growth of the representation size maintains the same asymptotic behavior.

Now, consider instead the stacking problem, where each object requires two robots to grasp it, as described in Section VII-D1. We still have the m additional vertices for each partial stack. Instead of the mn stacking transitions in the single-robot grasp version, we now have $mn(n-1)$ stacking transitions. The asymptotic growth of the representation size again remains unchanged from the two-robot pick, place, and handoff rearrangement problem.

3) Supported Stacking: We add an additional constraint to the stacking problem, where one robot must be *supporting* the stack before an object can be added. This adds an additional mn vertices corresponding to task space elements, where each of the n robots is supporting one of the m partial stacks. Additionally, instead of mn stacking transitions for the single-robot grasp problem or $mn(n-1)$ stacking transitions for the two-robot grasp problem, there are now $mn(n-1)$ and $mn(n-1)(n-2)$ supported stacking transitions, respectively, where each stacking action now involves an additional robot supporting the stack. In either case, this is less than or equal to the number of handoff transitions in either problem's task space hypergraph, leaving the asymptotic growth unchanged.

VIII. Experimental Evaluation

In our experiments, we demonstrate the improvement in planning times from using the hypergraph-based representation for both problems with large numbers of objects and geometrically constrained manipulation tasks. We evaluate two decoupled representation configurations of the approach for the multimanipulator rearrangement problem defined in Section VI, MM-DaSH and Greedy-MM-DaSH, on a set of physical and simulated multi-robot manipulation problems, as depicted in Figs. 1 and 13(a). We compare against SMART [10] as a multi-manipulator method considering a composite representation for rearrangement tasks. In all scenarios, the objective is to move the given objects from their respective start locations to specified goal locations. Robots can perform pick, place, and handoff actions. We report the time to return the first solution and the cost of that solution for each method. Any planning attempt longer than 10^4 s is considered to have failed.

A. Methods

We describe the two DaSH configurations and the implementation of SMART [10].

1) MM-DaSH: We describe the *construction, task planning*, and *conflict resolution* used in both MM-DaSH and Greedy-MM-DaSH.

Construction—Both variants construct the entire task space hypergraph $\mathcal{H}_\mathcal{T}$ in the initial representation construction. The size relative to the number of robots and objects is given in Section VII.

Each round of representation expansion samples one additional motion transition for each task space transition. We use an IK-based sampling method for grasps and handoffs. A small roadmap is initialized with ten vertices for each task space

