2022 IEEE International Conference on Automation Science and Engineering

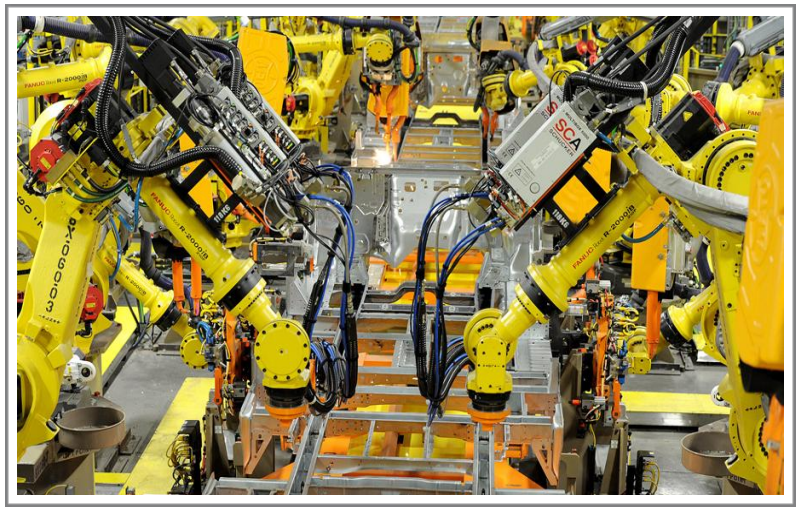
A MIP-based Approach for Multi-Robot Geometric Task-and-Motion Planning

Hejia Zhang, Shao-Hung Chan, Jie Zhong,
Jiaoyang Li, Sven Koenig, Stefanos Nikolaidis

ICAROS lab
University of Southern California

Manipulation tasks with multi-robot systems

- Multi-robot systems can perform manipulation tasks more efficiently than single-robot systems.
- Multi-robot systems can perform manipulation tasks that are beyond the capabilities of single-robot systems.



GM assembly plant, Lake Orion (Image by: ASSEMBLY magazine)

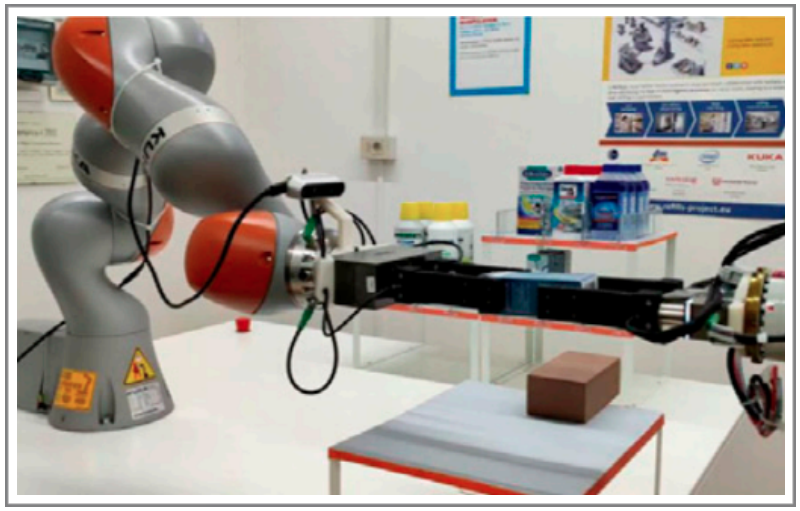
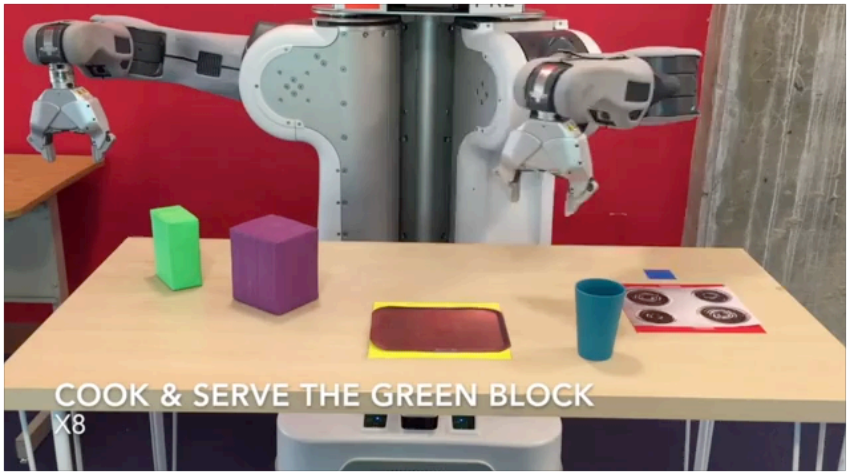


Image by: Costanzo et al. 2021

We want to generate efficient and executable multi-robot task plans and motion trajectories.

Task-and-Motion Planning (TAMP)

TAMP is the problem of dividing an objective into a series of robot-executable motion trajectories.



Cooking task (Garrett et al. 2020)

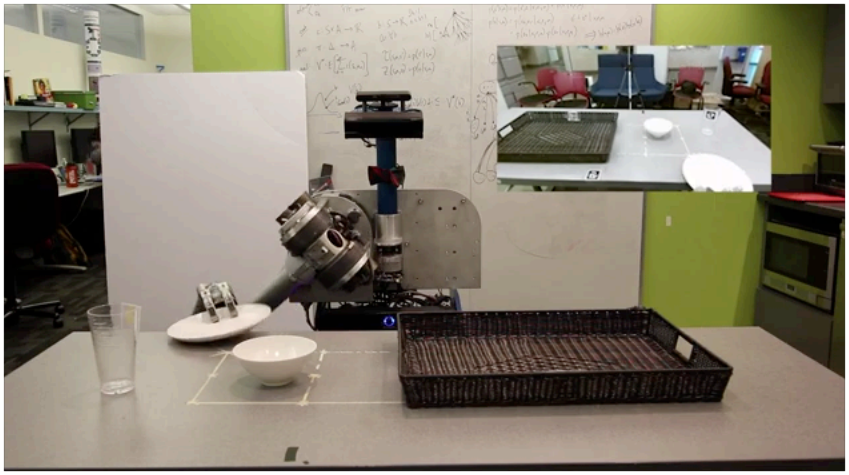
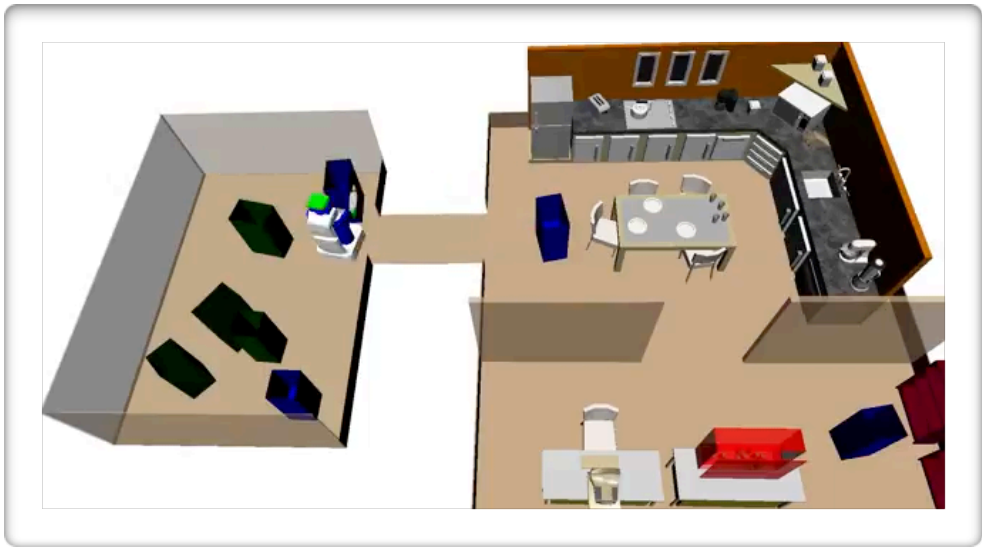


Table clearing task (King et al. 2015)

Geometric Task-and-Motion Planning (GTAMP)

GTAMP is an important subclass of TAMP.

