TAPAS: A Dataset for Task Assignment and Planning for Multi Agent Systems

Miguel Zamora*, Valentin N. Hartmann*, Stelian Coros
Computational Robotics Lab
ETH Zurich, Zurich, Switzerland

https://tapas-dataset.github.io

***Abstract*—Obtaining real-world data for robotics tasks is harder than for other modalities such as vision and text. The data that is currently available for robot learning is mostly set in static scenes, and deals with a single robot only. Dealing with multiple robots comes with additional difficulties compared to single robot settings: the motion planning for multiple agents needs to take into account the movement of the other robots, and task planning needs to consider which robot a task is assigned to in addition to when a task should be done.**

In this work, we present TAPAS, a simulated dataset containing task and motion plans for multiple robots acting asynchronously in the same workspace and modifying the same environment. We consider prehensile manipulation in this dataset, and focus on various pick-and-place tasks. We demonstrate that training using this data for predicting makespan of a task sequence enables speeding up finding low makespan sequences by ranking sequences before computing the full motion plan.

Code and datasets are open source and available on the project website, where videos of plans can be found as well.

I. Introduction

Adoption of robotics is crucial in the coming decades to tackle labor shortages and to do dangerous and repetitive jobs that humans currently do. However, to be economically viable, robots need to be able to do tasks at speeds comparable to humans, adaptability needs to be better, and they must be reliable.

To achieve the speeds at which humans do tasks, multi-agent systems are required, as single robot arms can currently not match the throughput of human workers. Multi-agent systems come with additional challenges that do not have to be solved in single robot settings, such as increased motion planning difficulties due to other robots moving in the same workspace, or having to deal with inter-robot dependencies of a plan. Further, one needs to decide which tasks are assigned to which robot, and in which order tasks need to be done. This assignment influences both the feasibility of a plan, and the speed at which it can be executed.

Learning-based approaches enable flexibility and resilience for robots, but most learning-based approaches at the moment are only applicable to single-agent systems. While some approaches might be able to readily handle multiple robots, most datasets [12, 21, 5] that are currently available focus on single agent settings with static environments. This is fine for some tasks, e.g., last-inch manipulation [15], where one can assume that we do not necessarily need to take interactions between robots into account. However, for other problems, such as motion planning, tracking a reference path, or assigning and sequencing tasks, the interaction between robots is crucial and can not be neglected. This highlights the need for access to data containing long-horizon plans for multiple agents, ideally performing a wide variety of tasks.

Following recent work in leveraging simulated data for learning, we build upon previous work in task and motion planning, and present **TAPAS**: a dataset for **T**ask **A**ssignment and **P**lanning for multi **A**gent **S**ystems. TAPAS contains both task plans and motion plans for a multi-robot setup with various robots, setups, scenes, and objects. We focus on prehensile manipulation, i.e., pick and place tasks as they are present in, e.g., logistics, machine tending, or kitting tasks.

TAPAS contains 4 structured base scenes of different difficulties with up to 4 robots. From these base scenes, we generate 7’000 randomized instances of these scenes with different robot arrangements and start and goal poses for the objects, and solve the task and motion planning problem using different robot-task assignments, and orders, resulting in 204’000 task plans and the corresponding motion plans.

We show a demonstration that makes use of this dataset by learning to predict makespan for a given candidate sequence, using this prediction to order the evaluation of sequences, and compare the learned policy with a random search policy.

II. Related work

Task and motion planning (TAMP) relies on solving both the task planning problem and the motion planning problem jointly to figure out what to do and how to do it. A comprehensive overview and taxonomy of various TAMP approaches is given in [6]. We focus here on multiagent planning, and learning for TAMP.

A. Learning in Task and Motion Planning

There are a variety of use cases for learning in TAMP: a common use-case is speeding up traditional TAMP pipelines through learning by, e.g., learning likely priors for the subproblems [13], through prediction of plan feasibility [20, 29], or by employing learned solvers for subproblems such as motion planning [9], or for predicting grasp locations by more

* Equal contribution.