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

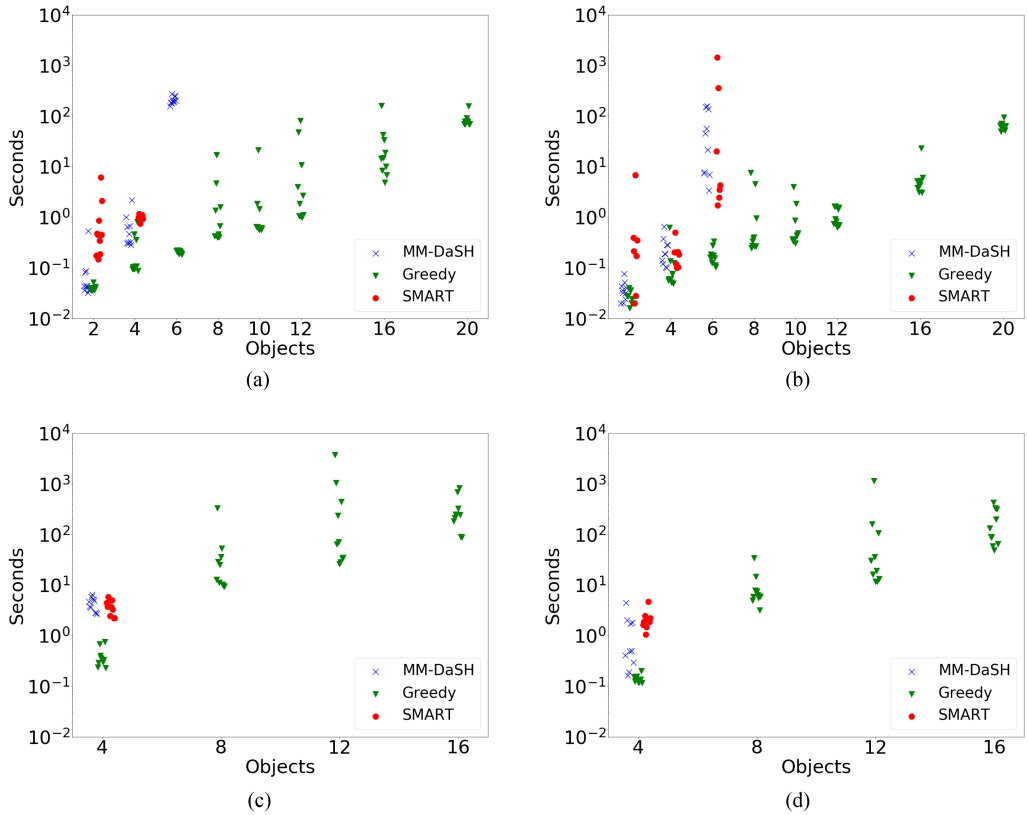


Fig. 12. Each figure shows the search time for every seed. The *Y*-axis denotes the seconds to compute a plan. The *X*-axis denotes the number of objects. The asymptotically optimal version, MM-DaSH, is marked with blue *X*. The greedy variant, Greedy-MM-DaSH, is marked with green triangles. The composite version, SMART [10], is marked with red circles. Figures (a) and (b) and (c) and (d) show the results for two-robot and four-robot sorting scenarios, respectively. Figures (a) and (c) have start locations such that each object must be handed off to reach the goal location. Figures (b) and (d) have random start locations for each object. (a) Sorting - 2 robots - cross. (b) Sorting - 2 robots - random. (c) Sorting - 4 robots - cross. (d) Sorting - 4 robots - random.

element using PRM*. The move planner uses these roadmaps to compute the paths included in the move hyperarcs by connecting the transition path start/end points and querying paths over the roadmaps. Each round of representation expansion adds ten additional samples to each roadmap.

Task Planning- MM-DaSH uses NBS and the A*-like hyperpath query, as described in Section IV-F4, to find optimal solutions for the current representation. This can require many calls to both the task planning and conflict resolution layers.

Greedy-MM-DaSH uses the Greedy hyperpath query, as described in Section IV-F4, and expands the representation after a single attempt at task planning and conflict resolution.

Conflict Resolution—Both methods use the scheduled variant of CBS-MP [11] to resolve conflicts in the optimistic schedule produced by the task planning layer.

Solution—Both methods return a solution once the representation contains one. MM-DaSH will find the optimal solution for that representation, while Greedy-MM-DaSH will return the first solution it finds.

2) SMART: We describe the details of our implementation of SMART [10].

Construction—We first create the object-centric-mode graph used to guide planning in SMART. We then run PRM* to create an initial roadmap with 100 vertices for each robot. We found that initializing each roadmap with 100 vertices led to the highest success rate when using the dRRT* [24] style MRMP in later stages of the algorithm, as detailed in [10]. The need for denser roadmaps than the CBS-MP style motion planning is an interesting question but is outside the scope of this work.

We modified the implementation of SMART to sample transition (picks, places, and handoffs) during the construction to create a fair comparison to the DaSH variants, which also sample transition in the construction phase. These are connected to the PRM* generated roadmaps.

Search—The implementation of the search stage follows the description in the original article [10]. We use the same MAPF formulation over the object-centric-mode graph as the guiding heuristic. The goal bias was 0.9. This was experimentally found to produce the best results. The same dRRT* style motion planning was used within each mode.

Solution—We return the first valid solution SMART finds.

Both MM-DaSH variants and SMART [10] were implemented in C++ in the Parasol Planning Library. We report the total search time in Figs. 12 and 13(b) and construction time, success rate, and solution quality in Table I.

