Synergistic Scheduling of Learning and Allocation of Tasks in Human-Robot Teams

Shivam Vats[1] Oliver Kroemer[1] Maxim Likhachev[1]

***Abstract*— We consider the problem of completing a set of n tasks with a human-robot team using minimum effort. In many domains, teaching a robot to be fully autonomous can be counterproductive if there are finitely many tasks to be done. Rather, the optimal strategy is to weigh the *cost* of teaching a robot and its *benefit*- how many new tasks it allows the robot to solve autonomously. We formulate this as a planning problem where the goal is to decide what tasks the robot should do autonomously (act), what tasks should be delegated to a human (delegate) and what tasks the robot should be taught (learn) so as to complete all the given tasks with minimum effort. This planning problem results in a search tree that grows exponentially with n – making standard graph search algorithms intractable. We address this by converting the problem into a mixed integer program that can be solved efficiently using off-the-shelf solvers with bounds on solution quality. To predict the benefit of learning, we propose a precondition prediction classifier. Given two tasks, this classifier predicts whether a skill trained on one will transfer to the other. Finally, we evaluate our approach on peg insertion and Lego stacking tasks, both in simulation and real-world, showing substantial savings in human effort.**

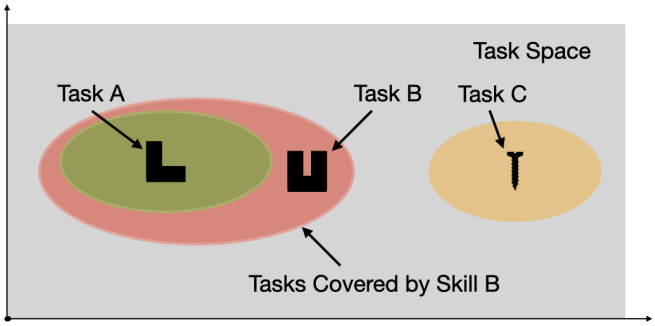

Fig. 1: Consider three assembly tasks visualized in a 2D task state-space. Each colored oval covers tasks that can be solved by a specific robot skill. Note that skill B covers more tasks than the other two skills while task C remains uncovered even after learning skill B. Our framework schedules teaching of only those skills that cover enough future tasks to offset the cost of robot teaching. Remaining tasks are delegated to a human for completion.

I. Introduction

For real world applications of robotics, like manufacturing and health-care, autonomy is not an end in itself, but a means to improve productivity and safety. It is often infeasible or expensive to teach robots to be fully autonomous due to changing environments and task requirements. In practice, these autonomous systems often have the option of falling back on human help when needed. In this work, we consider two modes of help provided by a human– giving demonstrations on how to do a new task and fully taking over a task. The former allows the capabilities of a robot to be extended, which is useful when similar tasks are expected to be encountered again in the future. The latter allows the robot to avoid attempting or having to learn one-off tasks.

Consider a manufacturing facility that gets its orders at the start of each day and needs to fulfil those orders by its end. The factory operators may have only an approximate idea about future demand. Hence, when the orders arrive there will be tasks that the robot can not complete autonomously. This leads to a number of questions: Should additional robot teaching be done? If so, on which tasks? What tasks should be done by robots and what tasks by humans?

To this end, we propose a decision making framework *Act, Delegate, or Learn* (ADL) that jointly reasons about autonomous execution in synergy with *both* of these modes of human assistance. In particular, we look at a setting where tasks come in a fixed sequence. This is motivated by time and cost critical domains like agile assembly lines in factories and robots in outer space, where a diverse but known set of tasks need to be accomplished with minimum human and robot effort. While human help is available, it is at a premium. Hence, we would like to use it optimally so as to minimize the overall effort.

Each of these two modes has been investigated individually in prior works. A number of works [1], [2], [3], [4] in Learning from Demonstrations (LfD) [5], [6] use measures of confidence in the robot's actions and active learning [7] to teach a robot with fewer human demos. By not considering the option of delegating tasks to a human, these approaches seek to achieve full autonomy, which is not cost-effective in settings where delegation is possible. In contrast, planners for task allocation [8], [9], [10] and adjusting the level of autonomy [11], [12] have been proposed for human-robot teams. The focus on these works is on handling spatiotemporal constraints and different human models [13], [14], [15]. However, they assume a static model of the robot's capabilities, which does not allow them to leverage robot learning in their framework.