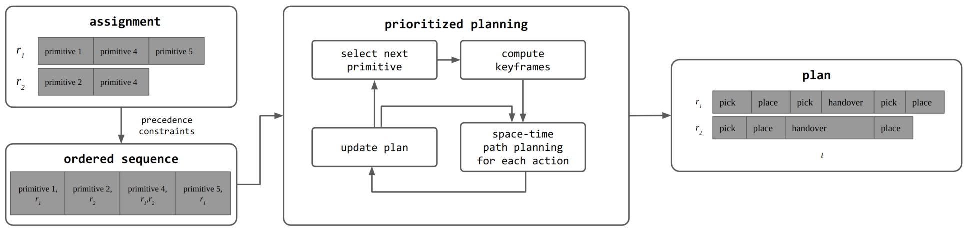


Fig. 1: Multi agent TAMP overview: First, the multi-agent assignment is serialized into a sequence. This sequence is then used for prioritized planning, which then results in a plan for all robots.

efficiently sampling constraint manifolds [14]. Learning was also previously used to enable skills that are then used in task and motion planners [28, 8].

Other, more recent approaches focus on learning the whole task and motion planer, e.g., using compositional models for TAMP, and chaining those together to achieve long-horizon planning capabilities [1, 18]. This can be done with data from play [26] without structure, or using expert demonstrations [16].

Finally, large language models have been used for TAMP as well, mostly for the high-level planning in multi-agent systems [17] leveraging the strong priors from language.

Contrary to what we present here, most work focused on single-agent settings. The dataset we present is a first step in the direction of learning for multi-agent task and motion planning, and we demonstrate a possible use-case in learning a policy to speed up the search for a good task assignment and sequence for a multi-arm scenario.

B. *Multi agent task and motion planning*

Most research in TAMP focuses on the single-agent use case. A simple way to extend the existing research is to treat all robots as a single one, and to plan in their combined joint space. This approach implicitly assumes synchronicity of all agents, and does not scale to many robots, as both task planning and motion planning in this combined space get exponentially harder. The assumption of synchronicity is common [22, 25, 17], but limiting, as it introduces the requirement of all robots starting and finishing at the same time, thereby slowing all robots down if the duration of the tasks differs even slightly from each other. In case many robots work in the same workspace, as one robot doing a task might block many other robots from fulfilling their tasks, thereby making them wait until the next possible timestep to fulfill their task.

Some other research relies on prioritized planning [11] or constraint solvers to assign tasks [2]. While these approaches are scalable, they are suboptimal, as they rely on prioritization, or compute suboptimal plans due to the nonconvexity of the problem setting. Additionally, these methods are currently too slow to deploy in the real world, and require perfect state estimation.

III. Multi agent TAMP

We consider prehensile manipulation of the objects in the scene, and allow the actions `pick`, `place`, and `handover`. We allow repeated handling of objects, leading to, e.g., one robot picking up an object and moving it into reach of another robot.

Our approach builds on the the work presented in [10], with various developments to speed up the planner. We briefly recap the most important parts here, and an overview is shown in Fig. 1:

1) The planner first generates a candidate assignment by assigning primitives to robots. A primitive is a combination of actions possibly involving multiple robots (e.g. the sequence `pick, handover, place`) that brings an object from its current location to the goal.
2) To then compute a plan, we serialize the plan into a fully ordered sequence that we obtain by introducing precedence constraints on the end times of the primitives.
3) Given this sequence, we do prioritized planning with the priority given by the order in the sequence, where we assume that the previously planned trajectories are fixed, and need to be respected by the following plans.

We describe the steps in more detail in the following sections.

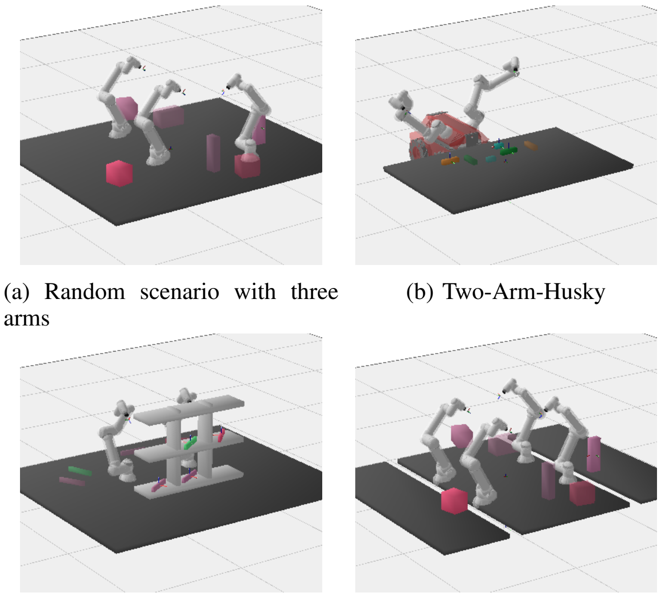
A. *Search over assignments*

To enable searching over (all valid) possible sequences and assignments, we serialize an assignment into a vector, and treat the ordering as a precedence constraint (Fig. 1, left). This allows us to enumerate all possible plans, leading us to find the best one given the suboptimal prioritized planning.