B. Experimental Configuration

We evaluate the methods on a set of rearrangement tasks. A *sorting* scenario highlights the ability of the methods to

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

TABLE I
DETAILED RESULTS FOR BOTH TWO- AND FOUR-ROBOT SORTING SCENARIOS

Robots	Objects	Scenario	Method	Construction (s)		Search (s)		Cost (Timesteps)		Success
				Avg	Std Dev	Avg	Std Dev	Avg	Std Dev	
2	2	Cross	MM-DaSH	**11.39**	2.41	0.10	0.15	**993.40**	171.96	**1.00**
			Greedy	**11.39**	2.41	**0.04**	0.01	1000.80	172.15	**1.00**
			SMART	156.96	5.75	1.13	1.85	14139.00	23648.27	**1.00**
		Random	MM-DaSH	**12.01**	2.81	0.04	0.02	**641.20**	350.86	**1.00**
			Greedy	**12.01**	2.81	**0.03**	0.01	664.40	352.07	**1.00**
			SMART	159.71	8.78	1.14	2.50	3875.71	4257.81	0.70
	4	Cross	MM-DaSH	**23.32**	7.18	0.65	0.59	**1845.60**	191.95	**1.00**
			Greedy	**23.32**	7.18	**0.23**	0.24	2035.10	290.85	**1.00**
			SMART	167.48	17.52	0.97	0.13	4387.25	1312.74	0.80
		Random	MM-DaSH	**24.82**	8.42	0.24	0.17	**1059.40**	380.99	**1.00**
			Greedy	**24.82**	8.42	**0.13**	0.18	1199.80	395.14	**1.00**
			SMART	157.36	7.04	0.19	0.13	1920.50	1361.05	0.80
	6	Cross	MM-DaSH	**40.74**	11.81	207.75	36.21	**2399.20**	234.32	**1.00**
			Greedy	**40.74**	11.81	**0.21**	0.02	2752.80	320.78	**1.00**
		Random	MM-DaSH	**44.90**	21.09	59.26	63.77	**1607.30**	546.54	**1.00**
			Greedy	**44.90**	21.09	**0.18**	0.07	1969.70	640.94	**1.00**
			SMART	289.99	86.01	260.48	532.82	7911.29	4389.18	0.70
	8	Cross	Greedy	**122.55**	172.79	**2.73**	5.17	**3758.30**	254.71	**1.00**
		Random	Greedy	**74.24**	23.53	**1.53**	2.51	**2584.60**	492.95	**1.00**
	10	Cross	Greedy	**89.92**	36.93	**2.87**	6.46	**4612.30**	266.45	**1.00**
		Random	Greedy	**105.59**	63.35	**0.94**	1.15	**3092.10**	553.30	**1.00**
	12	Cross	Greedy	**163.50**	100.39	**15.16**	27.13	**5802.80**	450.26	**1.00**
		Random	Greedy	**170.32**	111.31	**1.08**	0.44	**3880.90**	420.81	**1.00**
	16	Cross	Greedy	**450.36**	470.13	**31.27**	46.36	**7579.80**	486.61	**1.00**
		Random	Greedy	**263.56**	103.63	**6.31**	5.97	**4884.40**	866.13	**1.00**
	20	Cross	Greedy	**484.06**	134.07	**1767.64**	5323.53	**9357.20**	463.72	**1.00**
		Random	Greedy	**651.83**	248.32	**63.50**	13.08	**6467.50**	882.46	**1.00**
4	4	Cross	MM-DaSH	**43.45**	9.67	4.29	1.35	**1946.50**	175.20	**1.00**
			Greedy	**43.45**	9.67	**0.39**	0.18	3277.20	344.71	**1.00**
			SMART	341.01	167.41	3.79	1.27	5486.60	1484.64	**1.00**
		Random	MM-DaSH	**36.15**	8.88	1.21	1.35	**1269.80**	244.09	**1.00**
			Greedy	**36.15**	8.88	**0.14**	0.03	1650.90	434.67	**1.00**
			SMART	300.22	145.21	2.16	1.03	3415.67	1358.18	0.90
	8	Cross	Greedy	**568.03**	108.08	**52.73**	98.85	**5899.60**	572.84	**1.00**
		Random	Greedy	**611.60**	166.98	**9.56**	9.07	**2857.50**	674.12	**1.00**
	12	Cross	Greedy	**1701.46**	416.15	**571.01**	1157.34	**9430.40**	766.14	**1.00**
		Random	Greedy	**1644.71**	368.27	**153.60**	347.90	**5383.10**	924.73	**1.00**
	16	Cross	Greedy	**3691.00**	580.97	**314.17**	245.12	**12285.10**	1158.18	**1.00**
		Random	Greedy	**3799.55**	844.19	**174.95**	135.29	**7537.70**	1509.79	**1.00**

The bold values denote the best results.

plan for increasing numbers of robots and objects. A *shelving* scenario demonstrates the ability of the MM-DaSH variants to account for geometric constraints arising from object placement. Each method was run with ten random seeds in each scenario. Experiments were run using a desktop computer with an Intel Core i9-10900KF CPU at 3.7 GHz, 128 GB of RAM.