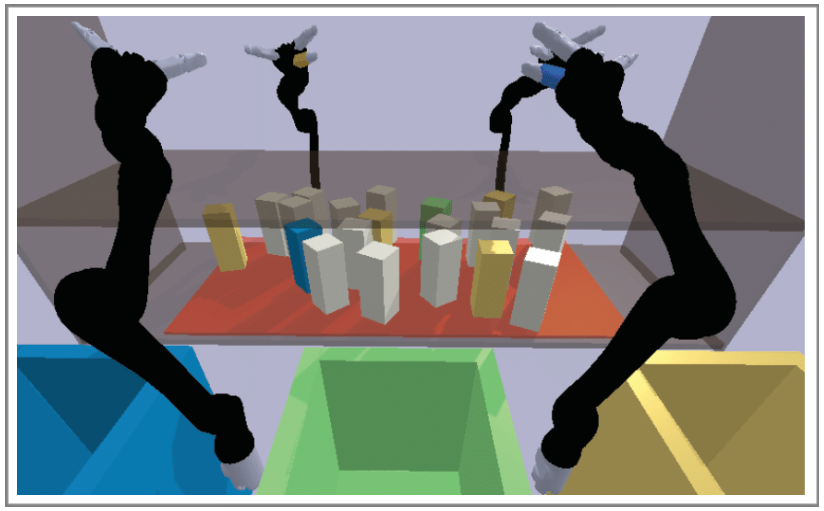


Box moving task (Kim et al. 2022)

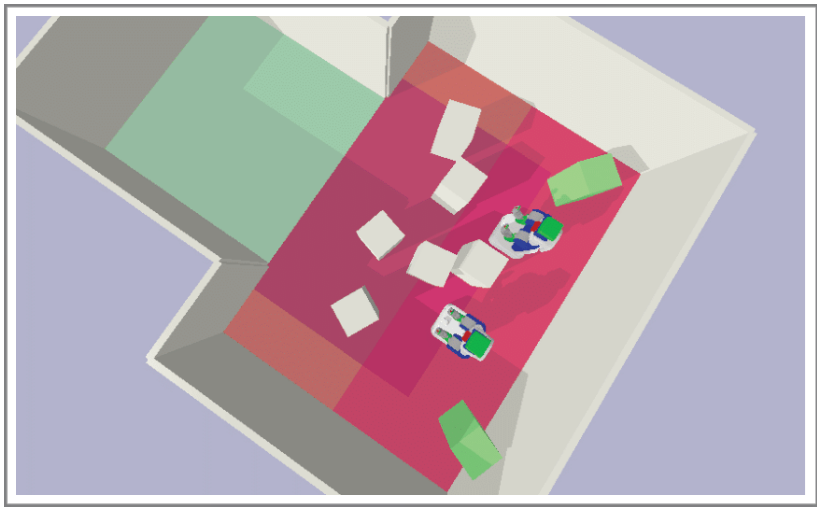
Packaging task (Kim et al. 2022)

Multi-Robot GTAMP (MR-GTAMP)

We extend GTAMP to multi-robot domains.



Packing colored objects into boxes.

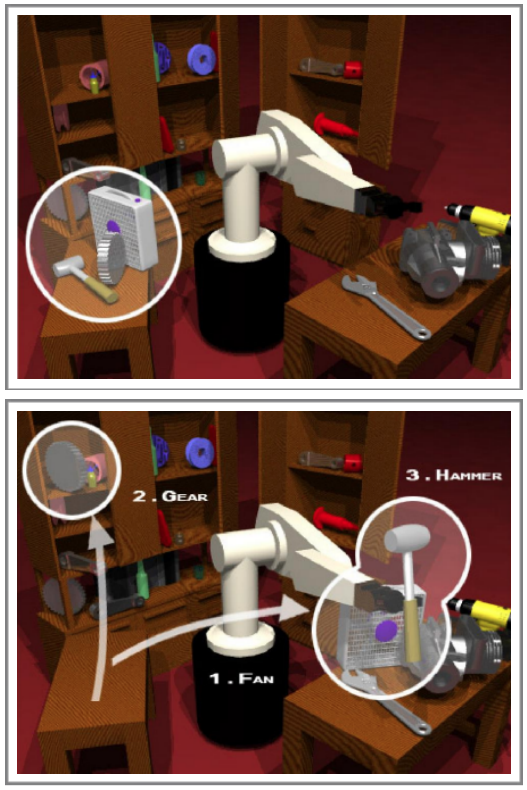


Moving the colored boxes to the green region.

Robots can collaborate to perform GTAMP tasks more efficiently.

Prior work: Single-Robot GTAMP

Retrieving the hammer



Stilman et al. 2007

Use the sampled future motions as constraints,
to identify new positions to which to relocate obstacles.

- Start from moving the goal object
- Move the blocking objects out of the way

Limitation: The object placements are only constrained by sampled future actions.

Prior work: Learning for Single-Robot GTAMP

Example GTAMP tasks

Kim et al. 2019

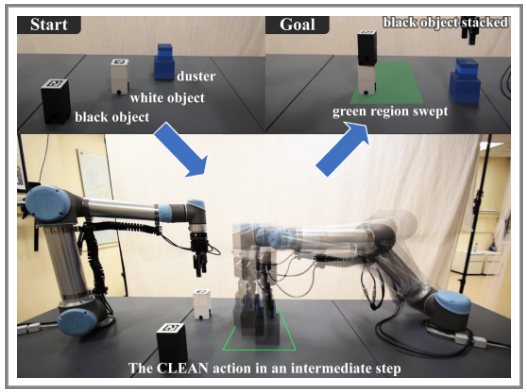
Learning to guide discrete action selection for GTAMP,
by representing states with robot manipulation information.

- Perform motion planning to identify obstacles for picking and placing each objects
- Reason about the collected information with learned neural networks

Limitation: Do not plan collaboration strategies in multi-robot domains.

Prior work: General MR-TAMP

A clean task



Pan et al. 2021

Use an intermediate layer for robot task scheduling,
to reduce the number of task planning calls.

- Formulate task planning as constraint satisfaction problems
- Update task planning constraints from motion planning failures

Limitation: Do not have guidance to search for object placements efficiently.

Contribution

To generate effective, collaborative task-and-motion plans for multi-robot systems to perform GTAMP tasks:

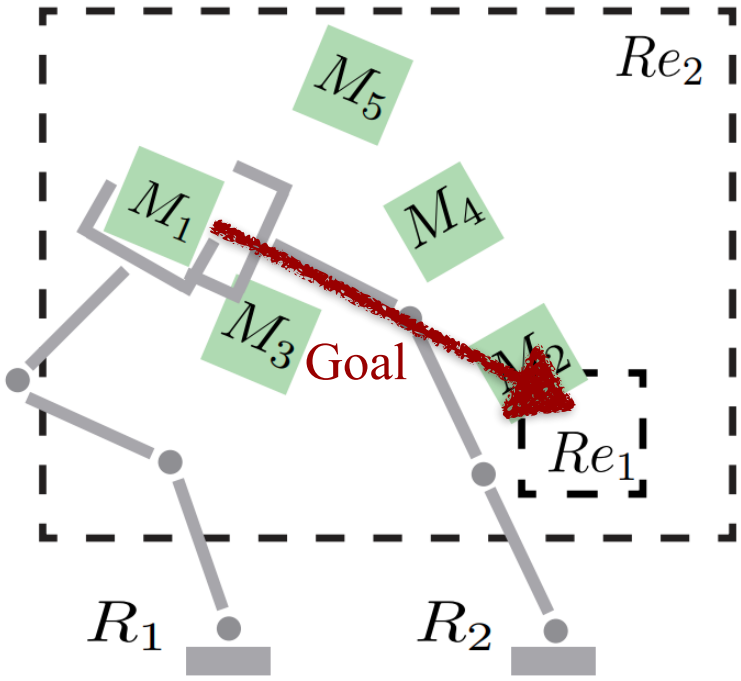
1. **Task-skeleton generating**:
   - *Which robot should move which objects?*
2. **Task-skeleton grounding**:
   - *Which locations should objects be moved to?*
3. **Combine generating and grounding with tree search**:
   - *How to search efficiently?*