In summary, the main contributions of our work are:

(1) *Act, Delegate or Learn Framework:* We formulate the problem of completing a given sequence of tasks with human help at minimum total expected human and robot effort as a Stochastic Shortest Path problem. (2) *Efficient Planning:* We propose a mixed integer programming formulation to efficiently solve this problem. (3) *Precondition Prediction:*

[1]Robotics Institute, Carnegie Mellon University {svats, okroemer, mlikhach} @andrew.cmu.edu

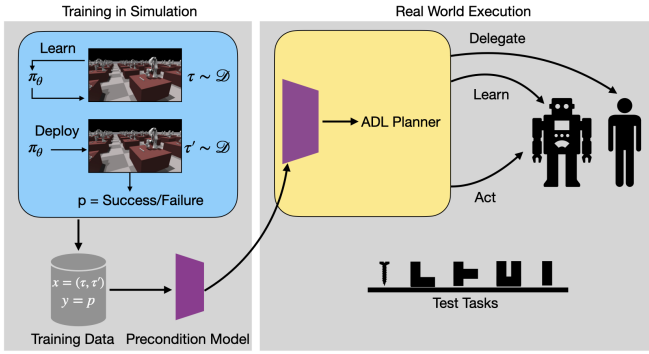

Fig. 2: Overall approach: (i) **Training in Simulation** Skills are learned (using RL) and deployed on tasks $\tau \sim \mathcal{D}$ to collect data on what other tasks can be solved by a skill learned for a particular task. A precondition prediction model is trained using this data. (ii) **Real World Execution** Our planner makes use of the learned model to decide when the robot should attempt a task, when it should delegate to a human and when it should learn a new skill for a task.

Planning requires the ability to foresee the benefit of robot teaching before committing to it. To this end, we propose a precondition prediction model that predicts what other tasks the robot will be able to solve after getting demonstrations for a task. We train this model offline using a domain-specific simulation. (4) *Simulated and Real World Evaluation* [1] *:* We evaluate the benefits of our approach on two challenging manipulation tasks: (a) Peg-in-a-hole: Insert pegs into holes under uncertainty using environmental contact for localization and (b) Lego Stacking: Robustly stack complex parts made with Lego bricks onto a Lego base plate.

II. Related Work

Function Allocation is the decision making problem of determining which functions should be performed by machines and which by humans [16], [17]. While a number of strategies have been proposed, the one closest to our work is *economic allocation* [16], [18] which finds an allocation that ensures economic efficiency.

Adaptive Automation can accommodate changes in the environment or the human for function allocation. A number of frameworks have been proposed over several decades [19], [20], [21], [22] which focus on optimizing operator workload, attention and efficiency. Consequently, their focus has been on modeling the human [13], [14], [15]. [12] recently propose an interactive model of autonomy, where a system learns a model of its competence online. All these strategies assume that the robot has certain fixed capabilities

Learning from Demonstrations: There are three main categories[23] of LfD– kinesthetic teaching, teleoperation and passive observation. Kinesthetic teaching is the most common approach for providing demos in manufacturing and health-care [23], while teleoperation does not require the user to be copresent with the robot. Passive observation usually requires multiple demos [4], special instrumentation (motion capture, force-torque sensors) depending on the task and is complicated to solve due to the need for retargeting. Despite recent progress, teaching robots generalizable skills still requires significant human effort.

Consequently, a number of works [1], [2], [3], [4] seek to minimize the number of demos required for teaching. In particular, Confidence-Based Autonomy [1] uses classification confidence to choose between autonomous execution and request for a demo. [24] propose an online approach to training a set of controllers from demonstrations that tries to myopically minimize human effort. ThriftyDAgger [25] uses estimated probability of task success to determine when to solicit human interventions.

Multi-task Learning: [26] look at learning a single policy in a multi-task setting with a continuous set of tasks. [27] learn a two level policy where the low level policy controls the robot for a given context and the high level policy generalizes among contexts. In contrast, we take a library of independent skills approach, where generalization happens only at the lower level.

III. Preliminaries

Skill Preconditions: We model skills using the options framework [28], [29], [30]. We use a probabilistic notion of skill preconditions [31], where the preconditions of a skill is a classifier $\rho : \Theta \rightarrow [0, 1]$ that takes in features describing a task and returns the probability that the skill will be able to successfully complete the task. This classifier is usually trained by executing the skill on a distribution of tasks to generate success/failure labels [32], [33]. However, this is an expensive process which requires real world execution of the robot.

Skill Library: A popular approach for solving related tasks is to learn a parameterized skill [34], that adapts the policy based on changes in the task. This approach is practical if only some aspects of the task can change. Adapting to various changes in the tasks requires a more complex skill parameterization that makes the learning problem harder and more sample complex. An alternative approach, which we take in this work, is to have the robot maintain a library $\mathcal{L} = \{\pi_1, \cdots, \pi_n\}$ of skills, each of which is learned on a narrow task distribution from demonstrations. Given a task τ, the robot picks an appropriate skill for it. by selecting a skill with the highest probability of success: $\arg\max_{\pi \in \mathcal{L}} \rho_\pi(\tau)$. This representation has a number of advantages over learning a monolithic skill, chiefly, modularity, allowing local updates and providing alternatives in case of execution failure.

