Effective Robot Teammate Behaviors for Supporting Sequential Manipulation Tasks

Bradley Hayes[1] and Brian Scassellati[1]

***Abstract*— In this work, we present an algorithm for improving collaborator performance on sequential manipulation tasks. Our agent-decoupled, optimization-based, task and motion planning approach merges considerations derived from both symbolic and geometric planning domains. This results in the generation of supportive behaviors enabling a teammate to reduce cognitive and kinematic burdens during task completion. We describe our algorithm alongside representative use cases, with an evaluation based on solving complex circuit building problems. We conclude with a discussion of applications and extensions to human-robot teaming scenarios.**

I. Introduction

Real-world manipulation tasks typically involve handling and reconfiguring multiple objects to accomplish abstract goal criteria. These tasks involve a high-dimensional action space, making satisfying solutions potentially very difficult to compute. The use of traditional symbolic planning to derive a task solution may produce sub-optimal or infeasible physical impositions on an agent, rendering it insufficient for many tasks of interest. Avoiding this problematic detachment from the physical world by using a motion planner to search and evaluate feasible actions within the task space yields a separate set of issues, as this approach can make searching the task space very computationally expensive.

Task and motion planning (TAMP) addresses these challenging problems, merging the task-level goal demands with the physical reality and geometric complexities of motion planning [1]. Even with the state-of-the-art in TAMP approaches, many tasks remain intractible within online planning scenarios for a variety of reasons, including object density, goal flexibility, and kinematic complexity. These issues become even more prominent when considering multi-agent problems, as geometric back-tracking can be prohibitively expensive [2]. Multi-agent approaches to TAMP generally assume an 'equal-partners' coordination paradigm, where agents are jointly working towards directly achieving a shared goal, such that the same planning heuristics can be used across the team.

In this work, we propose a novel approach for use within multi-agent TAMP domains, utilizing a 'leader-assistant' teamwork paradigm, explicitly enabling supportive roles for agents to assume. These supportive roles facilitate task completion for 'leader' agents (that employ traditional TAMP solution mechanisms) by performing actions to reduce the cognitive (symbolic planning) or kinematic (motion planning) difficulty of the on-task actions they perform. Such roles are critical for human-robot collaboration [3].

In particular, we focus on *sequential manipulation tasks*, where high level goals are achieved through a sequential series of object manipulations. This broad class of tasks is relevant to personal assistant robots and industrial robots alike (despite vast differences in expectations, capabilities, and tooling), making it an excellent candidate for application to the leader-follower teaming paradigm. In particular, the representative sequential manipulation task we utilize in this paper is that of circuit building within a spatially constrained environment. Given a collection of resources (circuit pieces such as wires, resistors, LEDs, etc.), there exists a set of motor action sequences that each yield a constructed circuit with the characteristics specified by the task's goal state.

In addition to these motor behaviors that result in the successful assembly of the circuit, there exist actions that facilitate task completion without directly contributing towards reaching the goal state. These actions (such as removing an obstacle on the table), which we define as *supportive actions*, are optional actions that contribute indirectly towards a more rapid or less challenging task solution. These behaviors are not nearly as well studied as their on-task counterparts, but become increasingly relevant as human-robot teaming and teams of robots with heterogeneous capabilities become more commonplace.

In the sections that follow, we provide a brief survey of related work, formally introduce our selected evaluation domain, and introduce a novel algorithm for improving team performance in sequential manipulation tasks within the leader-assistant teamwork paradigm. Our algorithm accomplishes this through the evaluation, selection, and execution of actions with the intent to optimize a task's environment for a collection of potential execution policies. By optimizing towards the maximization of a collaborator's performance, it is possible not only to improve makespan time and to reduce a partner's cognitive load, but also to manipulate them into making better decisions towards following more optimal policies. Our algorithm allows a supportive robot to drive its team to perform well even in the absence of intra-agent communication, with only coarse approximations of collaborators' kinematic abilities, while only assuming rational, goal-aligned teammates. Finally, we conclude with an evaluation within a circuit building domain, exploring algorithm performance and behavior.