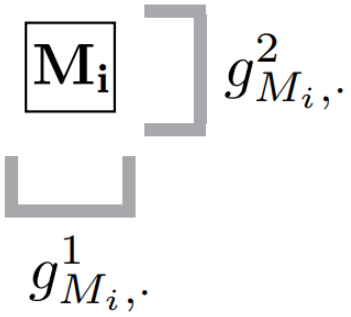
Contribution

To generate effective, collaborative task-and-motion plans for multi-robot systems to perform GTAMP tasks:

1. **Task-skeleton generating**:
   - *Which robot should move which objects?*
2. **Task-skeleton grounding**:
   - *Which locations should objects be moved to?*
3. **Combine generating and grounding with tree search**:
   - *How to search efficiently?*

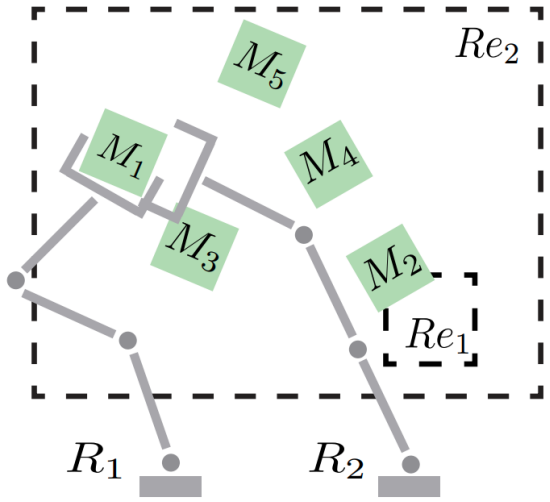
A 2D example MR-GTAMP task

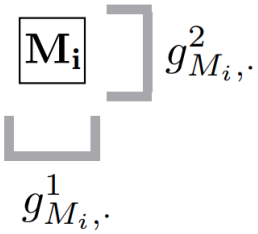




Compute collaborative manipulation information

- Given a MR-GTAMP problem instance, we compute the occlusion and reachability information for individual robots
  - "Object M_3 blocks robot R_2 from picking up object M_1"
  - "Robot R_1 can not reach region Re_1".
  - …

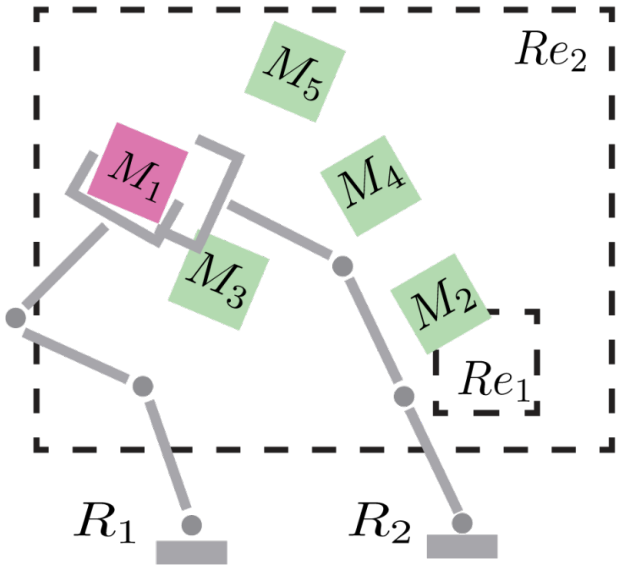




The computed information is useful for providing guidance for task-level search.

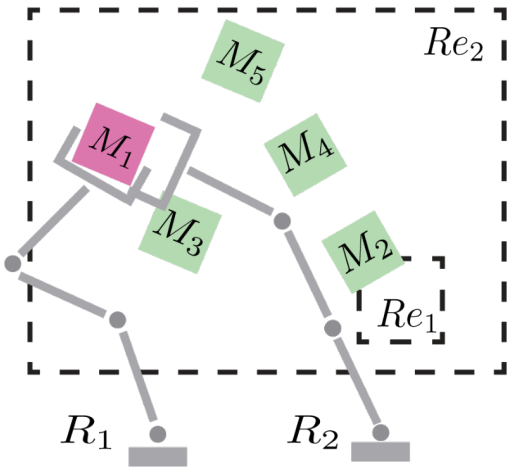
Generate collaborative task skeletons

- To generate task skeletons for moving object M_1.
  - We build a *collaborative manipulation task graph* (CMTG) to capture the relevant information about moving object M_1.
  - The CMTG supports us to formulate a series of mixed-integer linear programs (MIPs) to compute efficient task skeletons.



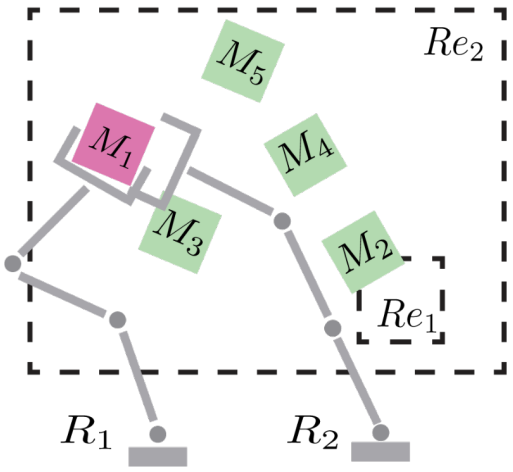
Build the CMTG

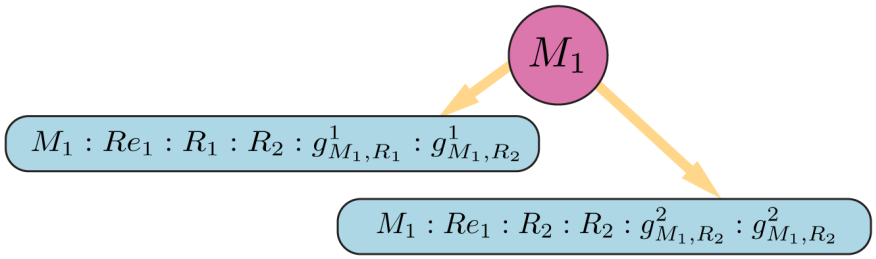
- To build the CMTG for moving object M_1 to region Re_1,
  - ➡ We first add object M_1 to an empty graph.
  - ◉ We then add actions that can move object M_1 to region Re_1.
  - ◉ We add the blocking objects for the added actions in a similar way.



Build the CMTG

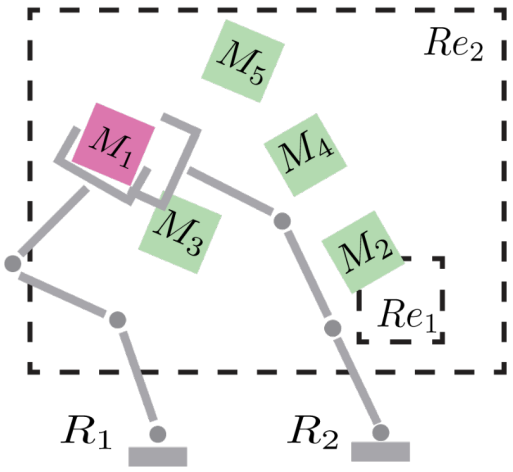
- To build the CMTG for moving object M_1 to region Re_1,
  - We first add object M_1 to an empty graph.
  - ➡ We then add actions that can move object M_1 to region Re_1.
  - We add the blocking objects for the added actions in a similar way.