IV. The Act, Delegate or Learn Framework

We are interested in completing a sequence of tasks with minimum total expected human and robot effort. At train time, we are provided a distribution $\mathcal{D}$ of tasks that are expected to be encountered. The robot may be pre-trained with a set of skills based on this knowledge. The actual tasks and the order in which they need to be done are revealed only at test time. In this stage, a decision needs to be made for every task: should the robot do the task, should it delegate the

[1] Videos and supplementary material are available at `https://sites.google.com/view/actdelegateorlearn`

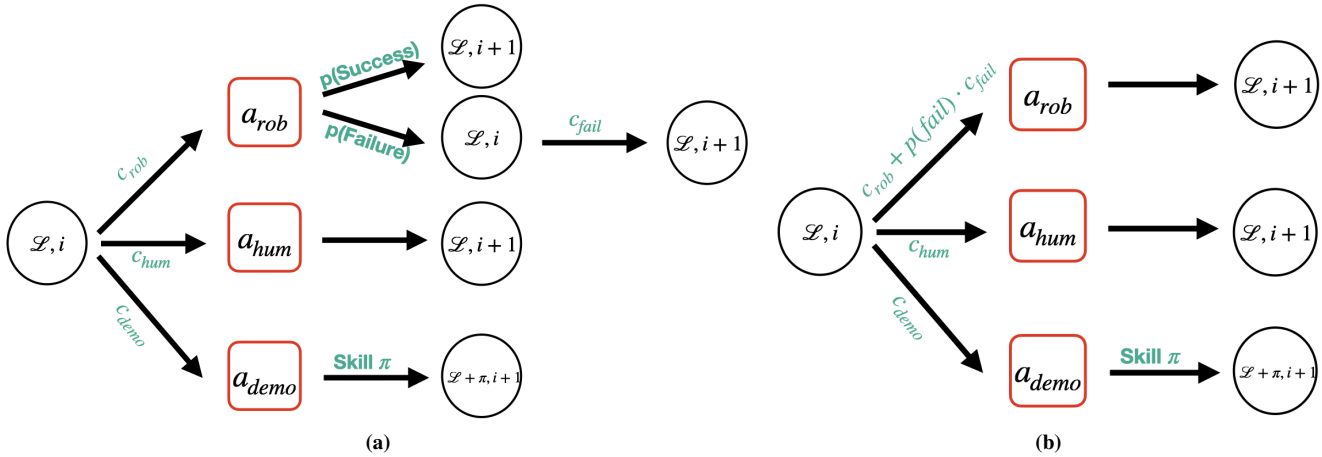

Fig. 3: (a) **Transition Model:** Our MDP has three actions: a_{rob}, a_{hum} and a_{demo} with associated costs of c_{rob}, c_{hum} and c_{demo} corresponding to the options act, delegate and learn. A human intervenes to complete a task if robot execution fails. We assume that a human can complete all the tasks and is available at all times to teach the robot. (b) **Simplified Transition Model:** We can replace the two stochastic outcomes due to a_{rob} with a single outcome whose cost is an expectation over them.

task to a human or should it ask to be taught how to do the task? We require that every task be completed. Hence, each robot failure incurs additional cost due to human intervention to complete the task and correct the setup. Finally, we assume that a human is available at all times to intervene if needed - either to correct a robot failure or to teach it, for example, by providing demonstrations.

A. Problem Formulation

We formulate this problem as a Stochastic Shortest Path (SSP) problem $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G})$ [35] where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a transition model, $\mathcal{C} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^+$ is a cost function and $\mathcal{G} \subset \mathcal{S}$ is a set of goal states. We define each of these components of the MDP for our problem:

State Space: Each state $s \in \mathcal{S}$ is a tuple $\langle \mathcal{L}, k \rangle$, where $\mathcal{L}$ is the skill library of the robot at that state and k refers to the tasks completed so far.

Action Space: $\mathcal{A} = \{a_{rob}, a_{hum}, a_{demo}\}$, where a_{rob} implies that the robot attempts to solve the task, a_{hum} implies that the human solves it and a_{demo} implies that the human teaches the robot a new skill for it in addition to solving it.

Transition Function $\mathcal{T}$ models whether the skill library got updated or a task was completed after an action. Though the outcome of robot execution is stochastic, we can convert it into a deterministic MDP by taking an expectation over the two outcomes (see figure 3 for details). We will be using the resulting simplified transition model in the rest of the paper. The transition model makes it clear that a_{rob} and a_{hum} do not affect the skill library in any way. On the other hand a_{demo} updates the library by adding a new skill π to its repertoire.