In practice, we randomly generate a possible sequence by sampling an object at random, checking which primitives are valid for which robots, and sampling uniformly from the set of valid primitive/robot combinations.

B. *Prioritized planning*

For a given primitive, we can compute feasible poses for each of the actions, i.e., poses how to grasp, handover, and place an object for the robots that are involved. We then use ST-RRT* [7] to plan motions for each of the actions that are part of a primitive, i.e., we plan from the previous pose of a robot to the location that corresponds to the


(a) Random scenario with three arms (b) Two-Arm-Husky

(c) Shelf (d) Conveyor

Fig. 2: The scenes we consider.

pick/handover/place pose. When planning the motions using ST-RRT*, we respect the precedence constraints that are encoded in the sequence. In addition, we plan an exit path, i.e., a path that ends at the home-pose of a robot, which is removed once we plan the next primitive[1].

After path planning, we post-process the path by shortcutting it, and smoothing it to obtain smooth velocity profiles.

IV. Data Generation

We consider multi-object rearrangement tasks where each object can be moved by a single robot (i.e. it is not too heavy), but might be out of reach of some robots, or require collaboration for reorientation.

A. Scenarios

We consider 4 different base scenarios (random, husky, shelf, conveyor), in which we randomize the number of robots, the number of objects, their size, and their start and goal locations. In the *random* scenario, we additionally randomize the robots base positions and orientations. We use the UR5 robot with a vacuum gripper in all the scenarios except for the shelf, where a two-finger gripper is used.

The scenarios are shown in Fig. 2, and Table I and Figs. 3 and 4 give an overview of the contents of the dataset and how the plans are distributed with respect to the environments, and number of objects. A comprehensive overview of the contents and features of the dataset can be found in Appendices A and B.

[1]This exit path makes the planner complete, as the existence of the path to the pose that does not block any other robot ensures that there is always a valid path for subsequently planned primitives.

TABLE I: Contents of the dataset.

Scenario	#Arms	#Objects	≈ #Scenarios	≈ #Plans
Husky	2	1-4	2k	40k
Conveyor	4	1-4	2k	58k
Random	1-4	1-4	2k	100k
Shelf	2	1-4	1k	6k
Total			7k	204k

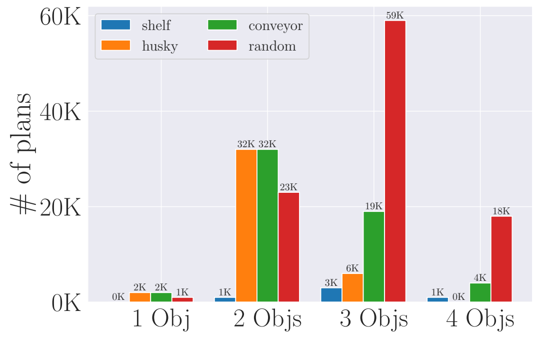


Fig. 3: Number of plans for each of the scenarios and number of objects.

B. Data collection

1) Scenario generation: We sample positions and sizes for each of the scenarios differently, corresponding to different tasks, and different difficulties in each of the settings:

- Random: The base positions of the robots are sampled using Poisson Disk Sampling with a disk radius smaller than that arm reach to ensure that handovers among robot arms are feasible. The start and goal positions of the objects are randomly sampled.
- Husky: Start positions and goal positions are randomly sampled.
- Shelf: Random poses of the objects, starts on the shelf, goals on the table.
- Conveyor: Start poses in the middle, goal poses outside.

For all sampled objects, the starting and goal poses are guaranteed to be within the reachable workspace of at least one arm.