We compute plans for virtual robots in each scenario. All robots are UR5e manipulators with hand-e grippers. A physical demonstration of the two-robot sorting scenario can be seen in the video linked in Fig. 1. SMART was adapted to account for transition paths to perform task space element switches instead of single-configuration switches to address the added complexity of using a two-finger gripper instead of the vacuum style gripper used in the original SMART experiments.

C. Sorting

To evaluate the ability of the method to efficiently find plans for large numbers of objects, we consider a scenario which imitates sorting tasks that may be encountered in warehouses or factories. A set of objects lies in a common workspace for a set of robots. Each object belongs to a particular class. The team of robots must sort the objects from the mixed group in the common workspace into bins of the corresponding class. In a full industry application, a computer vision component would be utilized to identify object classes and poses, but we focus only on the planning portion here and assume knowledge of this information. Our objects are simple cubes, and the class is denoted by the color in Fig. 1.

The initial configuration of the objects is randomly generated in the common workspace. The target configuration of the objects is randomly generated within the bin for the appropriate class. The comparison of our approach to SMART [10] is given in Fig. 12 and Table I.

1) Two-Robot Scenario: In this scenario, we consider two manipulators which must sort the blocks by color (see Fig. 1). We evaluate the runtime and solution quality of the methods as the number of objects increases.

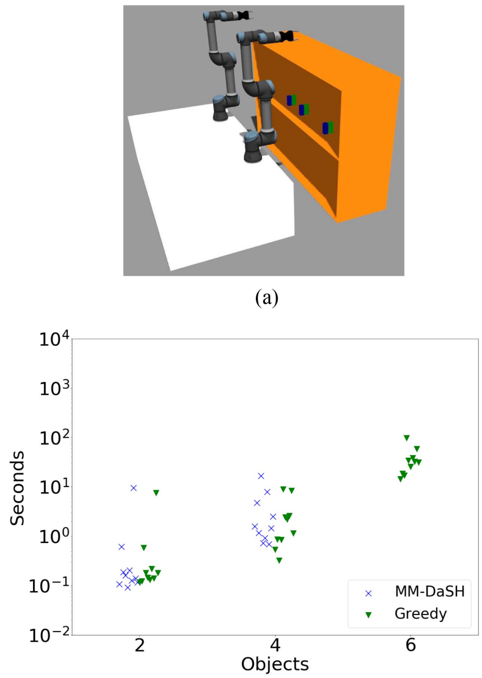
2) Four-Robot Scenario: In this virtual scenario, a group of four robots again must sort an increasing number of blocks. This is an expansion of the two-robot scenario with the robots arranged in a square. There are four classes of objects, one for each robot. This evaluates the performance of the method as the number of robots increases and the impact of the number of robots has on the performance of the method as the number of objects increases as well.

3) Analysis: As seen in Fig. 12, the DaSH variants are able to find plans faster and for a significantly larger number of objects (20) than the SMART baseline (6). SMART is unable to scale as the composite space it searches grows too large.

Both DaSH variants are able to efficiently find plans for smaller numbers of objects, and the heuristic based on the motion hypergraph effectively informs both variants which action to take next. MM-DaSH struggles with the task space complexity at six objects. This occurs when there are many roughly equivalent

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.