Cost Function The cost function is defined as

$$\mathcal{C}(s_i, a) = \begin{cases} c_{rob}(i) + \Pr(fail) \cdot c_{fail}(i) & a = a_{rob} \\ c_{hum}(i) & a = a_{hum} \\ c_{demo}(i) & a = a_{demo} \end{cases}$$

where, $\Pr(fail) = 1 - \max_{\pi \in \mathcal{L}} \rho_\pi(\tau_i)$. The cost of a robot execution includes the cost of a potential failure and hence depends on the robot's skill library. c_{rob}, c_{hum} and c_{demo} are domain and task dependent costs specified by a domain expert. For example, in manufacturing, where minimizing the *economic cost* of production is crucial, c_{rob} could reflect the cost of operating a robot, while c_{hum} and c_{demo} could depend on the efficiency of a human collaborator. There exist a number of approaches [14], [15], [36] to model human performance. c_{fail} corresponds to the difficulty of fixing a mistake made by the robot. In some domains, this could be as simple as asking a human in the factory to complete the remaining task, while in others, it may be high if there is a risk of damage due to a failure.

Goal: A goal state is reached once all the tasks have been completed.

Let $\{\tau_i\}_{i=1}^n$ be the sequence of tasks and $\eta = \{\eta_i\}_{i=1}^n$ be the sequence of actions taken. Then, the expected cost of execution is: $J(\eta) = \sum_{i=1}^n \mathcal{C}(s_i, \eta_i)$, where $s_{i+1} = \mathcal{T}(s_i, \eta_i)$ and our goal is to find an optimal plan $\eta^* = \arg\min_{\eta \in \mathcal{A}^n} J(\eta)$.

V. Planning

The standard techniques used to solve a deterministic SSP are graph search algorithms like Dijkstra's algorithm and A*. Unfortunately, the search graph induced by our problem has *exponentially* many states in the number of tasks to be done. Though A*-like algorithms can leverage heuristics to speed-up search, their performance is highly dependent on the quality of the heuristic and hence incur substantial overhead for designing good heuristics.

Motivated by this, we propose a mixed integer programming (MIP) formulation of the SSP which can be solved using off-the-shelf solvers without the need to design heuristics. These solvers provide high quality solutions (with sub-optimality bounds) and are highly scalable.

A. MIP Formulation

We introduce decision variables for every task: $x_i, y_i, z_i \in \{0,1\}, w_i \in [0,1], \forall i \in \{1, \cdots, n\}$. Let binary decision variable x_i be 1 if a demo is sought on task τ_i, y_i be 1 if a human is asked to solve it and z_i be 1 if the robot is asked to attempt the task. As the robot may fail in its attempt, we model the probability of human intervention with a continuous decision variable w_i– note that it is non-zero only if robot execution is chosen for a task. We exercise indirect control over w_i via the probability of failure of the action taken.

Our overall objective is:

$$\min \sum_{i=1}^{n} c_{demo}(i)x_i + c_{hum}(i)y_i + c_{rob}(i)z_i + c_{fail}(i)w_i$$

where, $z_i = 1 - x_i - y_i$ as we allow exactly one of these three actions for a task. Hence, the objective can be simplified.

$$\min \sum_{i=1}^{n} c'_{demo}(i)x_i + c'_{hum}(i)y_i + c_{fail}(i)w_i \tag{1}$$

where $\forall i \in \{1, \cdots, n\}$

$$c'_{demo}(i) = c_{demo}(i) - c_{rob}(i)$$
$$c'_{hum}(i) = c_{hum}(i) - c_{rob}(i)$$
$$w_i = 1 - \max\{\rho_0, \rho_1(\tau_i)x_1, \cdots, \rho_i(\tau_i)x_i, y_i\}$$

The max term in the last equation is a maximization over the success probabilities of the available ways to solve the task – using pre-trained skills (with precondition ρ_0), learning new skills (with preconditions $\rho_1 \cdots, \rho_n$) and delegating to a human (represented by y_i). y_i is 1 if the robot delegates the task to a human, in which case we are assured of task completion. In its current form this program is not linear due to the max operation. However, we can easily convert it into a linear MIP by introducing additional binary decision variables. Some solvers like Gurobi [37] can directly take this program and do the linearization under the hood.

Note: Alternatively, we can also formulate it as a facility location problem [38], where all tasks are customers and opening a facility corresponds to either seeking a demo or delegating to a human. While solving the facility location problem optimally is NP-hard, it has $O(\log n)$ approximation algorithms which can be useful if the MIP is too big to solve optimally.

VI. Precondition Prediction Model

A key requirement of our planner is the ability to foresee the benefit of robot teaching *before* committing to it. Past works [32], [33] have looked at the problem of precondition learning, wherein a classifier is trained for an existing skill to predict what other tasks can be solved by it. By contrast, we need to predict the preconditions of a skill that *will* be learned if we choose to teach the robot– a precondition prediction problem. Our proposed solution is to learn a classifier (see figure 4) that takes as input a train task and a test task and predicts whether a robot trained on the former will be able to solve the latter. Intuitively, this can be thought of as learning a similarity metric between tasks.

