2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

Cognitive Factories with Multiple Teams of Heterogeneous Robots: Hybrid Reasoning for Optimal Feasible Global Plans

Zeynep G. Saribatur Esra Erdem Volkan Patoglu

Abstract— **We consider cognitive factories with multiple teams of heterogenous robots, and address two key challenges of these domains, hybrid reasoning for each team and finding an optimal global plan (with minimum makespan) for multiple teams. For hybrid reasoning, we propose (i) modeling each team's workspace taking into account capabilities of heterogeneous robots, (ii) embedding continuous external computations into discrete symbolic representation and reasoning by combining different methods of integration, (iii) not only optimizing the makespans of local plans but also minimizing the total cost of robotic actions, where costs of actions can be defined in various ways. To find a global plan with minimum makespan, we propose a semi-distributed approach: we formulate the problem of finding an optimal coordination of teams that can help each other, prove its intractability, and describe how to solve this problem using existing automated reasoners. As a case study, we show applications of our hybrid reasoning and coordination approaches on a cognitive toy factory with dynamic simulations and physical implementation utilizing KuKa youBots and Lego NXT robots (supplementary video provided). We also present experimental results to discuss the scalability of these methods.**

I. INTRODUCTION

Multiple teams of robots with heterogeneous capabilities are commonly employed to complete a task in a collaborative fashion in many application domains, ranging from search and rescue operations to exploration/surveillance missions, service robotics to cognitive factories. In these domains, the goal is for all teams to complete their tasks as soon as possible, and should the need arise, teams help each other by lending robots.

In this paper, we consider cognitive factories with multiple teams of heterogeneous robots. Cognitive factory concept [1], [2] is a novel paradigm proposed to enhance productivity and ensure *economic sustainability and growth* in the manufacturing sector. Aimed towards highly flexible and typically small to medium size manufacturing plants, these factories are equipped with multiple teams of heterogeneous manufacturing tools, such as dexterous mobile manipulators. Since these factories are endowed with high-level reasoning capabilities, they can rapidly respond to changing customer needs and customization requests, and produce a large variety of customized products even in low quantities. Consequently, cognitive factories provide an ideal compromise between the flexibility of human workshops with cost-effectiveness of mass production systems.

This work is supported by TUBITAK Grant 111E116.

Z. G. Saribatur, E. Erdem, V. Patoglu are with Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey {zgsaribatur,esraerdem,vpatoglu}@sabanciuniv.edu

In the context of cognitive factories, we address two key challenges of such domains including teams of heterogeneous robots: hybrid reasoning (combining discrete task planning with continuous feasibility checks and perception) for each team and finding an optimal global plan (with minimum makespan) for multiple teams. In particular, we propose to use state-of-the-art automated reasoners (i) to endow each heterogeneous robot team with high-level reasoning capabilities in the style of cognitive robotics, such that each team becomes capable of planning their own actions; and (ii) to coordinate robot exchanges among the teams to ensure an optimal global plan. Both problems, planning for a team of robots and finding a coordination for multiple teams of robots, are shown to be NP-hard [3], [4].

For hybrid reasoning, we emphasize several core characteristics of cognitive factories, such as existence of heterogeneous robots with varying capabilities, ability of robots to execute concurrent actions, existence of complex (state/temporal) constraints/goal conditions, and provide a computational framework to find *feasible* and *optimal* local plans for each team. The proposed method is based on our earlier works on hybrid planning [5], [6] utilizing expressive nonmonotonic logic-based formalisms that allows us to embed external computations in continuous spaces, and the use of automated reasoners.

This paper extends our earlier studies on Cognitive Factories [2] to include hybrid reasoning. In particular, by combining discrete task planning with continuous feasibility checks and perception, we address existence of static/dynamic obstacles in the domain. For performing feasibility checks, we utilize pre-computation approach to embed information about static obstacles perceived by a Kinect RGB-D camera into the domain description, while we rely on guided replanning to account for possible robot-robot collisions [6]. Furthermore, we extend the domain to model heterogeneity of robots and conduct various optimizations on local plans. In particular, in addition to finding a local plan with minimum makespan, we further optimize the total cost of this plan by considering several other objectives relevant in cognitive factories: to minimize the number of robotic actions or to ensure that actions in a team are executed as early as possible or to minimize fuel consumption.