[1]Computer Science Department, Yale University, 51 Prospect Street, Connecticut, USA {bradley.h.hayes, brian.scassellati}@yale.edu

This work was supported by Office of Naval Research Grant #N00014-12-1-0822 ("Social Hierarchical Learning")

II. Related Work

Sequential manipulation tasks and multi-agent teamwork are rich research topics that are of great interest within the robotics community. Solving task and motion planning problems is well known to be a challenging area, with a great deal of tools and approaches developed to leverage more powerful abstractions and heuristics for the purpose of reducing search complexity [4], [5] or broadening the types of constraints that can be handled [6], [7], [8], [9].

The majority of multi-agent work within TAMP problems has focused on an 'equal partners' paradigm of teamwork, producing innovative solutions that rely on the decomposition of a task into subcomponents that can be distributed across agents to maximize efficiency and capability [10], [11], [12]. The ability to convey intent and to learn from or anticipate the behaviors of others is also critical to multi-agent TAMP. Accordingly, recent work on collaboration has been published with a focus on motion planning that implicitly conveys intent [13], as well as motion planning that accommodates the expected motions of others within an interaction [14].

Solving these TAMP problems is also of particular interest to the scheduling research community. Recent work in multi-agent scheduling has yielded a constraint-based method of performing extremely fast scheduling [15] that can be applied to heterogeneous teams of agents once a TAMP solution is translated into a simple temporal problem [16]. Each of these presented works handles an important subset of the issues central to multi-agent task and motion planning, including the management of diverse capabilities, world models, agent models, and behavioral expectations. Our contribution builds upon these considerations by broadening the utility of agents, even those without the capability to directly achieve task goals, by developing a formalized means of generating supportive actions to assist those that can.

III. Task and Motion Planning Domain

Our problem domain is described via traditional planning operators and symbolic predicates (similar to [17]) with the addition of special functions to handle geometric constraints. Goal states are described by these abstract predicates, providing a description of a desired world state that may include concepts that do not clearly map to a physical description of the individual components that contribute to it (e.g., a predicate *explicitly* requiring a circuit to be closed that happens to contain a diode *implicitly* requires the diode's anode and cathode to be properly connected). Operators are representative of robot motor primitives, parameterized by potentially high-dimensional continuous values that may indicate kinematic positions, grasp types, or object poses. We address the parameterization issue through the standard practice of discretizing the parameter space via sampling.

Despite discretizing these variables into finite domains, it remains impractical to enumerate the symbolic effects of each possible action, as second-order, third-order, or arbitrarily distant effects may impact aspects of the environment (such as object reachability or placement eligibility). As such, we implement a similar strategy in keeping a state representation with predicates comprised of static literals alongside geometric state-dependent predicates that are computed on-demand.

Recent work has introduced planning concepts that greatly speed the execution-time computation of valid plans by leveraging clever insight into the underlying geometric requirements for TAMP problems [4]. One of these insights is the need to efficiently characterize the free configuration space of the robot and the manipulable objects within the scene, which is not possible given static preconditions. Accordingly, in addition to classical static planning literals, we incorporate geometric conditionals to ground the symbolic planner within the physical world as it seeks task solutions.

Choosing optimal behaviors within this problem domain is challenging to accomplish in acceptable timeframes and can easily be intractible. Generating plans via symbolic planning can typically be accomplished within reasonable timeframes but will not account for geometric complexities and may not provide realistic estimates of action durations. The converse, attempting to plan by finding satisfying sequences of robot and environment configurations through motion planner sampling and pathfinding, rapidly becomes intractable with problem complexity.

Due to this disconnect between 'possible' and 'efficient' action sequences, seemingly reasonable plans may be far removed from desirable plans both in terms of complexity and resource utilization. Even if motion planners were capable of finding acceptable action sequences without incorporating planning abstractions, this would fundamentally limit the ability of an agent to reason about its task or environment. Many relevant tasks have abstract components that go beyond individual resource usage, and are neatly encapsulated within traditional planning formulations. For example, a goal predicate may specify that a '500Ω resistor' is soldered to 'Wire #1', but it may not matter which particular 500Ω resistor is used to accomplish the goal. As such, it is both desirable and advantageous to seek mechanisms incorporating both types of planner for sequential manipulation tasks.