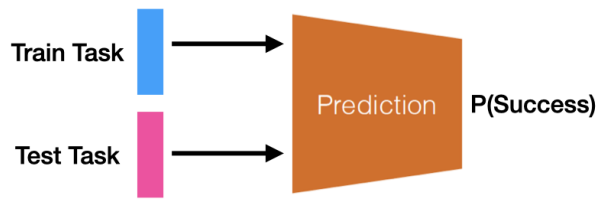

Fig. 4: Precondition prediction model predicts the probability of success on a test task τ' after the robot has been trained on a task τ.

We collect training data for the precondition model using algorithm 1. This can be prohibitively expensive as we need to learn robot policies to generate labels. We get around this limitation by observing that we do not need to transfer robot skills from sim2real but only task relationships— the former requires high fidelity simulation while the latter does not. It is often the case that a lower dimensional state representation is sufficient to discriminate between tasks. The key is to simplify the problem such that inter-task relationships remain intact– tasks that are similar/dissimilar in the real world should remain so in simulation and vice versa. Concretely, we define an *abstraction* [39], [29] M as a pair of functions (f, g) such that $f : S \to S'$ maps the original problem state space S to a smaller state space S' and $g : A \to A'$ maps the full action space to a smaller action space. The specific state and action abstraction to be used in training are provided as domain knowledge.

Algorithm 1 Data collection in abstract simulation.

```
procedure GetTrainingData(m, n)
    X ← ϕ, Y ← ϕ
    S ← Sample m tasks from D
    for i ∈ {1, ···, n} do
        Sample τ from D
        π ← Learn policy for τ
        for τ' ∈ S do
            x ← (τ, τ')
            y ← Evaluate π on τ'
            X.insert(x), Y.insert(y)
    return X, Y
```

VII. Experiments

We evaluate our approach, both in simulation and in the real world, on two challenging problems (1) block insertion under uncertainty and (2) Lego stacking. Our objectives are (1) to understand the benefits of the ADL framework as compared to baselines that are myopic or reason about only a subset of the three options (2) to evaluate our hypothesis that the precondition model can be trained in simulation. In both these experiments, we use a 2-layer fully connected neural network as our precondition prediction model and we are able to solve our mixed integer program optimally in well

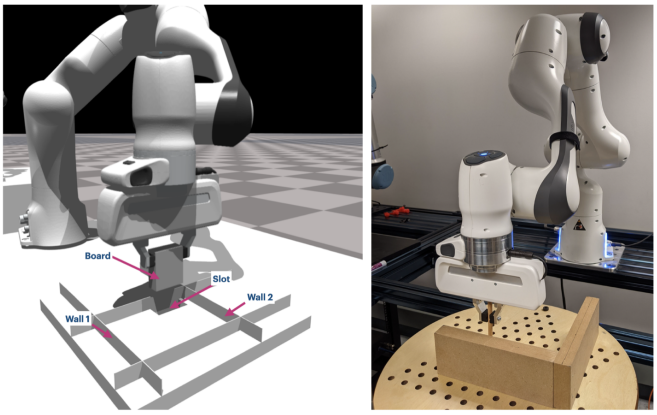

Fig. 5: Block (peg) insertion under uncertainty in simulation and in real world.

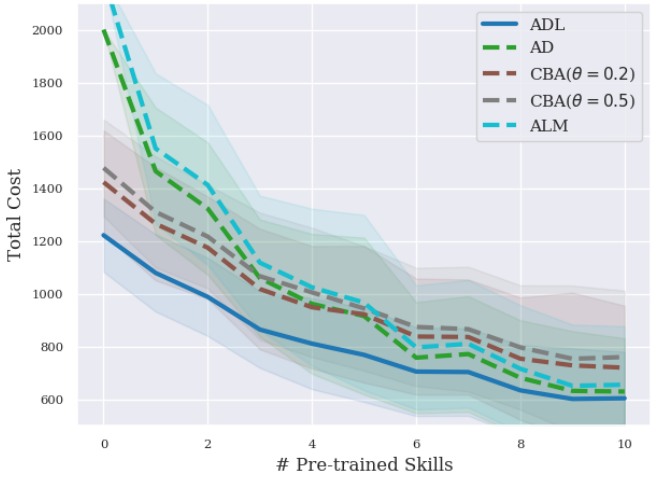

Fig. 6: Comparison of ADL vs baselines in total cost for solving 20 block insertion tasks at different levels of skill pretraining. Pretraining is done by teaching the robot randomly sampled tasks from the task distribution. ADL is strictly better than all baselines at every level of pre-training. However, after pre-training with 8 skills, both ADL and AD converge to full autonomy as the robot is able to solve most of the tasks with pre-trained skills. We use $c_{rob} = 10$, $c_{hum} = c_{fail} = 100$ and $c_{demo} = 200$.

under a second using Gurobi [37]. We provide additional details in the appendix.

