Lazy-DaSH: Lazy Approach for Hypergraph-based Multi-robot Task and Motion Planning

Seongwon Lee[1], James Motes[1], Isaac Ngui[1], Marco Morales[1,2], and Nancy M. Amato[1]

I. Introduction

Multi-robot task and motion planning (MR-TMP) is challenging due to the exponential expansion in the size of the planning space as the number of robots and tasks increases. Decoupled methods, which attempt to plan in individual robot state spaces, achieve a linear scaling in the size of the search space with respect to the number of robots but are incomplete and often fail to plan for problems requiring coordination [1], [2]. Coupled methods directly consider the composite space, achieving coordination at the cost of high computation effort, thus limiting them to smaller problem sizes [3], [4]. Hybrid approaches consider varying compositions of the state space and leverage the strengths of both coupled and decoupled methods [5]–[7].

A recent hybrid method, **D**ecomposable St**a**te **S**pace **H**ypergraph (DaSH) [8], utilizes a hypergraph-based representation to model the changes in state space composition corresponding to robot/object interactions (e.g. pick/place and handoff actions). This approach achieved two orders of magnitude faster planning time over comparable methods for multi-manipulator rearrangement planning problems. However, computing motion feasibility across the entire set of decoupled state space compositions captured by DaSH's hypergraph requires significant effort, most of which is not used in the final solution.

This extended abstract introduces Lazy-DaSH which further focuses computation effort by lazily computing motion feasibility only within the set of decoupled robot state space compositions that are potentially used in the solution, thus accelerating the construction and querying of the state space representation. Our preliminary results show that Lazy-DaSH can scale to twice the number of robots (up to 8 manipulators) and objects (up to 22 objects with 4 manipulators) compared to DaSH (up to 12 objects with 4 manipulators). It also achieves two orders of magnitude faster planning time for multi-manipulator rearrangement problems across three different scenarios.

II. The Lazy-DaSH Method

Lazy-DaSH builds on the DaSH method, which utilizes a hierarchical hypergraph-based model for MR-TMP problems (Figure 2). The hypergraph representation provides much more concise representations, allowing the representation size to remain manageable as the number of robots and objects grows.

[1] Parasol Lab, University of Illinois Urbana-Champaign, USA {sl148, jmotes2, ingui2, moralesa, namato}@illinois.edu
[2] ITAM, México. {marco.morales}@itam.mx.

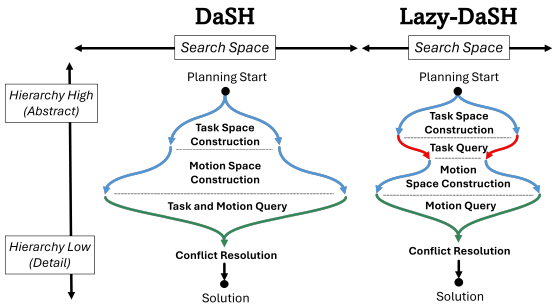

Fig. 1: Comparison of search space expansion: Blue arrows show expansion, while green (and red) arrows indicate narrowing. Lazy-DaSH introduces a task query phase, highlighted by red arrows.

A. Decomposable Task Space Hypergraph

The DaSH approach starts by constructing a hypergraph representation of the decoupled state space. This hypergraph encodes possible robot/object compositions as vertices, annotated with information such as grasp constraints, along with transitions between them (i.e., pick/place or handoff actions). The subsequent layer computes motion feasibility for all transitions encoded in the task space representation. This concise hypergraph representation allows faster query times for combined task and motion planning across decoupled state spaces, producing an *optimistic schedule* of motions and actions that satisfy the task. A final conflict resolution layer resolves any motion conflicts between paths in decoupled state spaces producing a final *valid schedule* or solution. Any failures at this layer are passed back to the query phase as scheduling constraints.

B. Lazy-Decomposable Task Space Hypergraph

The overall framework is illustrated in Figures 1 and 2. Lazy-DaSH differentiates itself from DaSH by employing a hierarchical query framework: high-level task query within the *task plan layer* and subsequent low-level motion query within the *motion plan layer*. Each layer includes a representation construction phase and a query phase over respective task/motion plan. This allows the motion feasibility to be *lazily* computed only within the robot state spaces and the transitions indicated by the task plan instead of the exhaustive motion feasibility computed by DaSH.

1) Task Plan Layer: The task plan layer captures the most abstract level of information about interactions between robots and objects through the *task space hypergraph*. The task space hypergraph contains start and goal information, enabling the query process to find a valid transition history

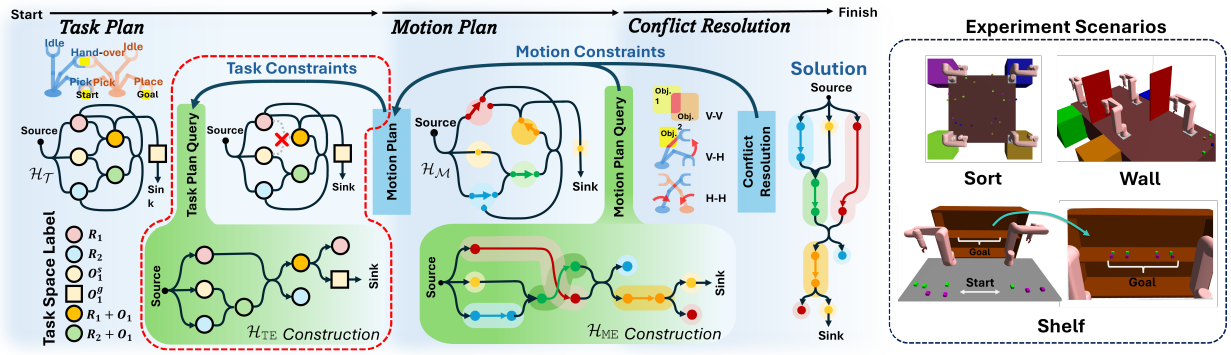

Fig. 2: Illustration of Lazy-DaSH and experiment scenarios. The left figure shows two manipulators (R_1 and R_2) rearranging a single object (O_1). The distinguishing feature of Lazy-DaSH compared to DaSH is highlighted with a dashed red line in the task plan phase.