IV. Supportive Behaviors

We define *supportive behaviors* as optional, off-goal actions that contribute indirectly towards a more rapidly satisfiable and/or less difficult task solution.

Particularly within assembly domains, there exist actions that can be performed to ease cognitive or physical burdens that don't directly contribute towards the completed construction. One such example is organizing circuit components into discrete piles, such that all wires are axis-aligned and together, or physically sorting resistors in ascending order of resistance. By performing these organizational behaviors that are tangential to the task goal, future part-seeking actions are facilitated and the overall makespan for the assembly task may be reduced. For motion planning, actions that remove obstacles from the environment of an object grasping behavior will reduce the problem complexity by allowing for more valid or possibly less complex motion paths.

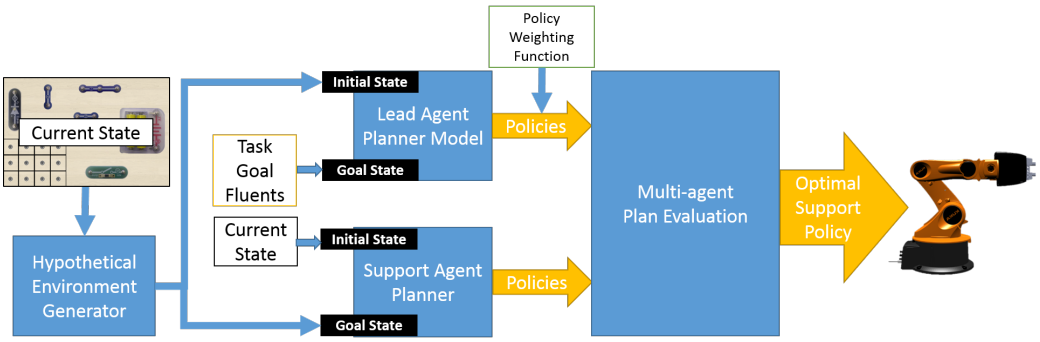


Fig. 1: A graphical representation of the supportive behavior generation pipeline we propose.

Similarly, positioning objects near their final positions (e.g., screws near appropriate guide holes, components in an approximation of their final configuration, etc.) may reduce the cognitive load on the lead agent actually performing the assembly, as it becomes less ambiguous how to assemble the components. Within the TAMP domain, this is equivalent to reducing the symbolic planning complexity. The key insight being leveraged involves organizing the environment into a configuration favorable to a lead agent's planning heuristics, such that the agent's search order over objects to use for action parameterization is optimized.

The goal of the algorithm presented in this work is the autonomous generation of these types of behaviors, with the explicit goal of improving a collaborating agent's performance through the simplification of various aspects of a task via calculated environmental manipulations. The ability to optimize an agent's behaviors for a supportive role broadens its utility, especially within problem domains in which the agent cannot fully perform a specified task either due to liability, lack of capability, or unacceptable risk. Further, the potential reduction in cognitive load attained by incorporating a supportive agent into a problem domain, particularly in the case of a human-robot team, can serve to increase worker productivity and safety.

Though we are concerned explicitly with teams of robots performing tasks, we make no assumptions regarding available information transfer between collaborators. As such, the algorithm we present is based upon maximizing the expected performance improvements achieved by supportive actions across a range of possible execution policies. As communicative capabilities are introduced or agents are given sufficient information to model each other's preferences and capabilities, the likelihood of a supportive robot providing increasingly beneficial actions improves.

We require the supportive robot to minimally have a model or approximation of the lead collaborator's kinematics for the purposes of estimating plan completion times given particular environment configurations.