Baselines: We compare our approach against three baselines: (1) *Act Delegate (AD):* The robot chooses between acting and delegating based on the expected costs of these two actions. (2) *Confidence-Based Autonomy (CBA) [1]:* Given a fixed threshold θ, the robot attempts a task if its confidence in success is greater than θ and asks for demonstrations, otherwise. (3) *Act-Learn Myopic (ALM):* Similar to the strategy used by [24], the robot chooses between attempting a task and asking for human demos by comparing the immediate expected costs of both the actions.

Metrics: The main evaluation metric is the total cost of completing all the given tasks. We also compare the methods based on the number of demonstrations and human interventions and the number of failures.

Skill Representation: In both our experiments, the robot end-effector is controlled using Cartesian-space impedance control which commands torques at the end-effector based on errors in the Cartesian space using a spring-damper system. A skill is a sequence of waypoints in the robot's end-effector frame, where each waypoint is defined by a 6D pose and the stiffness to be used in the corresponding spring-damper system.

A. Block Insertion

Our first evaluation is in simulation to understand how well our planner performs in comparison with standard non-planning approaches.

Task: Each task involves inserting a block of dimensions 1 cm x 1 cm x 6 cm into a slot of dimensions 1.2 cm x 1.2 cm x 2 cm in a known environment with a noisy estimate of the slot location $\sim \mathcal{N}(0, 0.3^2 cm^2)$. We generate four different environments of dimensions 20 cm x 20 cm each, with different numbers of walls arranged in a grid. We use the Nvidia Isaac Gym simulator [40] to simulate the tasks and to train the precondition model.

Simulation Results: We compare ADL with AD, CBA($\theta = 0.5$) , CBA($\theta = 0.2$) and ALM in figure 6, where 0.2 is the optimal CBA threshold found using grid-search. ADL outperforms all the baselines at every level of pretraining. However, the improvement provided by ADL drops with increase in pretraining as the robot can complete more of the tasks autonomously without seeking additional demos or delegating. Also note that CBA outperforms AD at low levels of pretraining but the opposite holds at higher levels as demos sought by CBA are not cost-effective for the task set. We provide comparisons using different costs and qualitative results from a real-world experiment in the appendix.

B. Lego Stacking

In our second domain of Lego stacking we seek to evaluate how well our method works in the real world. In particular, we want to understand whether a precondition model learned using an abstract simulation is able to reduce effort in real world.

Task: Each task involves picking up a part made up of Lego bricks from a table and stacking it firmly onto a Lego base plate. A robot execution fails if two or more corners of the part are not locked onto the plate or the robot hits the base at any point. The robot is provided a bounding box around the part, a grasp location and a target location by the user. We use a 66D feature vector for each task– binarized and resized image (to 8×8) along with its original size. Before running the experiments, we record 5 demos for each task in the ground set. Every time the robot requests a demo for a task, one of the 5 pre-recorded demos is provided by sampling randomly.

Skill: Each stacking skill consists of three sub-skills executed in sequence: pickup, place-and-wiggle and robust-tapping. The first two are hand-designed and common across all tasks, while robust-tapping needs to adapt the number and

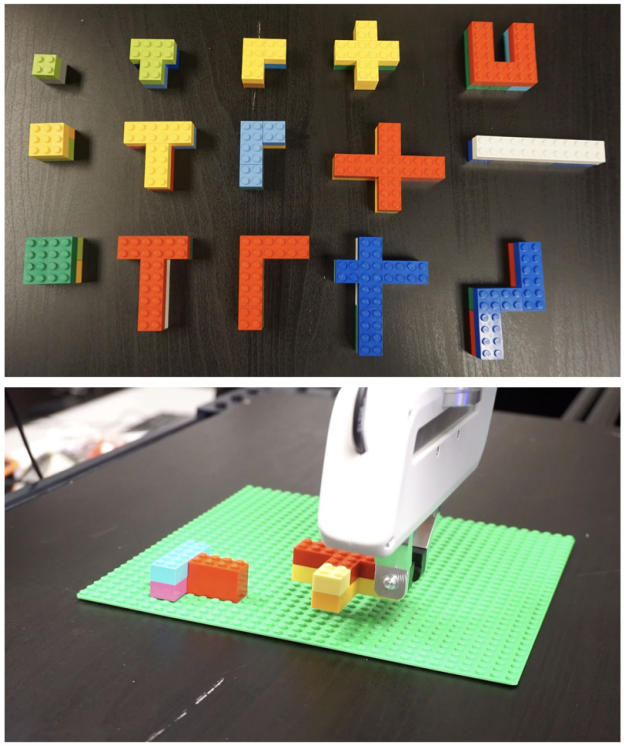

Fig. 7: (*Top*) The ground set of 15 tasks from which test sets of 10 tasks each are sampled uniformly randomly. (*Bottom*) The Franka-Emika Panda robot stacking one of the parts onto the base plate.

location of taps based on the geometry of the part. The latter is learned in the grasp-frame and scaled based on the size of the part. This allows the skill to generalize to different locations and across parts of similar shape but different sizes.