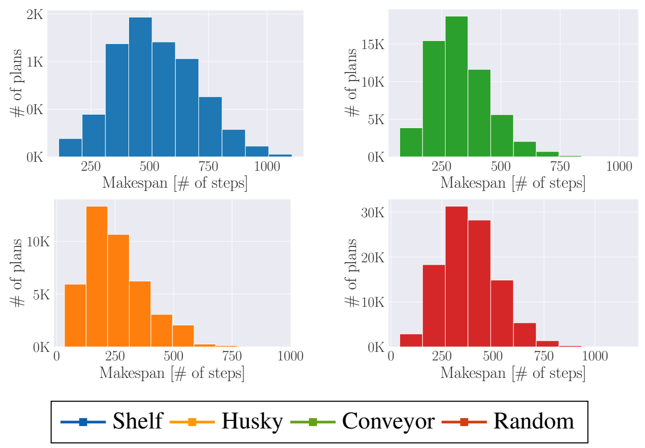


Fig. 4: Makespan of the plans for the different scenarios.

In total, we randomly generate approximately 7'000 different instances of the base scenarios. For each of them, we produce multiple task and motion plans that differ by task assignment, and task order, amounting to roughly 204'000 trajectories[2].

2) TAMP solving: We use the approach introduced in Section III and run it for a maximum of 120 seconds[3] to generate possible candidate assignments and sequences, and solve them to compute a valid plan. Since most assignments and sequences are suboptimal, most trajectories contained in the dataset are suboptimal. However, these can be used to, e.g., predict makespan for a given sequence, or to learn motion planning policies.

Further, we want to emphasize that the data generation process is not deterministic, as we use ST-RRT* for path planning and do not run it until convergence, as this would be too slow. As such, the makespan of a sequence should be regarded as noisy labels. In practice, this is not a problem, as more randomness will be injected by the non-deterministic execution of the plan, e.g., when trying to pick up an object inaccurately, and having to retry.

V. Experiments

We demonstrate a possible use-case of the dataset by speeding up sequencing and task assignment for multi-agent TAMP through learning.

A. Learning to predict makespan of task sequences

In this setting, we focus on the scenario with random robot and random object placements, and only use the data from this scenario. While we could learn from all data and leverage the possibility of transfer learning, in this initial work, we wanted to avoid the difficulties that arise for representation of the scene [23] that would be required.

In order to speed up the search for a good candidate sequence, we want to decrease the time that we spend on evaluating suboptimal sequences. We focus on learning to predict the makespan for a candidate sequence, and using this prediction to inform the search, i.e., rank the candidate sequences according to the prediction. This approach of predicting a score (the makespan) and ranking them is a *pointwise ranking* [4].

Compared to predicting a sequence directly, predicting the makespan for a candidate sequence has the advantage of being able to leverage optimizers to determine the feasibility of a primitive, to, e.g., filter out task assignments where a robot can not reach an object. This computation is quick compared to the complete planning of a sequence.

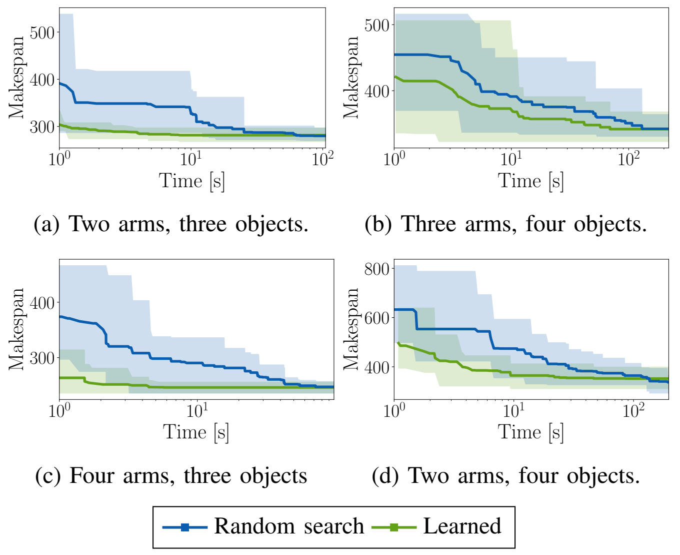
1) Policy training: Our policy is conditioned on the start state of the scene and the candidate sequence. To handle a variable number of robots and objects in a scene, we tokenize the task assignment into tokens that are defined as the concatenation of an action id, a robot id, the pose of the robot's base, the object id, and the object's initial and final poses. Token embeddings are obtained by learning a linear layer that maps the original tokens to an embedding dimension of 128, similar to the one used in [3]. Such embeddings are fed into a standard transformer encoder [27], which is then used as a backbone for a multi-layer perceptron head that outputs a predicted makespan.