V. Approach

We frame the supportive behavior problem as one of indirect policy optimization, decoupling the support role from the lead agent's planner. Given exclusion criteria such as operators (actions) the supporter is unable to utilize, spatial regions the supporter is not permitted to occupy, objects a supporter is prohibited from manipulating, or predicates the agent is not allowed to effect on the world, we generate a permissable action sequence resulting in an environment that minimizes the expected execution time of the lead agents working on the task. Conceptually, we accomplish this by finding the most desirable environmental configurations for the set of likely execution policies being executed by the lead agents, limited by the supportive agent's ability to effect the environment in a way that achieves this ideal environmental configuration. This process is summarized in Figure 1.

A. Formalizing the Supportive Behavior Problem

We define the supportive behavior problem as the tuple $\Sigma = (T, \Pi_T, a_s, C_s, s_c, P)$ where

- $T = \{A, O, C, s_0, s_G\}$ is a TAMP problem such that:
  - A is a set of ('lead') agents
  - O is a set of operators in the form of motor primitive prototypes, representing unparameterized action types
  - C is a capabilities mapping function between agents and operators, indicating actions that may be performed by each agent in A
  - s_0 is the set of predicates precisely specifying the starting environment state
  - s_G is the set of predicates specifying the goal state of the task
- Π_T is a set of symbolic plan solutions for T
- a_s is a supportive agent
- C_s is a mapping function indicating operators from T usable by a_s
- s_c is the current environment state
- P is a set of predicates or partially specified predicate prototypes that indicate prohibited operator parameters

A solution to T is a policy $\pi \in \Pi_T$ that achieves state s_G through a specified sequence of operators in O with geometrically sound parameterizations, executed by a subset of agents in A. A solution to the supportive behavior problem Σ is a plan π_s which, when executed by a_s, reduces the expected duration for an agent in A of physically executing a plan in Π_T, or reduces the expected search complexity required for agents in A to find solutions $\Pi \subseteq \Pi_T$.

To solve the supportive behavior problem described by Σ, we sample from and reason about alternative environmental configurations, evaluating them based on metrics such as the estimated cognitive (planning complexity) or physical (motion complexity) demands imposed on the agents in the original TAMP problem. This optimization must account for the anticipated time costs associated with achieving the hypothesized improved environment states, as well as the relative likelihoods of a lead agent following particular plans in Π_T.

B. Constructing Hypothetical Environments

At the current environment state s_c we build a set of parameterized operators executable by a_s, which we refer to as O_s. For each action in the support robot's capability set C_s, we sample parameterizations from a discretization of the task world, and valid instantiations are added to O_s. For a 'pick' operator, the parameterizations of this action include the set of all objects except those whose effects match predicate prototypes specified in P. For example, if the current environment has predicate 'inCircuit(battery$_1$)' and 'inCircuit(*)' is a member of P, a prohibited action parameterization may be 'pick(battery$_1$)' as it would remove battery$_1$ from the circuit. In the case of a 'place' operator, parameterizations include samples drawn from the set of possible legal placement positions and poses in the task area for each eligible object.

With O_s constructed, we then sample different environmental configurations ('hypothetical environments') obtainable by changing individual aspects of the environment. The sampling method we use involves choosing a single object from the scene and moving it to a randomly sampled new location and/or pose. For each hypothesized environment, a plan is determined using actions in O_s enabling the support agent to reconfigure the current environment into the hypothesized state. An estimate is computed for the execution duration of each plan associated with a hypothetical environment (unobtainable configurations are discarded). These duration estimates are later used to evaluate the inconvenience associated with the support robot creating this environment (and tying up the associated resources). Each attainable hypothetical environment is finally encoded as the tuple $\xi = \{\pi, d, s\}$ indicating a plan composed of allowable parameterized operators, an estimate of the duration required to achieve the desired environment, and a set of predicates describing the resulting environment state. We refer to the set of all hypothetical environment tuples as Ξ.