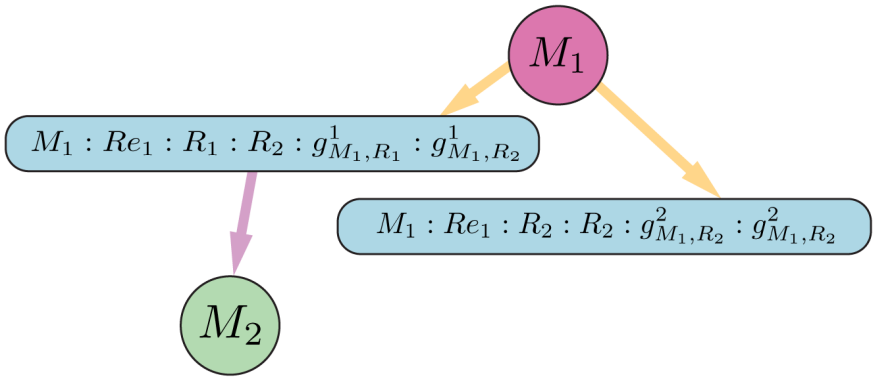




Build the CMTG

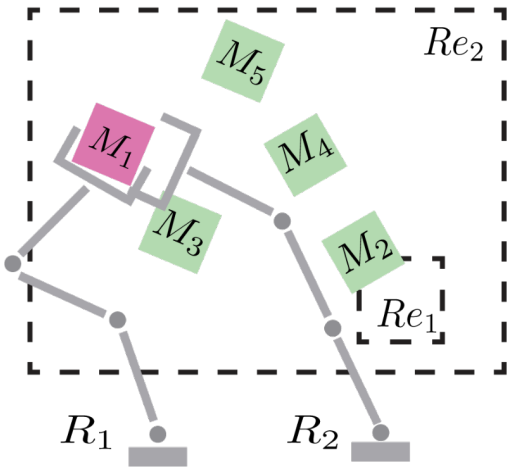
- To build the CMTG for moving object M_1 to region Re_1,
  - We first add object M_1 to an empty graph.
  - We then add actions that can move object M_1 to region Re_1.
  - ➡ We add the blocking objects for the added actions in a similar way.

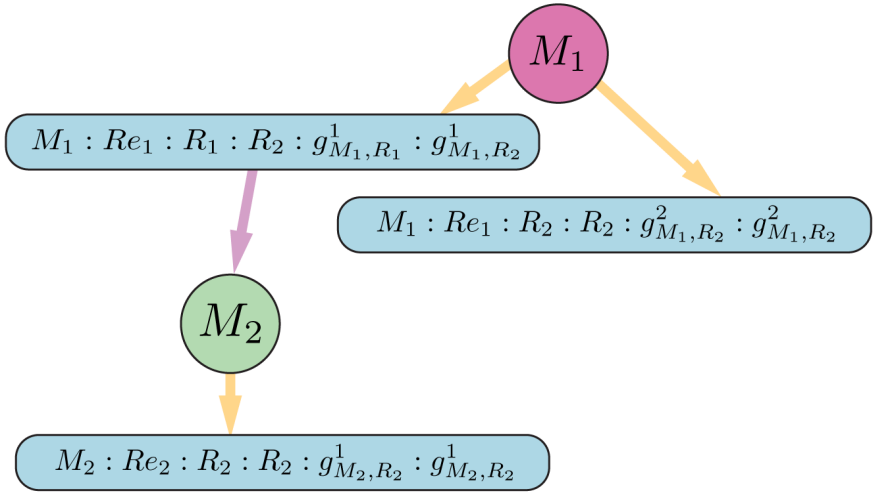




Build the CMTG

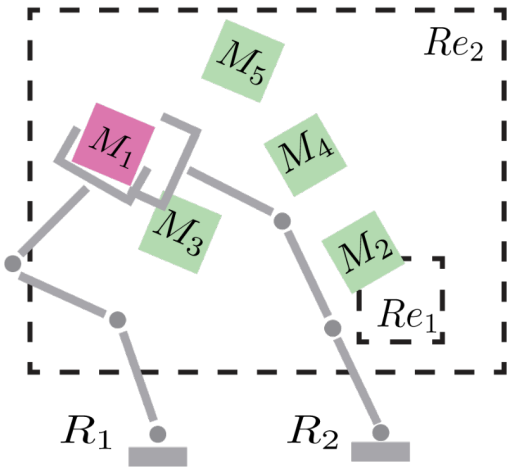
- To build the CMTG for moving object M_1 to region Re_1,
  - We first add object M_1 to an empty graph.
  - We then add actions that can move object M_1 to region Re_1.
  - ➡ We add the blocking objects for the added actions in a similar way.

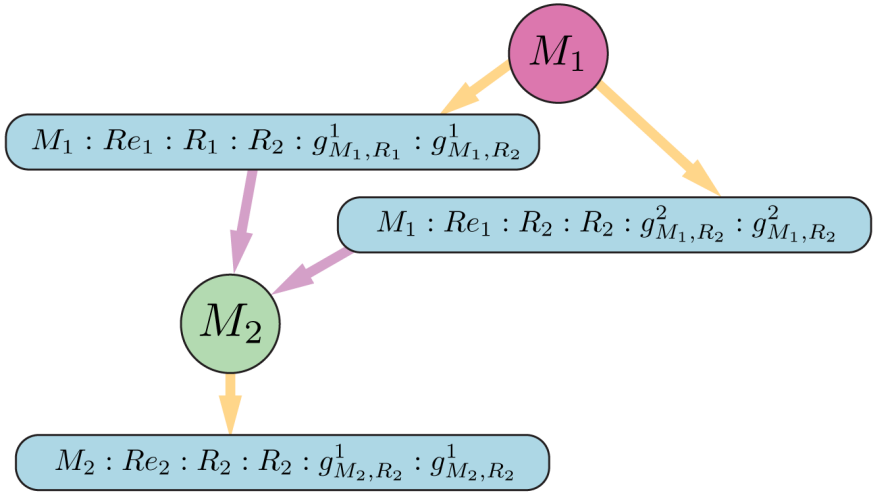




Build the CMTG

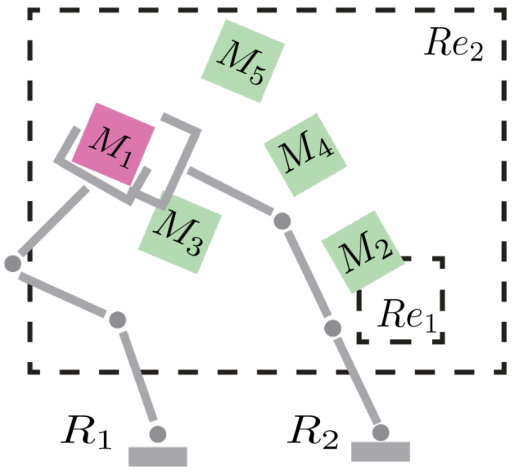
- To build the CMTG for moving object M_1 to region Re_1,
  - We first add object M_1 to an empty graph.
  - We then add actions that can move object M_1 to region Re_1.
  - ➡ We add the blocking objects for the added actions in a similar way.