Additional details about the policy training can be found in Appendix C.

2) Inference: We generate 100 possible sequences through random sampling, and rank them using the learned policy. We then evaluate them in the order of the ranking, until the planner times out (in case there is a computation time limit).

[2]Since the keyframe-generation for the two-finger gripper takes longer than for the vacuum gripper, the shelf-scenario has fewer plans computed than the other scenarios in the same time.

[3]Early stopping is possible if all sequences were tested.



(a) Two arms, three objects. (b) Three arms, four objects.

(c) Four arms, three objects (d) Two arms, four objects.

Fig. 5: Evolution of the best obtained makespan with compute time. We show the mean makespan and its min-max range over 10 runs.

VI. Results

We compare the approach introduced in the previous section to a basic random search. We evaluate these policies on the scenario with randomly sampled robot locations on 2, 3, and on 4 robots, with different number of objects. The instances of the scenes we evaluate here are not part of the training dataset.

The best found makespan over time is shown in Fig. 5. It is clearly visible that the learned policy outperforms the random search. While the learned policy is not always perfectly predicting the best makespan, the ranking of the sequences is good enough to obtain a much better anytime behaviour than the baseline approach.

VII. Discussion

The results on using this data to learn a policy that predicts the makespan for a given sequence in order to speed up the search for a low makespan multi agent task and motion plan show a possible use-case for TAPAS. While we learned a makespan predictor here, there are many other possible uses for the data presented in this work, such as learning to predict a sequence directly, or to learn a policy for multi agent motion planning.

We briefly discuss the question of what should we learn from simulated data, and what role will it play in training robot policies in the future.

A. Learning from simulation data

We believe that simulated data will play an important role in learning systems in the future. While it is feasible, and recommendable to learn, e.g., manipulation from expert data, such expert data is not available from human operators for many multi-agent systems. This is partially due to unintuitive plans in many cases, and in a much higher dimensional space than the one we as humans are in. While reinforcement learning approaches can overcome this problem, even for reinforcement learning approaches, it is useful to initialize a policy with, e.g., a behaviour cloning policy from suboptimal data.

When computing a solution to a path planning problem, very often many suboptimal plans are evaluated before converging to the optimal plan. We believe that a learner will be more sample efficient if learning from all (even suboptimal) data is facilitated. One possibility is the approach we presented here in this work, namely learning to predict cost of all solutions.

VIII. Conclusion

We presented a dataset containing 4 different scene categories with different levels of difficulty for task and motion planning encompassing 7K scenarios and more than 204k trajectories.

Compared to most multi-agent settings, we present plans that do not assume synchronization of the robot's actions, thereby enabling higher throughput. We demonstrated a simple use case of the data to learn to predict makespan, and showed how this policy can improve the search over possible candidate sequences.

In the future, we want to extend this dataset with more (and more realistic, see, e.g., [19]) scenes, and more primitives. Further, adding real-world data which is non-deterministic due to uncertainties in state estimation and execution uncertainty of policies could be interesting. On the learning side, promising future work is using this dataset to investigate sequence prediction further (with, e.g., mean variance estimation networks [24]), and explore learning of motion planning.

References

[1] Christopher Agia, Toki Migimatsu, Jiajun Wu, and Jeannette Bohg. Stap: Sequencing task-agnostic policies. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 7951–7958. IEEE, 2023.

[2] Jingkai Chen, Jiaoyang Li, Yijiang Huang, Caelan Garrett, Dawei Sun, Chuchu Fan, Andreas Hofmann, Caitlin Mueller, Sven Koenig, and Brian C Williams. Cooperative task and motion planning for multi-arm assembly systems. Preprint at http://web.mit.edu/yijiangh/www/papers/chen2022cooperative.pdf, 2022.

[3] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/7f489f642a0ddb10272b5c31057f0663-Paper.pdf.

[4] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems*, 22, 2009.