C. Plan weight determination

We establish a set of plan weights to influence the type of support provided. The selection of this weight function can have strong effects on the behaviors generated, and as such we characterize three types of weighting schemes that can be used to direct a system toward particular outcomes. To establish these relative weights, we utilize a plan execution duration approximator outlined in Algorithm 1. We define

$$m = \min_{\pi \in \Pi_T} \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1)$$

to be the duration of the shortest (temporally optimal) known plan. Here we describe three cases of useful weighting functions and their resulting behaviors:

1) A conservative optimization function that weights plans relative to their estimated optimality of execution duration. We chose
$$w_\pi = \left(\frac{m}{\text{duration}(T, \pi, \emptyset, s_0, f(x) = 1)} \right)^2$$
for each known plan $\pi \in \Pi_T$, though any similar positive, monotonically decreasing function can be used to produce similar results.
2) A greedy optimization function that optimizes for what is estimated to be the best known plan and ignoring consequences for all other possible policies, such as:
$$w_\pi = \begin{cases} 1 & ; \quad \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1) = m \\ 0 & ; \quad \text{otherwise} \end{cases}$$
3) An aggressive optimization function that not only prioritizes making the best plans better, but also making the worst plans even worse, with the intention of driving a rational agent away from selecting poorly. This is desirable in cases similar to those where a lead agent may either first connect a circuit to a power source or first build the remainder of the circuit. Attaching the power source may increase the danger of the construction operation, and thus makespan, due to the need for increased care in assembly. This can be avoided by the support agent introducing a resource conflict for the power source connection piece, removing it from the available actions the lead may take.
Functionally, this can be accomplished by providing undesirable plans with negative plan weights. It is important to keep the magnitude of negative weights less than the positive weights (using a normalization factor α), or the support robot may perpetually block task progress in the case of partial plan overlap between 'good' and 'bad' plans. A functional example of such a weighting scheme can be achieved by modifying the results from the conservative weighting presented above. We use ϵ to denote a value equal to or slightly greater than m:
$$w_\pi = \begin{cases} w_\pi & ; \text{ duration}(T, \pi, \emptyset, s_0, f(x) = 1) \leq \epsilon \\ -\alpha w_\pi & ; \text{ otherwise} \end{cases}$$

D. Environment state analysis

Combining the information gathered thus far, we choose the best supportive action plan $\xi \in \Xi$ according to:

$$\min_{\xi \in \Xi} \sum_{\pi \in \Pi_T} w_\pi * \text{duration}(T, \pi, \xi, s_c, \gamma)$$

where w_π indicates a plan's associated weight and the duration function computes an estimate of the TAMP solution accounting for the cost of the supportive behavior. The final argument to the duration function, $\gamma : \mathbb{Z}^+ \rightarrow [0, 1]$, is a decay function used to modulate the prioritization of supportive actions causing near-term effects over those that

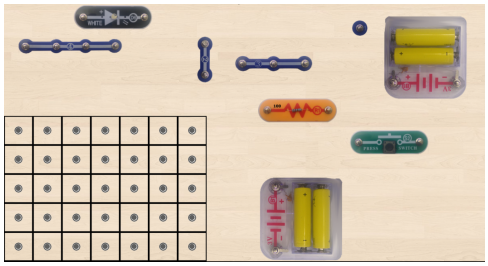


Fig. 2: One sample scenario used in our evaluation. In this task, the lead agent must create a circuit using two power sources wired in series to drive an LED on a switched circuit. Random configurations of objects were utilized to create a variety of starting conditions.

Fig. 3: Example of a SnapCircuits solution. Snap buttons indicate valid connection points between components, which must be joined to form circuits in a manner that satisfies the implicit 3D spatial constraints imposed by the pieces available and the desired goal circuit function.

cause longer-term consequences. Providing a function such as $\gamma = \{f(x) = 1\}$ removes this decay functionality.

Algorithm 1: TAMP Solution Duration Estimation

Input: TAMP Problem T, TAMP solution plan π, Supportive plan tuple ξ, Start state s_c, Decay function γ
Output: Estimated execution duration of π

```
// PLAN_NULLIFICATION_PENALTY is a value greater than the cost of the worst known plan
elapsed_time ← 0;
current_state ← s_c;
foreach i in 1 to |π| do
    Operator o ← π[i];
    if o.preconditions not satisfied by current_state then
        return PLAN_NULLIFICATION_PENALTY;
    step_time ← 0;
    if the set of objects and physical space utilized between o and ξ.π ≠ ∅: then
        step_time ← ξ.d; // Must wait for support plan to finish
    step_time ← step_time + execution_duration(o);
    elapsed_time ← elapsed_time + step_time ∗ γ(i);
    ξ.d = max(0, ξ.d − step_time);
    apply_operator_effects(o, current_state);
return elapsed_time;
```