Data Collection: Physics-based simulators struggle to simulate interactions among multiple Lego bricks and the interference fit mechanism used in them. Consequently, we use a custom simulation based on our observation that the primary reason for variability in skills is the geometry of the parts. We can afford to ignore physics and robot dynamics as we do not transfer the learned skills to the real world. Our coverage-based simulation takes in a 2D image of a part and identifies only the number and location of taps needed to cover the whole part by randomly sampling points on the image. Experimentally, we found that a single tapping action has an effect upto about 3cm from the tapping location. We use this knowledge in the simulation to determine whether a part is covered or not after a sequence of taps. We capture 10 images of each of the 15 tasks, along with a bounding box around the part and the grasp location. After training skills for each of the resulting 150 tasks in our coverage-based simulation, we evaluate them on all the tasks to generate binary success labels.

Real World Results: We evaluate all the approaches on 10 sets of 10 Lego-stacking tasks each using $c_{rob} = 10, c_{hum} = c_{fail} = 100$ and $c_{demo} = 200$. We choose $c_{demo} > c_{hum}$ as it takes much more time to provide a demo than for the human to stack the Lego themself, while $c_{hum} = c_{fail}$ as a failed robot execution can be fixed quickly by a human. c_{rob}

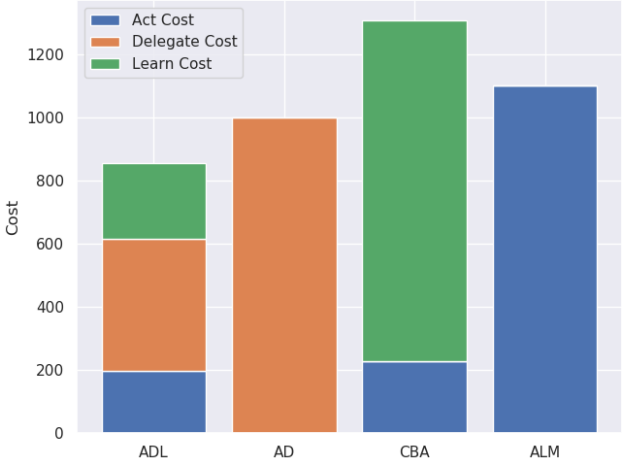

Fig. 8: Comparison of ADL vs baselines in the Lego stacking domain using $c_{rob} = 10, c_{hum} = c_{fail} = 100$ and $c_{demo} = 200$. ADL is the only planner that leverages synergy among acting, delegating and learning to complete tasks at minimum cost.

is the smallest cost as we value human time much more than robot time in this domain. Figure 8 shows the total cost of completing all tasks using each of the methods. AD delegates all tasks as the skill library is empty at the beginning, CBA asks for too many human demos as it doesn't take into account their relevance to the task set and ALM doesn't ask for any demos as its upfront cost is higher than failing at a task. In contrast, ADL finds the optimal synergy among all the three options to solve the tasks with minimum cost.

VIII. Conclusion and Future Work

We propose a planning and learning framework for completing n tasks with a human-robot team using minimum total effort. Our approach has two key components: (1) a *general* mixed integer programming formulation and (2) a learned *domain-dependent* precondition prediction model to predict the benefits of learning a new skill. Simulated and real world evaluations on two challenging manipulation domains indicate that our approach saves significant human and robot effort compared with approaches that do not plan ahead.

In the future, we are interested in extending the planner so that it can also optimize the order of tasks. We would also like to continue working on the precondition prediction problem to make it less data-hungry and more accurate by using multiple sources of data. Finally, a major limitation of the precondition prediction model is that it currently assumes each skill is trained on only one task. We would like to extend this to skills that are trained on a set of tasks which will allow the use of parameterized robot skills in our framework.

IX. Acknowledgement

The authors thank Kevin Zhang for help with robot experiments and Jayanth Krishna Mogali for discussions. This work is supported by ONR Grant No. N00014-18-1-2775 and ARL grant W911NF-18-2-0218 as part of the A2I2 program.

REFERENCES

[1] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, pp. 1–25, 2009.

[2] M. Cakmak, C. Chao, and A. L. Thomaz, "Designing interactions for robot active learners," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 2, pp. 108–118, 2010.

[3] E. Gribovskaya, F. d'Halluin, and A. Billard, "An active learning interface for bootstrapping robot's generalization abilities in learning from demonstration," in *RSS Workshop Towards Closing the Loop: Active Learning for Robotics*, vol. 86, 2010.

[4] B. Hayes and B. Scassellati, "Discovering task constraints through observation and active learning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4442–4449.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[6] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.

[7] B. Settles, "Active learning literature survey," 2009.

[8] M. Gombolay, R. Wilcox, and J. Shah, "Fast scheduling of multi-robot teams with temporospatial constraints," 2013.