[5] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[6] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(Volume 4, 2021):265–293, 2021. ISSN 2573-5144. doi: https://doi.org/10.1146/annurev-control-091420-084139. URL https://www.annualreviews.org/content/journals/10.1146/annurev-control-091420-084139.

[7] Francesco Grothe, Valentin N. Hartmann, Andreas Orthey, and Marc Toussaint. ST-RRT*: Asymptotically-optimal bidirectional motion planning through space-time. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2022.

[8] Sagar Gubbi, Shishir Kolathaya, and Bharadwaj Amrutur. Imitation learning for high precision peg-in-hole tasks. In *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, pages 368–372, 2020. doi: 10.1109/ICCAR49639.2020.9108072.

[9] Huy Ha, Jingxi Xu, and Shuran Song. Learning a decentralized multi-arm motion planner. In *Conference on Robot Learning (CoRL)*, 2020.

[10] Valentin N. Hartmann and Marc Toussaint. Towards computing low-makespan solutions for multi-arm multi-task planning problems. In *ICAPS Workshop on Planning and Robotics*, 2023.

[11] Valentin N. Hartmann, Andreas Orthey, Danny Driess, Ozgur S. Oguz, and Marc Toussaint. Long-horizon multi-robot rearrangement planning for construction assembly. *IEEE Transactions on Robotics*, 2022. ISSN 1552-3098. doi: 10.1109/TRO.2022.3198020.

[12] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, and Kirsty Ellis et al. Droid: A

large-scale in-the-wild robot manipulation dataset. 2024.

[13] Beomjoon Kim and Luke Shimanuki. Learning value functions with relational state representations for guiding task-and-motion planning. In *Conference on Robot Learning (CoRL)*, pages 955–968. PMLR, 2020.

[14] Teguh Santoso Lembono, Emmanuel Pignat, Julius Jankowski, and Sylvain Calinon. Learning constrained distributions of robot configurations with generative adversarial network. *IEEE Robotics and Automation Letters*, 6(2):4233–4240, 2021.

[15] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning, 2024.

[16] Ruidong Ma, Jingyu Chen, and John Oyekan. A learning from demonstration framework for adaptive task and motion planning in varying package-to-order scenarios. *Robotics and Computer-Integrated Manufacturing*, 82:102539, 2023. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2023.102539. URL https://www.sciencedirect.com/science/article/pii/S0736584523000157.

[17] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models, 2023.

[18] Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning (CoRL)*, 2023. URL https://openreview.net/forum?id=HtJE9ly5dT.

[19] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

[20] Joaquim Ortiz-Haro, Valentin N. Hartmann, Ozgur S. Oguz, and Marc Toussaint. Learning efficient constraint graph sampling for robotic sequential manipulation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021.

[21] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[22] Rahul Shome and Kostas E Bekris. Synchronized Multi-Arm Rearrangement Guided by Mode Graphs with Capacity Constraints. *arXiv preprint arXiv:2005.09127*, 2020.

[23] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019.

[24] Nicki Skafte, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] Ilyes Toumi, Andreas Orthey, Alexander von Rohr, and Ngo Anh Vien. Multi-arm bin-picking in real-time: A combined task and motion planning approach. *arXiv preprint arXiv:2211.11089*, 2022.

[26] Núria Armengol Urpí, Marco Bagatella, Otmar Hilliges, Georg Martius, and Stelian Coros. Efficient learning of high level plans from play. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 10189–10196. IEEE, 2023.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[28] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *International Journal of Robotics Research*, 40(6-7):866–894, 2021. doi: 10.1177/02783649211004615.

[29] Zhutian Yang, Caelan R Garrett, Tomás Lozano-Pérez, Leslie Kaelbling, and Dieter Fox. Sequence-Based Plan Feasibility Prediction for Efficient Task and Motion Planning. In *Proc. of Robotics: Science and Systems (R:SS)*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.061.

Appendix

A. Scenes

We show the initial conditions of some of the randomized scenes in Fig. 6. This illustrates the range of the object placements, robot placements, and objects size for the *random* scene.

B. Data Features

We export the following data for each of the scenarios:

- the *scene description* containing the required information to reconstruct the scene, i.e., robot base pose, the obstacles, their sizes, and the start and end poses of the objects,
- the *sequence* that was used to generate the trajectory, containing the primitives, the robots and the object that is assigned to a primitive,
- the resulting *symbolic plan* with the start and end times for each action,
- and the *trajectory*, consisting of some metadata, and an array of steps, with each step containing the joint pose, the symbolic states, and the end-effector-position.