(b)

Fig. 13. (a) Robots must move the objects (which start randomly on the table) to positions on the shelf. The blue objects must be placed directly in front of the green objects, which requires the green objects to be placed first. (b) This figure reports the search time for each seed for the shelving scenario. The *Y*-axis reports seconds taken to find a solution. The *X*-axis reports the number of objects to be stocked. Both methods reported 100% success for the two- and four-object problems. Greedy reported 100% success for six objects. (a) Shelves - scenario. (b) Shelves - results.

solution options with incompatible transition histories (e.g., robot 1 starting with blocks 1 or 2 when both need to be handed off to robot 2). Greedy-MM-DaSH alleviates this issue by building a single transition history and is able to plan for 20 objects for the two-robot scenario and 16 objects for four-robot scenario. SMART is only able to obtain a 70% success on the random scenario for two robots, six objects, and a 0% on the same cross scenario.

MM-DaSH produces lower cost solutions than either method, although the Greedy-MM-DaSH solution cost is often close (see Table I). SMART produces much higher cost solutions despite its much denser representation (using PRM* roadmaps). This is due to both the requirements that all actions be performed synchronously and quality of the initial motion paths found using the dRRT* techniques (initial paths are used, as we consider the initial solution produced by SMART). In Table I, we can see that the construction time for SMART is an order of magnitude greater than the DaSH variants.

The cross scenario presents a harder problem as local decisions for individual objects often conflict with the best global decision for which the object should be passed across the formation first. This shows up in the hyperpath query stage for the DaSH variants and in the MAPF heuristic for SMART. The variance is much higher for MM-DaSH and SMART in the random scenario, as the more the problem resembles the cross scenario, the higher the difficulty and increased planning time. Greedy-MM-DaSH does not exhibit this as it greedily selects an object to pass all the way to its goal in either scenario, ignoring the best global decisions. In either scenario, most of the MM-DaSH and Greedy-MM-DaSH variances come from the conflict resolution stage and are a product of the sampled transitions and roadmaps available to resolve conflicts.

D. Shelving

To evaluate the ability of the method to account for geometrically constrained manipulation problems, we consider a shelf stocking scenario with two rows of objects [see Fig. 13(a)]. In this scenario, two manipulators must stock a shelf where the target configuration of objects on the shelf is random placements of object pairs such that one object lies immediately behind the other on the shelf, completely blocked by the front object. This forces the method to reason over the collision-based scheduling constraints and determine the correct order to place the objects on the shelf. The initial configuration of the objects is randomly generated on the same flat surface the robots rest on. The results are shown in Fig. 13(b).

1) Analysis: The planning times, as shown in Fig. 13(b), illustrate the ability of both DaSH variants to efficiently find plans for this geometrically constrained problem, placing each green object before placing its coupled blue object [see Fig. 13(b)]. Greedy successfully finds plans for more objects than MM-DaSH. They do take longer than the sorting scenarios, as it often takes multiple iterations for the motion planning layer to identify each of the motion-based scheduling constraints.

IX. Conclusion

In this article, we presented the DaSH method, a general task and motion planning framework, which considers varying levels of coupled and decoupled spaces. The framework enabled the tailoring of the space coupling to the coordination required for different problem stages rather than a static tradeoff between coordination and planning speed. We provided a decomposition of task and motion planning spaces into subspaces and presented a hypergraph representation capable of representing the decomposed space. We presented a method for leveraging this hypergraph representation to solve task and motion planning problems.

We illustrated the application of the general approach to the MRMP-DaSH and the MM-DaSH, provided the analysis of the representation for the multimanipulator problem, and empirically evaluated our approach against a state-of-the-art rearrangement planning method, where we showed significant improvement in planning. This improvement in planning was in large part due to the ability to construct the hypergraph-based representation. This is not the case for all task and motion planning problems. Future work should consider how to reason over implicit representations of the decomposed space.

Acknowledgment

The authors would like to thank the other members of the Parasol Lab at the University of Illinois Urbana-Champaign (UIUC) for their suggestions regarding this work, in particular, J. Ngui and J. B. Love for their input and H. Lee for the creation of the figures in this article.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

REFERENCES

[1] J. T. Schwartz and M. Sharir, "On the 'piano movers' problem—II: General techniques for computing topological properties of real algebraic manifolds," *Adv. Appl. Math.*, vol. 4, no. 3, pp. 298–351, 1983.

[2] J. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA, USA: MIT Press, 1988.