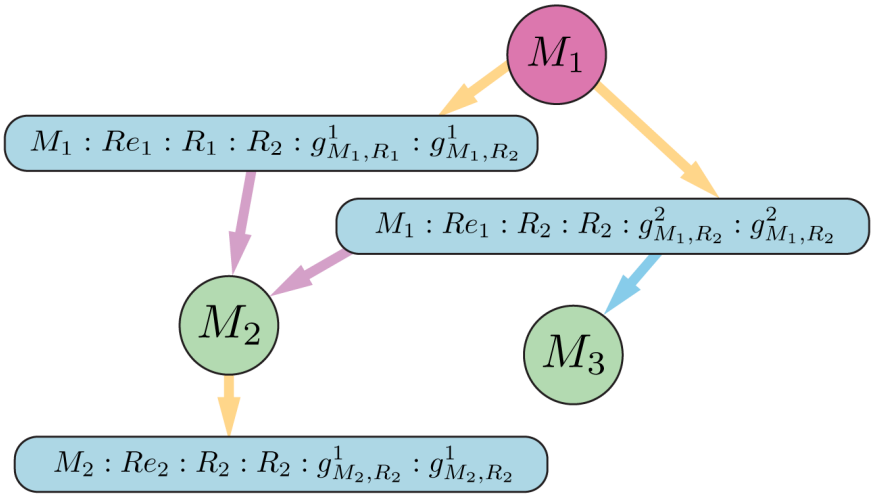




Build the CMTG

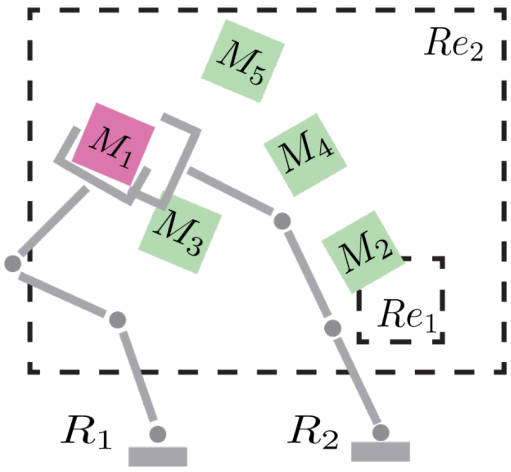
- To build the CMTG for moving object M_1 to region Re_1,
  - We first add object M_1 to an empty graph.
  - We then add actions that can move object M_1 to region Re_1.
  - ➡ We add the blocking objects for the added actions in a similar way.

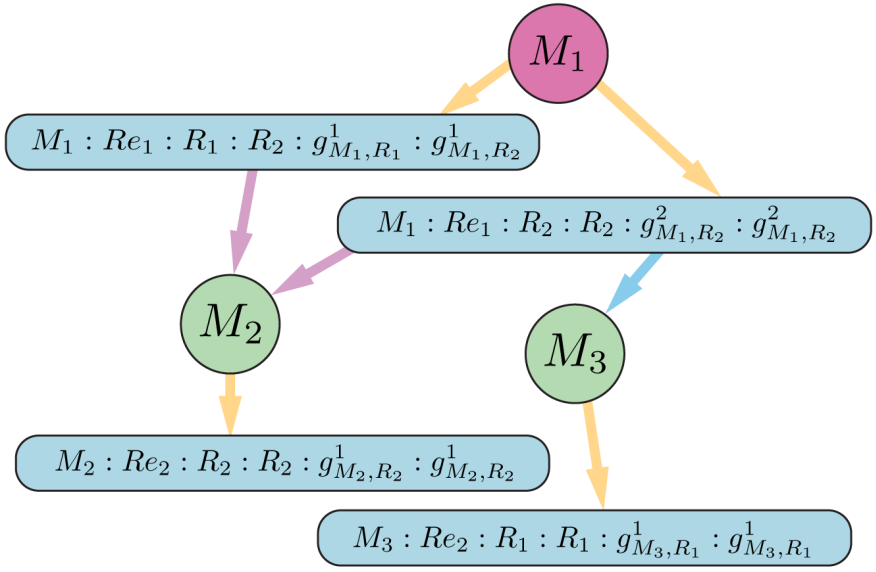




Build the CMTG

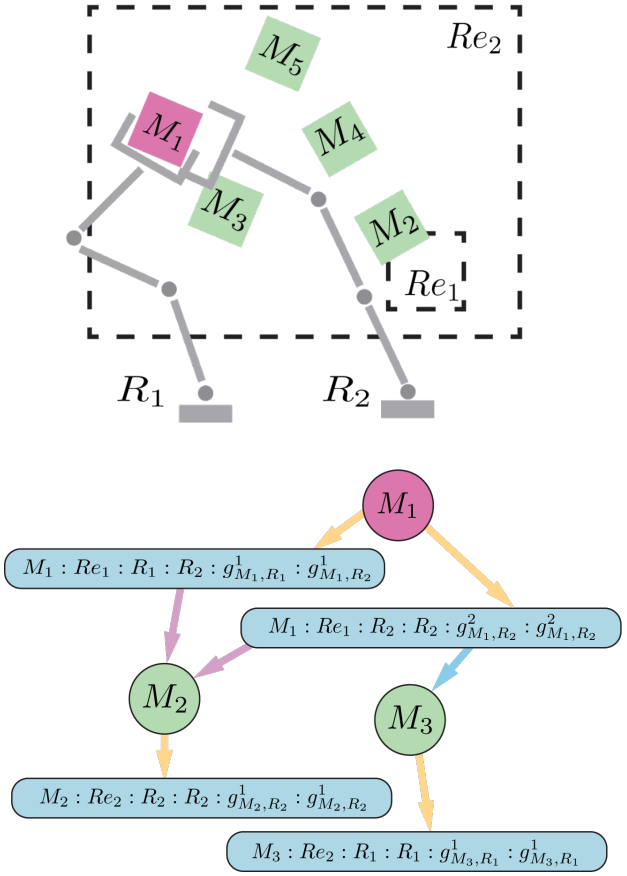
- To build the CMTG for moving object M_1 to region Re_1,
  - We first add object M_1 to an empty graph.
  - We then add actions that can move object M_1 to region Re_1.
  - ➡ We add the blocking objects for the added actions in a similar way.





MIP formulation and solving

- We get multiple task skeletons by formulating and solving MIPs.
  - Task skeleton 1
    - **Time step** 1: *Robot R_2 relocates object M_2 in region Re_2.*
    - **Time step** 2: *Robot R_1 picks object M_1 and hands it over to robot R_2. R_2 then places it to region Re_1.*
  - …



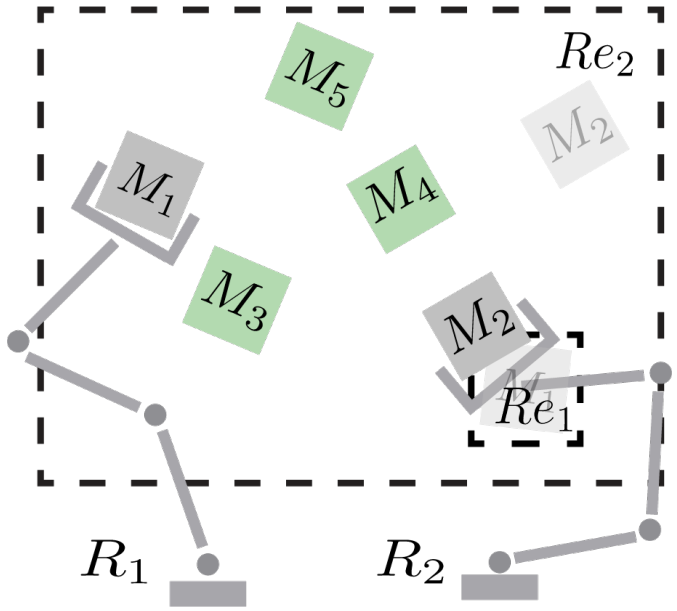
Contribution

To generate effective, collaborative task-and-motion plans for multi-robot systems to perform GTAMP tasks:

1. **Task-skeleton generating**:
   - *Which robot should move which objects?*
2. **Task-skeleton grounding**:
   - *Which locations should objects be moved to?*
3. **Combine generating and grounding with tree search**:
   - *How to search efficiently?*