Additionally, we export a metadata file that contains the number of robots, the number of objects, the cumulative computation time, and the makespan.

C. Policy details

The code for the training and inference is available on the website. Fig. 7 shows the loss curve for different scenarios. We report the hyperparameters we used for training in Table II.

TABLE II: Training Hyperparameters

Hyperparameter	**Value**
Learning rate	1e-3
Batch size	128
Optimizer	RAdam
Learning Rate Scheduler	Linear
Epochs	10

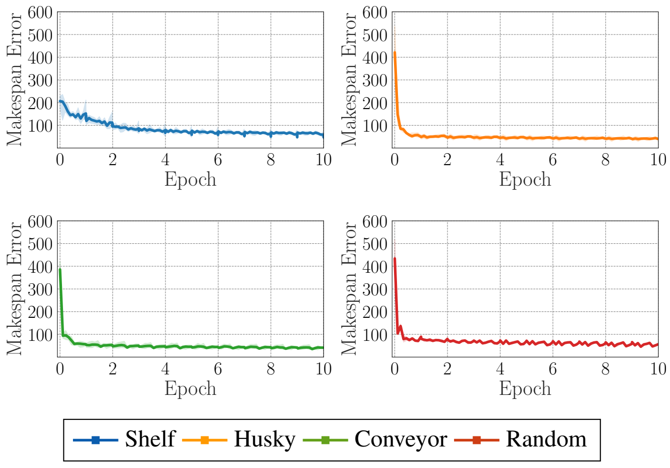


Fig. 7: Training loss for all scenarios.

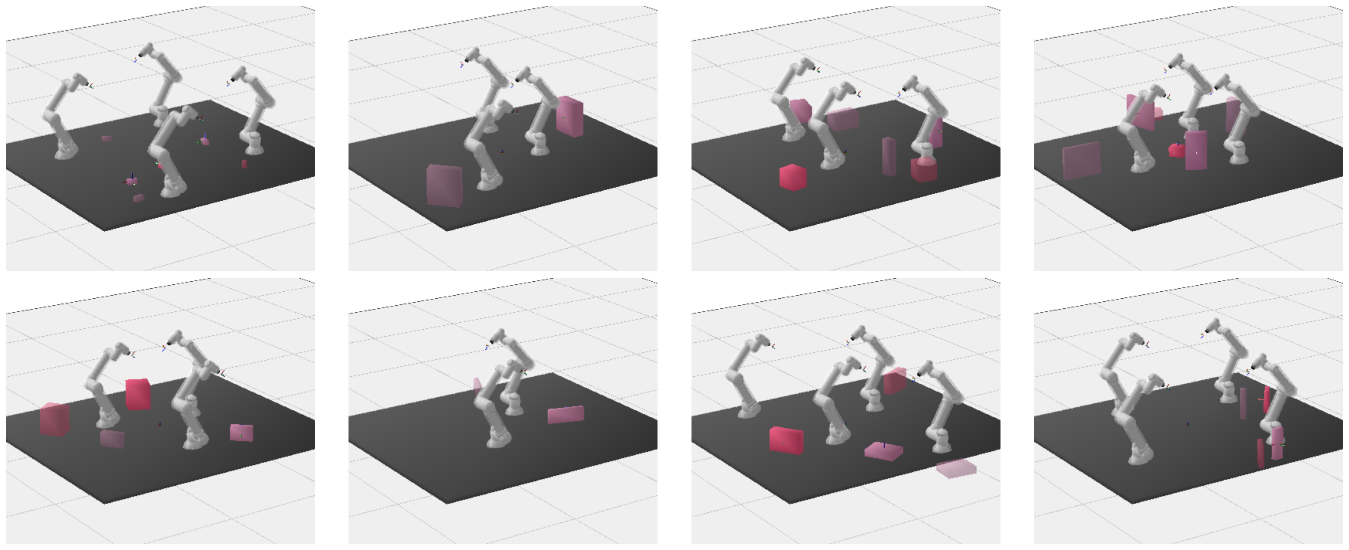


Fig. 6: Snapshots of the initial conditions for some of the randomized scenes we consider. The opaque object is the start pose and the transparent object is its goal pose.