[3] J. Motes, R. Sandström, H. Lee, S. Thomas, and N. M. Amato, "Multi-robot task and motion planning with subtask dependencies," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3338–3345, Apr. 2020.

[4] K. Brown, O. Peltzer, M. A. Sehr, M. Schwager, and M. J. Kochenderfer, "Optimal sequential task assignment and path finding for multi-agent robotic assembly planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 441–447.

[5] C. Henkel, J. Abbenseth, and M. Toussaint, "An optimal algorithm to solve the combined task allocation and path finding problem," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4140–4146.

[6] C. R. Garrett, T. Lozano-Parez, and L. P. Kaelbling, "FFRob: An efficient heuristic for task and motion planning," in *Algorithmic Foundations of Robotics XI*. Berlin, Germany: Springer, 2015, pp. 179–195.

[7] K. Hauser and J.-C. Latombe, "Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries," in *Proc. Workshop Bridging Gap Between Task Motion Plan.*, 2009.

[8] A. Dobson and K. E. Bekris, "Planning representations and algorithms for prehensile multi-arm manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 6381–6386.

[9] R. Shome and K. E. Bekris, "Anytime multi-arm task and motion planning for pick-and-place of individual objects via handoffs," in *Proc. Int. Symp. Multi-Robot Multi-Agent Syst.*, 2019, pp. 37–43.

[10] R. Shome and K. E. Bekris, "Synchronized multi-arm rearrangement guided by mode graphs with capacity constraints," in *Proc. Int. Workshop Algorithmic Foundations Robot.*, 2020, pp. 243–260.

[11] I. Solis, J. Motes, R. Sandström, and N. M. Amato, "Representation-optimal multi-robot motion planning using conflict-based search," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4608–4615, Jul. 2021.

[12] L. Cohen, T. Uras, T. S. Kumar, and S. Koenig, "Optimal and bounded-suboptimal multi-agent motion planning," *Proc. 12th Annu. Symp. Combinatorial Search*, vol. 10, no. 1, pp. 44–51, 2019.

[13] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979.

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

[15] G. Sanchez and J.-C. Latombe, "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, vol. 2, pp. 2112–2119.

[16] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," *Int. J. Robot. Res.*, vol. 35, no. 5, pp. 501–513, 2016.

[17] M. Saha and P. Isto, "Multi-robot motion planning by incremental coordination," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 5960–5963.

[18] C. R. Garrett et al., "Integrated task and motion planning," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 4, pp. 265–293, 2021.

[19] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *Int. J. Robot. Res.*, vol. 29, no. 7, pp. 897–915, 2010.

[20] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *Int. J. Robot. Res.*, vol. 30, no. 6, pp. 678–698, 2011.

[21] J. Barry, L. P. Kaelbling, and T. Lozano-Pérez, "A hierarchical approach to manipulation with diverse actions," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 1799–1806.

[22] R. Shome, "The problem of many: Efficient multi-arm, multi-object task and motion planning with optimality guarantees," Ph.D. dissertation, School Graduate Studies, Rutgers, State Univ. New Jersey, New Jersey, NJ, USA, 2020.

[23] I. Umay, B. Fidan, and W. Melek, "An integrated task and motion planning technique for multi-robot-systems," in *Proc. IEEE Int. Symp. Robot. Sensors Environ.*, 2019, pp. 1–7.

[24] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Auton. Robots*, vol. 44, no. 3, pp. 443–467, 2020.

[25] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki, "A general task and motion planning framework for multiple manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 3168–3174.

[26] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 239–252, Feb. 2023.

[27] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, "Directed hypergraphs and applications," *Discrete Appl. Math.*, vol. 42, no. 2/3, pp. 177–201, 1993.

[28] G. Wagner and H. Choset, "Subdimensional expansion for multi-robot path planning," *Artif. Intell.*, vol. 219, pp. 1–24, 2015.

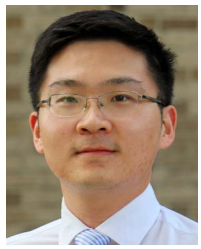
[29] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 7643–7650.

[30] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Robotics: Science and Systems VI*. Cambridge, MA, USA: MIT Press, 2011.

[31] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 26, pp. 563–569, 2021.