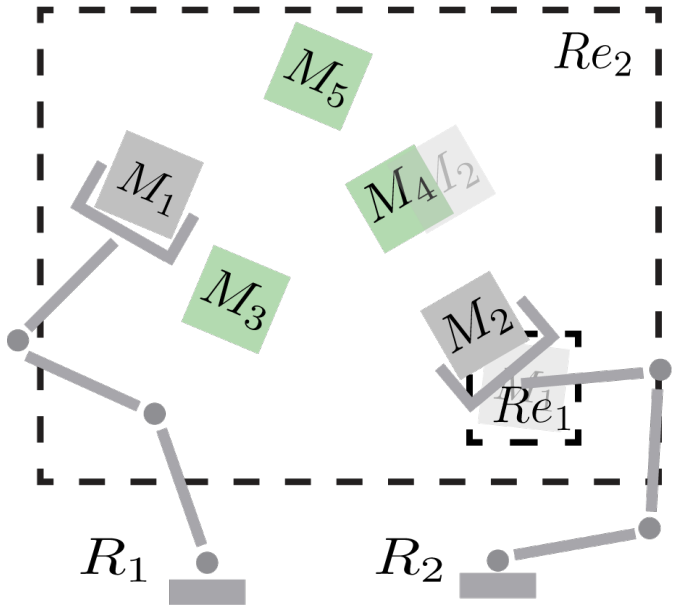
Task-skeleton grounding

- We sample object placements and robot motions to relocate objects.
  - We use reverse-search strategy.
  - We first treat objects that are not planned to be moved by the task skeleton as fixed obstacles.



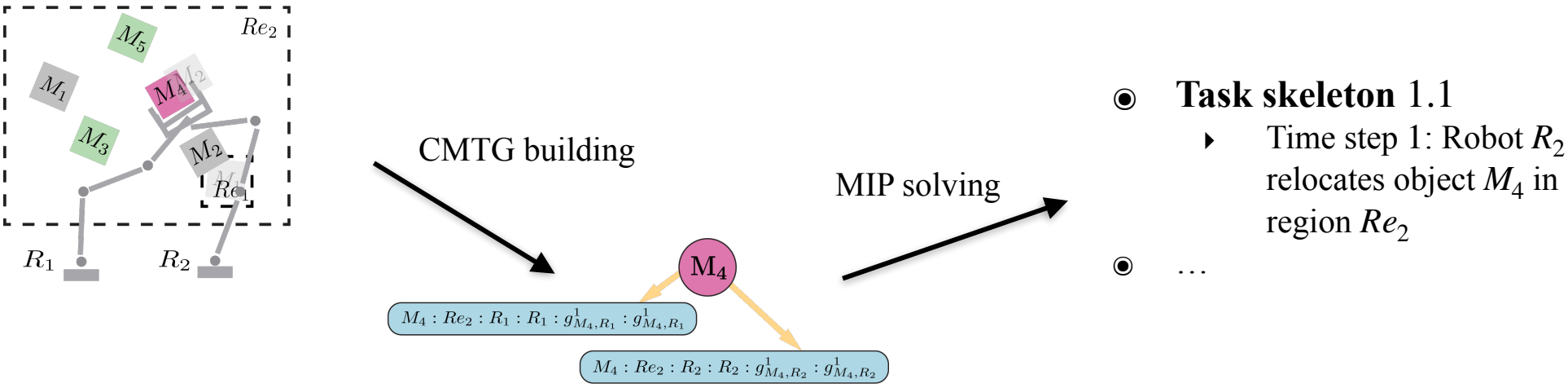
Task-skeleton grounding

- We sample object placements and robot motions to relocate objects.
  - We may fail at sampling object placements without colliding with objects that are not planned to be moved.
  - We then allow those objects to be collided with.



Task-skeleton grounding

- We sample object placements and robot motions to relocate objects.
  - We generate new task skeletons to move objects that are collided while are not planned to be moved in the original task skeleton.



- **Task skeleton** 1.1
  - Time step 1: Robot R_2 relocates object M_4 in region Re_2
- …

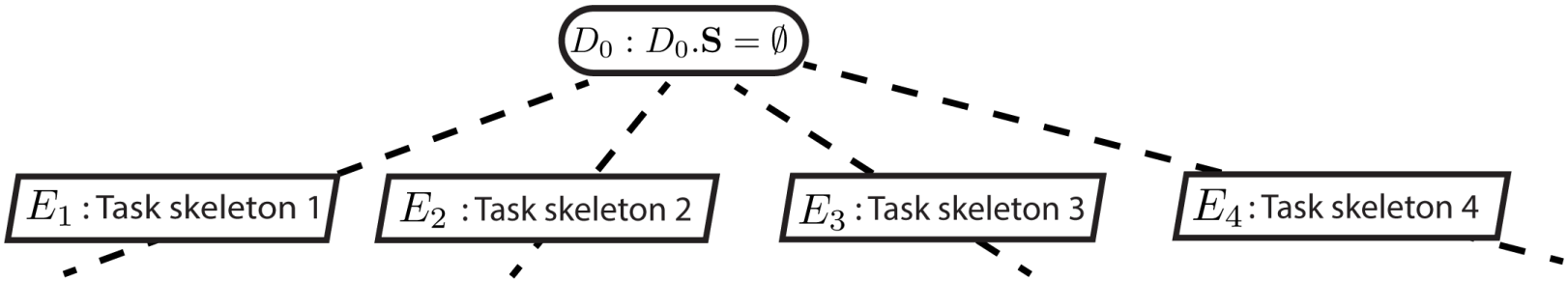
Contribution

To generate effective, collaborative task-and-motion plans for multi-robot systems to perform GTAMP tasks:

1. **Task-skeleton generating**:
   - *Which robot should move which objects?*
2. **Task-skeleton grounding**:
   - *Which locations should objects be moved to?*
3. **Combine generating and grounding with tree search**:
   - *How to search efficiently?*

Tree search for selecting task skeletons to ground

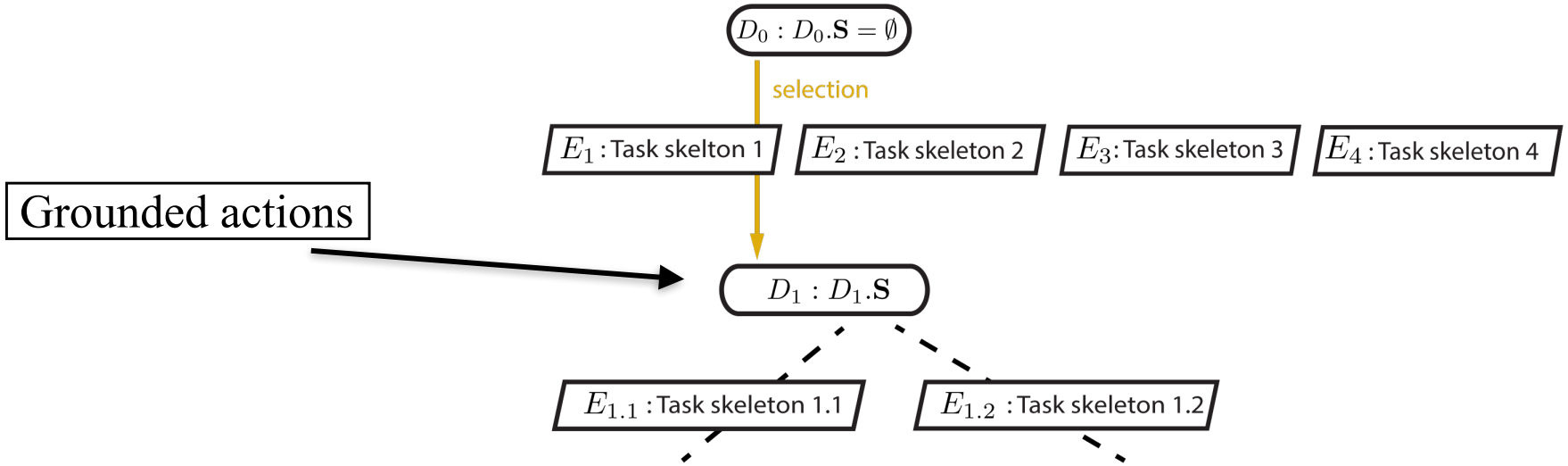
- After each task-skeleton grounding, we may get a new set of task skeletons.
  - *Which task skeleton should we try to ground next?*
- We build a search tree for selecting task skeletons to ground.