[9] M. C. Gombolay, R. J. Wilcox, A. Diaz, F. Yu, and J. A. Shah, "Towards successful coordination of human and robotic work using automated scheduling tools: An initial pilot study," in *Proc. Robotics: Science and Systems (RSS) Human-Robot Collaboration Workshop (HRC)*, 2013.

[10] C. J. Shannon, L. B. Johnson, K. F. Jackson, and J. P. How, "Adaptive mission planning for coupled human-robot teams," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 6164–6169.

[11] K. H. Wray, L. Pineda, and S. Zilberstein, "Hierarchical approach to transfer of control in semi-autonomous systems," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016, pp. 1285–1286.

[12] C. Basich, J. Svegliato, K. H. Wray, S. Witwicki, J. Biswas, and S. Zilberstein, "Learning to optimize autonomy in competence-aware systems," 2020.

[13] R. W. Pew, "The speed-accuracy operating characteristic," *Acta Psychologica*, vol. 30, pp. 16–26, 1969.

[14] J. Y. C. Chen and M. J. Barnes, "Human–agent teaming for multirobot control: A review of human factors issues," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 1, pp. 13–29, 2014.

[15] C. J. Shannon, D. C. Horney, K. F. Jackson, and J. P. How, "Human-autonomy teaming using flexible human performance models: An initial pilot study," in *Advances in human factors in robots and unmanned systems*. Springer, 2017, pp. 211–224.

[16] T. Inagaki *et al.*, "Adaptive automation: Sharing and trading of control," *Handbook of cognitive task design*, vol. 8, pp. 147–169, 2003.

[17] P. Fitts, M. S. Viteles, N. L. Barr, D. R. Brimhall, G. Finch, E. Gardner, W. F. Grether, W. E. Kellum, and S. S. Stevens, "Human engineering for an effective air-navigation and traffic-control system, and appendixes 1 thru 3," 1951.

[18] A. Dearden, M. Harrison, and P. Wright, "Allocation of function: scenarios, context and the economics of effort," *International Journal of Human-Computer Studies*, vol. 52, no. 2, pp. 289–318, 2000.

[19] W. B. Rouse, "Adaptive allocation of decision making responsibility between supervisor and computer," in *Monitoring behavior and supervisory control*. Springer, 1976, pp. 295–306.

[20] M. Scerbo, "Theoretical perspectives on adaptive automation," 1996.

[21] D. B. Kaber, J. M. Riley, K.-W. Tan, and M. R. Endsley, "On the design of adaptive automation for complex systems," *International Journal of Cognitive Ergonomics*, vol. 5, no. 1, pp. 37–57, 2001.

[22] T. B. Sheridan, "Adaptive automation, level of automation, allocation authority, supervisory control, and adaptive control: Distinctions and modes of adaptation," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 4, pp. 662–667, 2011.

[23] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[24] M. Rigter, B. Lacerda, and N. Hawes, "A Framework for Learning from Demonstration with Minimal Human Effort," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2023–2030, 2020.

[25] R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg, "Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning," *arXiv preprint arXiv:2109.08273*, 2021.

[26] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, "Multi-task policy search for robotics," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2014, pp. 3876–3881. [Online]. Available: http://arxiv.org/abs/1307.0813.

[27] A. Kupcsik, M. P. Deisenroth, J. Peters, A. P. Loh, P. Vadakkepat, and G. Neumann, "Data-Efficient Generalization of Robot Skills with Contextual Policy Search," *Artificial Intelligence*, vol. 247, pp. 415–439, 2017.

[28] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[29] G. Konidaris and A. Barto, "Efficient skill learning using abstraction selection," in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[30] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *arXiv preprint arXiv:1907.03146*, 2019.

[31] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Symbol acquisition for probabilistic high-level planning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[32] O. Kroemer and G. S. Sukhatme, "Learning spatial preconditions of manipulation skills using random forests," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 676–683.

[33] M. Sharma and O. Kroemer, "Relational learning for skill preconditions," *arXiv preprint arXiv:2012.01693*, 2020.

[34] B. Da Silva, G. Konidaris, and A. Barto, "Learning parameterized skills," *arXiv preprint arXiv:1206.6398*, 2012.

[35] A. Kolobov, "Planning with markov decision processes: An ai perspective," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–210, 2012.

[36] M. Gombolay, A. Bair, C. Huang, and J. Shah, "Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences," *The International journal of robotics research*, vol. 36, no. 5-7, pp. 597–617, 2017.

[37] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021. [Online]. Available: http://www.gurobi.com

[38] V. V. Vazirani, *Approximation algorithms*. Springer Science & Business Media, 2013.

[39] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for mdps." *ISAIM*, vol. 4, p. 5, 2006.

[40] Nvidia. (2020) Isaac sim. [Online]. Available: https://developer.nvidia.com/isaac-gym