James Motes received the undergraduate degree in computer engineering, in 2018, and the master's degree in computer science from Texas A&M University, College Station, TX, USA, 2019. He is currently working toward the Ph.D. degree in computer science with Computer Science Department, University of Illinois at Urbana-Champaign, Champaign, IL, USA, affiliated with Coordinated Science Lab.

His research interests include task and motion planning, multi-robot systems, and autonomous factories.



Tan Chen (Member, IEEE) received the B.S. degree in mechanical engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2013, the dual M.S. degrees in automation Ecole des Mines de Douai, Douai, France, in 2014, and in mechatronics engineering from SJTU, Shanghai, China, in 2016, and the Ph.D. degree in aerospace and mechanical engineering from the University of Notre Dame, Notre Dame, IN, USA, 2021.

He is an Assistant Professor with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI, USA. Prior to joining Michigan Tech, he was a Postdoctoral Research Associate affiliated with Coordinated Science Laboratory, University of Illinois Urbana-Champaign. His research interests include nonlinear control, applied mechanics, and AI as applied to robotic systems.

Timothy Bretl (Senior Member, IEEE) received the B.S. degree in engineering and the B.A. degree in mathematics from Swarthmore College, Swarthmore, PA, USA, in 1999, and the M.S. and Ph.D. degrees both in aeronautics and astronautics from Stanford University, Stanford, CA, USA, in 2000 and 2005, respectively.

Subsequently, he was a Postdoctoral Fellow with the Department of Computer Science, Stanford University. Since 2006, he has been with the University of Illinois at Urbana-Champaign, Champaign, IL, USA, where he is a Severns Faculty Scholar, an Associate Professor of aerospace engineering, and a Research Associate Professor with Coordinated Science Laboratory.

Dr. Bretl was a recipient of the National Science Foundation Faculty Early Career Development Award in 2010, numerous teaching awards at Illinois, including the AIAA Student Chapter Teacher of the Year Award in 2015, both the William L. Everett Award for Teaching Excellence and the Rose Award for Teaching Excellence in 2016, and both the College of Engineering Teaching Excellence Award and the Campus Award for Excellence in Undergraduate Teaching in 2018.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2024, Yokohama, Japan. Cite as T-RO paper.

Marco Morales Aguirre received the B.S. degree in computer engineering and the M.S. degree in electrical engineering from Universidad Nacional Autonoma de Mexico (UNAM), Mexico City, Mexico, in 1996 and 1998, respectively, and the Ph.D. degree in computer science from Texas A&M University, College Station, TX, USA, in 2007.

He is an Associate Professor with the Department of Computer Science, Instituto Tecnológico Autónomo de México, Mexico City, Mexico, and a Teaching Associate Professor with the Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL, USA. He is a National Researcher (Level I) within the National System of Researchers of Mexico and a member of the Mexican Academy of Computing. He has also been a Visiting Professor with Texas A&M University and a Lecturer with UNAM. His research interests are in motion planning and control for autonomous robots, artificial intelligence, machine learning, and computational geometry.

Nancy M. Amato (Fellow, IEEE) received the undergraduate degree in mathematical sciences and economics from Stanford University, Stanford, CA, USA, in 1986, and the M.S. degree in computer science from University of California, Berkeley, CA, USA, and Ph.D. degree in computer science from the University of Illinois Urbana-Champaign, Champaign, IL, USA, in 1988 and 1995, respectively.

She is the Head of Computer Science Department and Abel Bliss Professor of engineering with the University of Illinois at Urbana-Champaign, Champaign, IL, USA. Before returning to her alma mater in 2019, she was a Unocal Professor and Regents Professor with Texas A&M University and the Senior Director of Engineering Honors Programs. She is known for algorithmic contributions to robotics task and motion planning, computational biology and geometry, and parallel and distributed computing.

Dr. Amato was a recipient of the inaugural Robotics Medal in 2023, the 2019 IEEE RAS Saridis Leadership Award in Robotics and Automation, the inaugural 2014 NCWIT Harrold/Notkin Research and Graduate Mentoring Award, and the 2014 CRA Haberman Award for contributions to increasing diversity in computing. She is a Fellow of the AAAI, AAAS, and ACM.