The initial search tree.

Tree search for selecting task skeletons to ground

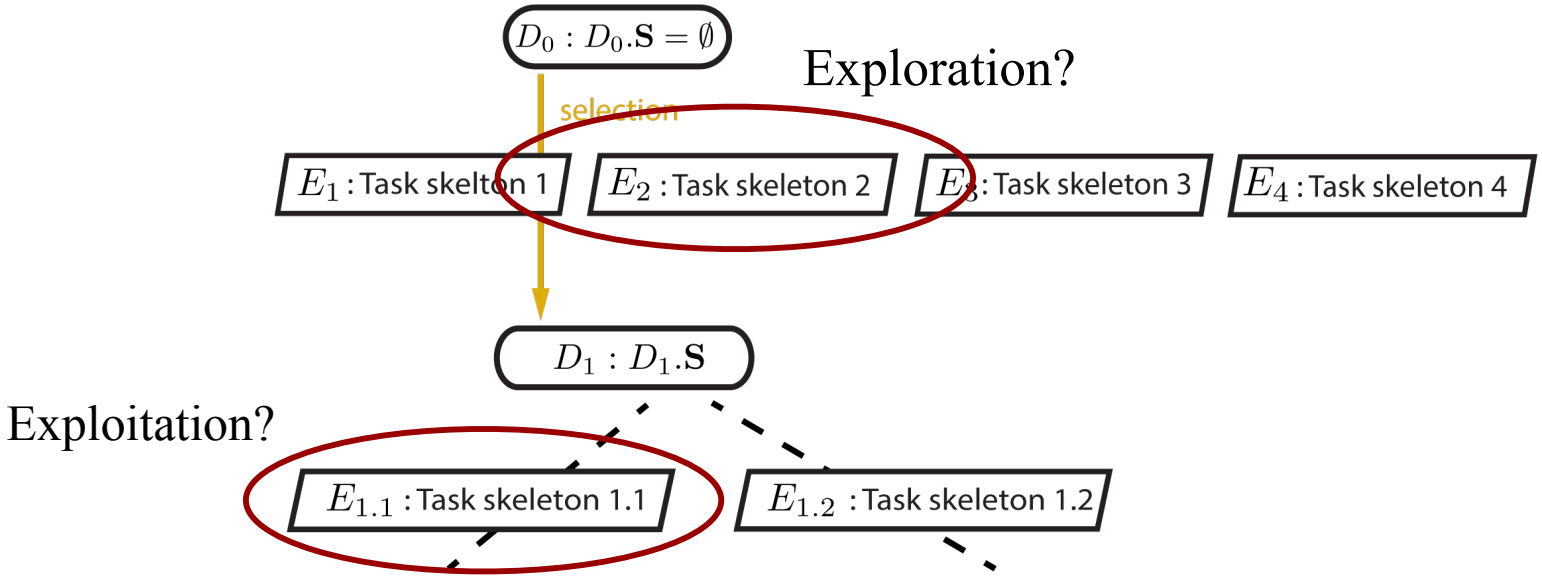
- We grow the search tree based on the results of task-skeleton groundings.



The result search tree after grounding task skeleton 1.

Tree search for selecting task skeletons to ground

- We achieve exploration and exploitation balance by computing upper confidence bound value for selecting each edges.

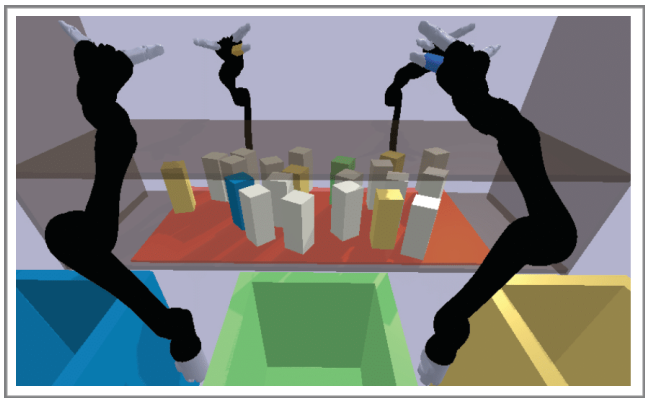


Experiments: Baselines

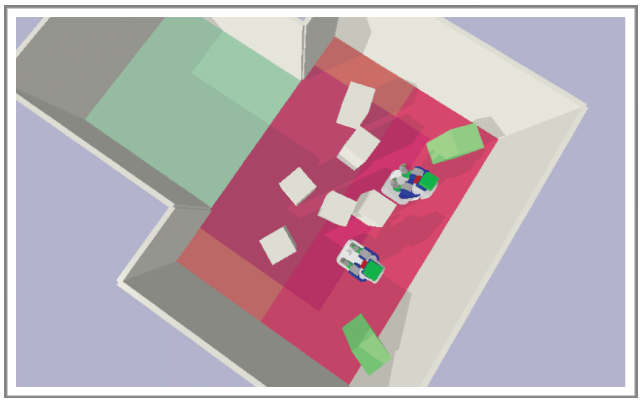
- We compare our framework with two state-of-the-art TAMP frameworks
  - **Ap1** is a multi-robot extension of the ResolveSpatialConstraints algorithm (Stilman et al. 2007) by assuming that the robots form a singe composite robot.
    - It may move objects that are unnecessary to be moved.
  - **Ap2** is a general MR-TAMP framework (Pan et al. 2021).
    - It is inefficient at sampling feasible object placements.

Experiments: Benchmark domains

- We evaluate our method and baselines in two benchmark domains.
  - **Packaging (PA):**
    - The goal is to pack colored objects into boxes with Kinova Gen2 arms.
  - **Box-moving (BO):**
    - The goal is to move colored boxes to the green region with PR2 robots.



PA: Packing colored objects into boxes.



BO: Moving the colored boxes to the green region.

Experiments: Success rate and planning time

- Our approach and Ap1 achieve higher success rates than Ap2 on all problem instances:
- Our approach achieves shorter planning times than Ap1 on all problem instances:

Problem Instance	Success rate %			Planning time (s)		
	Ap1	Ap2	Ours	Ap1	Ap2	Ours
PA5	**100.0**	80.0	**100.0**	5.6 (±1. 3)	6.1 (±2.1)	**2.4 (±0.2)**
PA7	80.0	70.0	**100.0**	39.8 (±12.8)	10.5 (±2.9)	**4.0 (±0.9)**
PA10	55.0	40.0	**90.0**	129.2 (±58.2)	N/A	**19.6 (±6.1)**
BO8	85.0	N/A	**100.0**	246.5 (±54.2)	N/A	**182.2 (±48.3)**

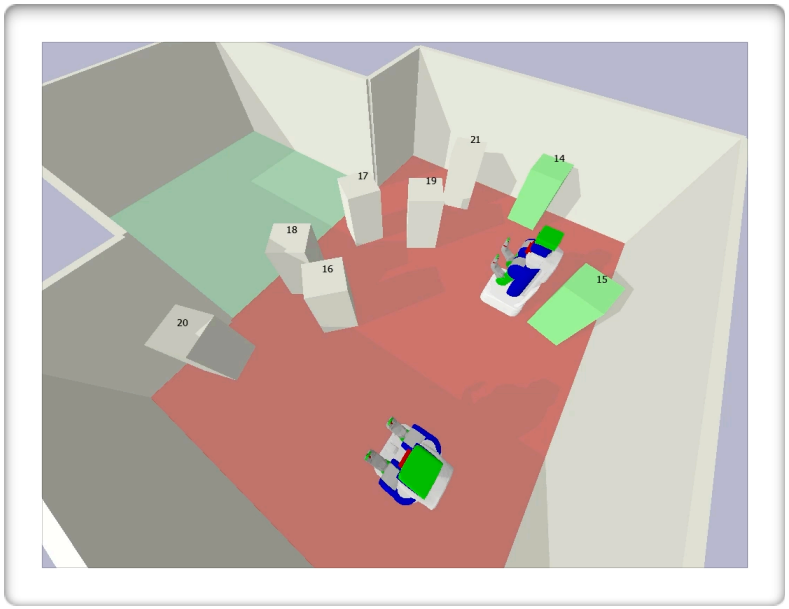
Experiments: Solution quality

- Our approach generates plans with lower motion costs and shorter makespans than Ap1 on all problem instances.