E. Assumptions and Weaknesses

The method we describe can become arbitrarily computationally expensive depending on the approximations used for the duration estimation function and the size of the known plan set Π_T. As this puts calls to a motion planner in the inner loop of a computation that is highly dependent on the current environment (and challenging to pre-compute), the selection of how coarsely to approximate an agent's kinematics and reachability becomes quite important. As such, we recommend initially choosing rapidly computable, low-fidelity approximations relevant to the task domain, such as a cost function that computes the shortest, collision-free path for an agent's end-effector to take (independent of its kinematic chain) in the case of tabletop pick-and-place operations.

Once the space of candidate environments and policies is sufficiently reduced, higher fidelity simulations can be utilized, a common technique for expensive evaluations. Directly as a result of this expensive computation in the inner-most loop of the planning computation, our proposed decoupled approach parallelizes better and converges far faster than an interleaved approach in which the lead and support roles are not decoupled. While this improved performance occurs at the cost of solution optimality, we maintain that the benefits of achieving 'good enough' improvement on a rapidly solvable timescale are far more useful than achieving optimality on a far less rapid timescale.

VI. Evaluation

We performed an evaluation of our algorithm using a collaborative circuit-building task within a simulation environment (Figure 2). The circuit-building task is closely modeled after SnapCircuits, a popular children's toy. Using a grid-like board and blocks with electronic components attached to them (Figure 3), children use this toy to build simple circuits or devices to learn about electronics. This puzzle is a particularly interesting domain for study as it contains complex, abstract concepts that cannot easily be represented as static literals within operator effect specifications (such as circuit resistance or shorted paths). Additionally, there are non-trivial spatial concerns, as pieces may not be connected side-by-side. Instead, they must be connected via the snap buttons on the blocks (Figure 3), resulting in an extra dimension of placement (height) to consider. For example, one could not connect two LEDs in series directly, as a wire piece would be necessary to bridge them together.

This domain is particularly attractive due to the prevalence of cases where multiple valid solutions exist that differ in terms of their optimality of resource usage. By providing an agent with an excess of SnapCircuit blocks, it is possible to influence it to implement less optimal solutions that are more immediately apparent. We use this case in our evaluation to demonstrate our algorithm's ability to influence an agent

away from utilizing these sub-optimal plans, by making more desirable solutions more obvious.

A. Task Environment

We utilize a predefined grasp library, allowing our simulated agents the ability to grasp and place blocks at 90 degree increments. Task execution occurs around a table with full shared reach between the lead agent and supporting agent. Though this reach distance is configurable within the simulation we utilized, we did not add this additional complexity in order to more clearly illustrate the conditions under which our algorithm generates particular supportive behaviors. The snap board, upon which circuits are constructed, is accessible only to lead agents. Accordingly, support agents are prohibited from executing operators with parameterizations that result in environment effects directly modifying anything on the snap board (specified in the prohibited predicate set $\Sigma.P$ initially mentioned in Section V-A).

We conservatively model collisions, maintaining a virtual buffer space around each robot during action execution. In the event of an anticipated collision, the agent with the most priority is permitted to continue its task while the other agent receives a command to return to its home position just off-table. Priority is determined by resource utilization, with the lead agent winning ties. In practice, during conflicts this manifests as the support agent having to place any objects it is carrying on the table, returning to its home position, and then attempting another action. This resolution strategy respects important considerations relevant to real-world collaborative robotics, where robots do not necessarily have the ability to either coordinate behaviors or to explicitly communicate with their collaborators.

B. Scenarios

Our evaluation concerns itself with two possible worker team scenarios, based upon the level of knowledge and coordination assumed between the lead and support agent:

The first scenario we consider is that of *unpredictable team behavior*. In this scenario, the lead agent is aware of a random subset of $\Sigma.\Pi_T$. This example has the lead agent following the best valid plan of that random subset with the assistance of a support robot. This support robot attempts to maximize its assistance without specific knowledge about the lead robot's policy, using a balanced plan weighting scheme that favors more optimal solutions (weighting case 1).