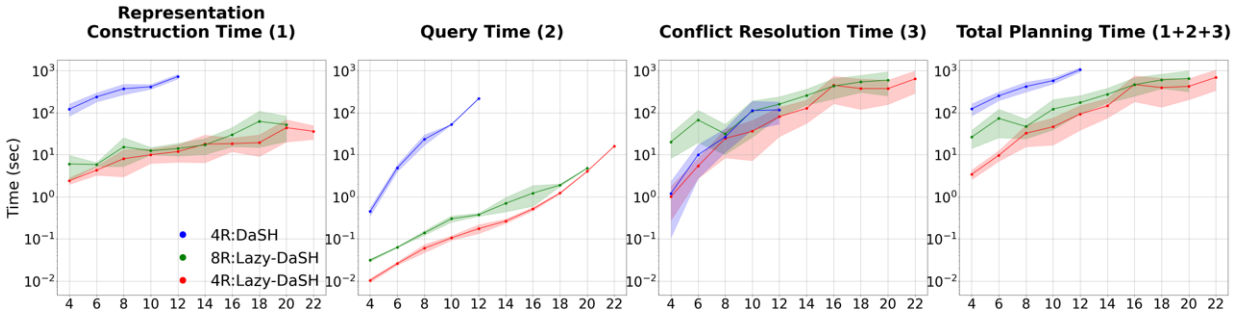

Fig. 3: The exemplary experiment results of the Sort scenario with up to 8 manipulators and an increasing number of objects.

connecting these states. The task plan is queried by constructing the *task-extended hypergraph*, which involves expanding the task space hypergraph. This results in an *unvalidated schedule* since the task query phase does not encode the motion feasibility within the state spaces.

2) Motion Plan Layer: The motion plan layer captures motion details through the *motion hypergraph* and queries the motion plan through the *motion-extended hypergraph*. Motion feasibility is *lazily* validated during the motion query phase, generating an *optimistic schedule*. By constructing motion details only for the relevant state spaces as informed by the task plan, the motion representation size remains compact (red arrows and subsequent blue arrows in Figure 1).

3) Conflict Resolution Layer: The conflict resolution layer aims to address any conflicts in the *optimistic schedule*, producing a final *valid schedule*. Detected constraints are passed back to the task or motion plan layers, triggering replanning or expanding the representations.

III. Validation

We evaluated Lazy-DaSH on the multi-manipulator rearrangement problem, where manipulators transport blocks from randomly generated start positions to goal positions through pick, place, and hand-over operations. We designed three scenarios to show our algorithm's capabilities: Sort, Wall, and Shelf, as shown in Figure 2. The Sort scenario demonstrates the algorithm's scalability by having robots sort objects into boxes. The Wall and Shelf scenarios highlight its ability to handle the constraint feedback. Since the task planning layer only encodes abstract interactions without geometric details, the subsequent motion planning layer must detect infeasible actions and inform the upper planning layers of the infeasibility. In the Wall scenario, walls prevent some robot interactions, necessitating a task and motion plan that circumvents these obstacles. Similarly, the Shelf scenario demands specific sequencing of tasks to comply with geometric constraints.

Preliminary results demonstrating the scalability of the algorithm are presented in the Sort scenario, where up to 8 manipulators sort objects, as shown in Fig. 3. By expanding and adding motion details to only the state spaces composing the current task plan, the representation construction time cost is two orders of magnitude lower than in DaSH. The smaller representation size enables the query phases to manage up to 22 objects with 4 manipulators and 20 objects with 8 manipulators. In contrast, DaSH can only scale up to 12 objects with 4 manipulators. This improvement allows for task and motion query times in Lazy-DaSH up to two orders of magnitude faster than DaSH. The conflict resolution time is similar between DaSH and Lazy-DaSH, but the significant reduction in the construction and query time enhances overall planning scalability and planning time.

Acknowledgement

This work was supported in part by the U.S. National Science Foundation's "Expeditions: Mind in Vitro: Computing with Living Neurons" under award No. IIS-2123781, and by the IBM-Illinois Discovery Accelerator Institute and the Center for Networked Intelligent Components and Environments (C-NICE) at the University of Illinois.

REFERENCES

[1] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 430–435.

[2] K. Hauser and J.-C. Latombe, "Integrating task and prm motion planning: Dealing with many infeasible motion planning queries," in *ICAPS09 Workshop on Bridging the Gap between Task and Motion Planning*. Citeseer, 2009.

[3] R. Shome and K. E. Bekris, "Synchronized multi-arm rearrangement guided by mode graphs with capacity constraints," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 243–260.

[4] I. Umay, B. Fidan, and W. Melek, "An integrated task and motion planning technique for multi-robot-systems," in *2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*. IEEE, 2019, pp. 1–7.

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

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

[7] L. Cohen, T. Uras, T. S. Kumar, and S. Koenig, "Optimal and bounded-suboptimal multi-agent motion planning," in *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[8] J. Motes, T. Chen, T. Bretl, M. M. Aguirre, and N. M. Amato, "Hypergraph-based multi-robot task and motion planning," *IEEE Transactions on Robotics*, 2023.