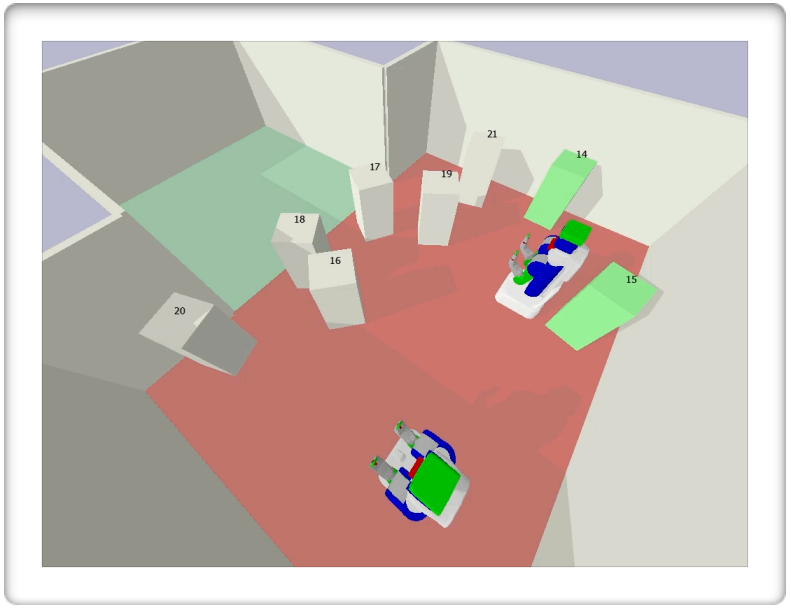
Problem Instance	Makespan			Motion cost		
	Ap1	Ap2	Ours	Ap1	Ap2	Ours
PA5	3.0 (±0.2)	2.9 (±0.2)	**2.8 (±0.2)**	3.8 (±0.2)	**3.6 (±0.2)**	**3.6(±0.2)**
PA7	3.7 (±0.3)	**3.0 (±0.3)**	3.1 (±0.2)	4.8 (±0.3)	4.3 (±0.2)	**4.1 (±0.2)**
PA10	4.6 (±0.6)	N/A	**4.2 (±0.3)**	5.6 (±0.6)	N/A	**5.2 (±0.4)**
BO8	4.8 (±0.2)	N/A	**3.4 (±0.3)**	7.6 (±0.1)	N/A	**5.0 (±0.6)**

Experiments: Simulation demonstrations

Ap1

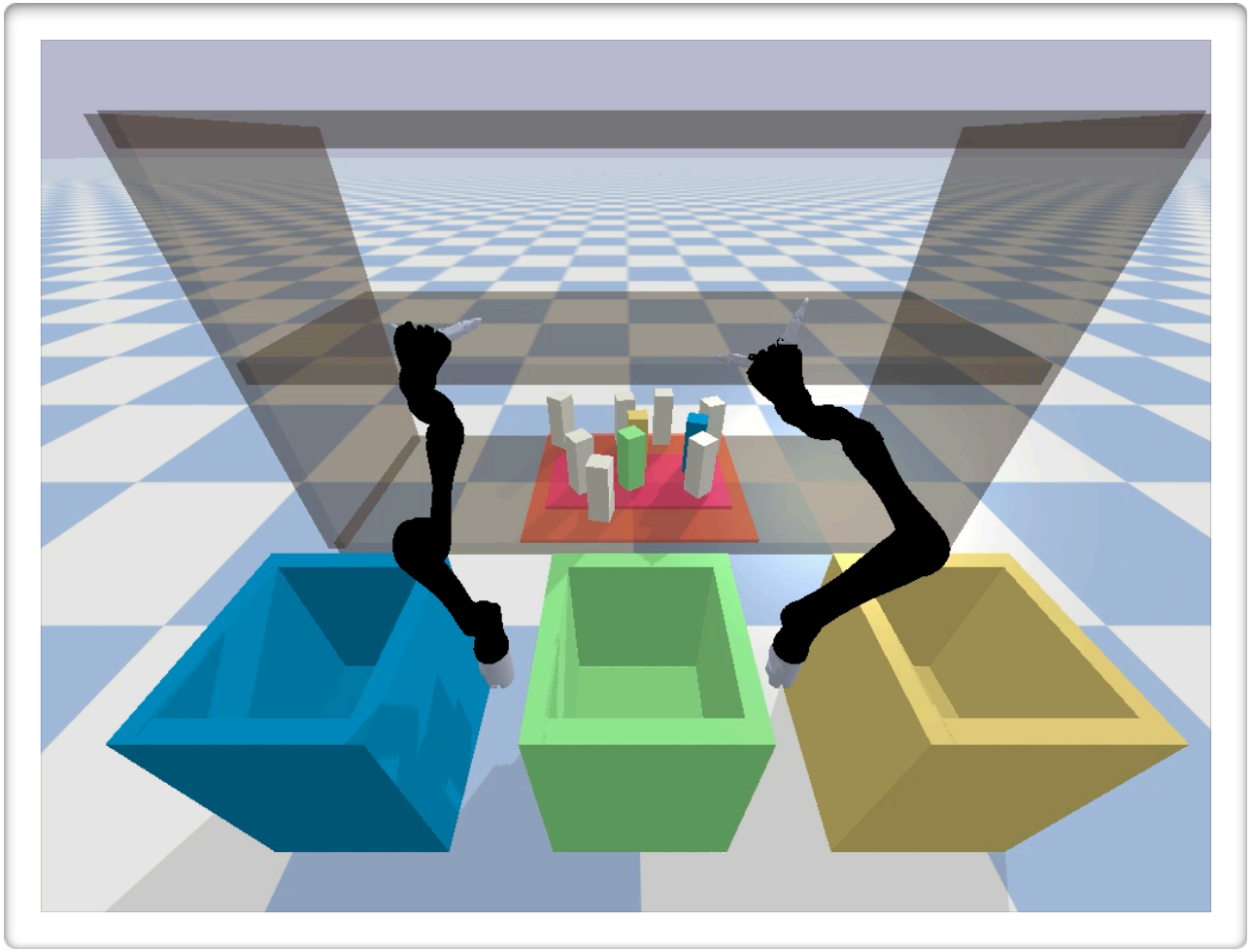


Ours



Our framework generates efficient task-and-motion plans.

Experiments: Simulation demonstrations



Limitations

1. We have assumed full observability of the scene.
2. We have assumed monotone instances of the MR-GTAMP problem.
3. We have assumed synchronous executions of actions by the robots.

Future work

1. Use learning to improve the planning efficiency.
2. Extend the developed techniques to more general MR-TAMP problems.
3. Extend the developed techniques to more diverse environments.

Contribution

To generate effective, collaborative task-and-motion plans for multi-robot systems to perform GTAMP tasks:

1. **Task-skeleton generating**:
   - *Which robot should move which objects?*
2. **Task-skeleton grounding**:
   - *Which locations should objects be moved to?*
3. **Combine generating and grounding with tree search**:
   - *How to search efficiently?*

2022 IEEE International Conference on Automation Science and Engineering

A MIP-based Approach for Multi-Robot Geometric Task-and-Motion Planning

Hejia Zhang, Shao-Hung Chan, Jie Zhong,
Jiaoyang Li, Sven Koenig, Stefanos Nikolaidis

ICAROS lab
University of Southern California