The second evaluation concerns *naïve heuristic-driven team behavior*. In this scenario, the lead agent must attempt to solve the task online, aided only by basic planner heuristics that favor plans following a simple prioritization scheme: actions are parameterized using pieces most proximal to the board in their current orientations, before branching out to explore more spatially or rotationally distant alternatives. This agent is also unconcerned with optimality, which may have otherwise affected its planning strategy to be more conservative with the circuit blocks utilized or final configuration achieved. For this example, we utilize a plan weighting scheme that can actively disincentivize poor plans while driving the lead agent towards more optimal solutions through careful environmental reconfiguration (weighting case 3).

Agent teams were evaluated using lead and support pairs, characterized by their movement speed as being either fast (100% speed) or slow (33% speed). We evaluated support scenarios where a lead agent was paired with either a less, equal, or more rapidly moving helper. While we evaluated several circuit building activities (with differing component availability/goal specifications) with our algorithm, we specifically report on our most complex task since the relative benefits between conditions did not differ strongly with variations in goal complexity for this class of task.

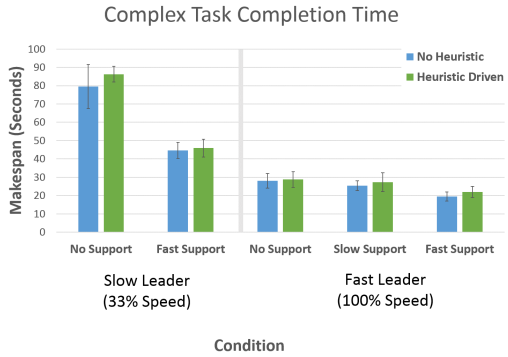
C. Circuit Building Task Results

As shown in figures 4a and 4b, our algorithm successfully generated behaviors that reduced the cognitive load and physical effort required by a lead agent solving a complex circuit-building sequential manipulation task. Particularly encouraging for applications into ad-hoc robot teaming is the result from our first scenario, where a knowledgable support robot and a competent lead robot could successfully collaborate without any direct coordination. Our second scenario is especially encouraging for human-robot teaming, where it is possible to leverage the potentially superior symbolic reasoning and planning of a robot while still deriving the substantial benefits attained via the dexterous manipulation and broader contextual awareness of human workers.

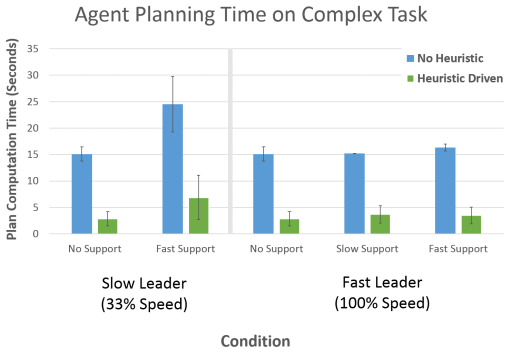
Within our makespan (task completion time) evaluation (Figure 4a), *our results show that supportive agents had helpful effects that drove average task completion times down universally*. We show that a slow lead robot with a fast *supportive-behavior only* agent was able to perform competitively with a single fast lead robot. This improvement demonstrates that through our contribution, a collection of potentially less costly, diversely capable agents can approach or surpass the task completion quality of the less realistic case of a single, universally capable agent.

In this condition, the support robot provided an efficiency increase of 44%, yet did not require the same task-relevant capabilities (precision, tooling, etc.) as the lead. Despite this, its inclusion substantially improved task performance through autonomously generated, off-task, supportive actions. We also see a benefit across task executions for agents paired with slower or equally rapid supportive collaborators, with average time improvements of 9.8% and 30.5% over the unsupported agent scenario, respectively. These gains were consistent regardless of whether the lead agent was using a planning heuristic known to the support agent. As this task takes place over a small, shared work surface, one can reasonably expect the magnitudes of improvement to scale with task environment size.

We also evaluated the effects of our support algorithm on a lead agent's ability to rapidly plan high quality solutions for these circuit building tasks (Figure 4b). We see this planning time as analogous to the cognitive load imposed on the lead agent while solving these circuit building tasks. The heuristic



(a) Average makespan over multiple random initial environmental configurations of the base task. Adding a support robot provided substantial task completion time improvement even though the supportive agent could not directly contribute towards the task completion.



(b) Mean planning duration of the lead agent for solving various SnapCircuits TAMP problems. In multi-agent scenarios, due to the decoupled nature of our approach, the agents had to replan when resource or spatial conflicts emerged (as compared to no replanning occurring during the unsupported conditions).

Fig. 4: Planning and execution performance results for a variety of circuit construction tasks.

driven results we present show that *a supportive agent, aware of a lead agent's planner heuristic(s), can dramatically reduce the search space or planning burden of a task while still maintaining high makespan performance* by autonomously reconfiguring the environment to be more favorable for the thought process of the lead agent. Notably, it was important for the support agent to be faster than the lead as it may have needed to manipulate objects before the lead had a chance to use them. In the case of a lead agent paired with a faster support agent, the 10% average loss in task completion efficiency between the unsupported agent conditions (non-heuristic driven vs. greedy heuristic driven) disappears, while planning speed improvements largely persist. This benefit persists despite our implementation's need to fully replan when conflicts arise. The addition of motion plan caching would further improve these results.

VII. Conclusion

We conclude by summarizing our primary contribution in this work: an algorithm that generates supportive behaviors in ad-hoc multi-robot teams for sequential manipulation tasks. By utilizing an approach that proposes and evaluates possible 'desirable' environment states in the context of potential task execution plans, we present a novel idea that can be used to autonomously create off-goal behaviors that improve robot-robot and potentially human-robot team performance. Alongside our algorithm presentation, we show alternative weighting schemes that can be employed to produce behaviors that can be tailored to particular team dynamics or task considerations. Finally, we present an evaluation of our work within a complex circuit-building domain, showing positive effects on task completion speed and cognitive load.

References

[1] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009.

[2] F. Lagriffoul, D. Dimitrov, A. Saffiotti, and L. Karlsson, "Constraint propagation on interval bounds for dealing with geometric backtracking," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 957–964.

[3] B. Hayes and B. Scassellati, "Challenges in shared-environment human-robot collaboration," in *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2013) Workshop on Collaborative Manipulation*, 2013.

[4] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[5] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, pp. 1194–1227, 2013.

[6] A. Bhatia, M. R. Maly, E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *Robotics & Automation Magazine, IEEE*, vol. 18, no. 3, pp. 55–64, 2011.

[7] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao, "Perception, planning, and execution for mobile manipulation in unstructured environments," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 58–71, 2012.

[8] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 3684–3691.

[9] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 5002–5008.

[10] R. Alami, F. F. Ingrand, and S. Qutub, "A scheme for coordinating multi-robots planning activities and plans execution," in *13th European Conference on Artificial Intelligence*, 1998, pp. 617–621.

[11] M. Dogar, R. A. Knepper, A. Spielberg, C. Choi, H. I. Christensen, and D. Rus, "Towards coordinated precision assembly with robot teams," *Proceedings of the 2014 International Symposium on Experimental Robotics*, 2014.

[12] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "Ikeabot: An autonomous multi-robot coordinated furniture assembly system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 855–862.

[13] A. Dragan and S. Srinivasa, "Generating legible motion," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[14] J. Mainprice and D. Berenson, "Human-robot collaborative manipulation planning using early prediction of human motion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 299–306.

[15] M. Gombolay, R. Wilcox, and J. A. Shah, "Fast scheduling of multi-robot teams with temporospatial constraints." in *Robotics: Science and Systems*, 2013.

[16] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial intelligence*, vol. 49, no. 1, pp. 61–95, 1991.

[17] J. Hoffmann and B. Nebel, "The ff planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.