For finding an optimal global plan (with minimum makespan) for multiple teams, we advocate the use of a semi-distributed approach to protect privacy of workspaces and to reduce message-passing and computational effort. Privacy is a concern in micro manufacturing plants that specialize on prototyping pre-release products, while reduction of com-

978-1-4799-6934-0/14/$31.00 ©2014 IEEE

Authorized licensed use limited to: Universita degli Studi di Genova. Downloaded on June 13,2024 at 12:11:59 UTC from IEEE Xplore. Restrictions apply.

munication among teams may be preferable when the means of communication is lossy or costly. Furthermore, a semi-distributed approach is advantageous in that it reduces the size of the global domain into manageable pieces, and provides a solution methodology that can utilize parallelization of computations.

To find an optimal global plan, we capitalize on the fact that each team is capable of hybrid reasoning, and use automated reasoners to find an optimal coordination as in our earlier work on optimal decoupled planning [4]. This paper extends our earlier study on coordination between teams of robots, by considering heterogeneous capabilities of robots and generalizing the formal framework (including definitions, mathematical modeling and formalizations) accordingly. We also prove that, with the addition of heterogeneity in teams and robot exchanges, the complexity of the coordination problem does not get harder than the case with homogeneous teams; that is, we show that the coordination problem is still NP-hard.

We use state-of-the-art automated reasoners to solve both the hybrid planning and coordination problems. In particular, we propose to utilize Answer Set Programming (ASP) [7] for reasoning, as its underlying non-monotonic semantics is general enough to represent the computational problems of interest and it provides very efficient solvers, such as CLASP [8]. ASP can handle many challenges: concurrency, the frame problem [9], the qualification problem [10], ramifications [11], nondeterminism, etc., while its efficient solvers can solve planning problems, which may involve complex goals and constraints, and perform optimizations. Furthermore, while computing a (discrete) plan, some feasibility checks (such as geometric/kinematic constraints on continuous trajectories) can be embedded in domain description, to ensure feasible (e.g., collision-free) plans.

Using ASP, we show applications of our hybrid reasoning and coordination approach through a case study of a cognitive toy factory, by means of dynamic simulations and physical implementation using KuKa youBots and Lego NXT robots. In these applications, local feasible plans with minimum makespans are further optimized by minimizing the total cost of actions with respect to three different action cost functions. We also investigate the scalability of our overall approach of hybrid reasoning and coordination, by means of some experiments in this domain. We observe that, since our approach allows utilization of parallelization, many large problems (e.g., involving 16 teams, 144 robots of 4 types, upto 32 robot transfers, with makespan upto 29) can be solved in a few minutes.

II. RELATED WORK

Our work on coordination of teams to find a global plan is similar to works on decoupling plans of multiple agents to coordinate their actions [12] in that local plans are computed by agents and then combined in order to compute a global plan. In these related works, a conflict-free coordination is ensured by specifying social laws before local planning [13], [14] or putting restrictions on local plans to be able to merge them or synchronize them into a global plan [15]–[18], or by exchanging information between teams about their partial plans or goals [14], [19], [20].

Our method is different from these works in that no restrictions are put on the order of actions for local planning of each team, and that teams do not exchange information about their plans or goals with each other. Each team communicates with the mediator, by only answering the mediator's yes/no questions; so the teams do not have to share private information with each other. Also, we do not assume that all teams are in the same workspace, or all robots are of the same sort. Moreover, our goal is not to find any coordination of teams that would allow decoupling of their local plans, but to find a coordination of teams for an optimal global plan (with minimum makespan); therefore, we also consider exchange of robots between teams. Furthermore, after the mediator informs the teams about when they are expected to lend/borrow robots, each team computes optimal local plans (with minimum makespans, and possibly with some other optimizations of total action costs).

The mediator in our approach is a neutral coordinator like in [21]; however, it does not negotiate with the teams but simply gathers information from them to find an optimal global solution. The mediator does not know anything about the teams's goals, tasks or workspaces, and the teams do not know what the mediator is trying to optimize. In that sense, our approach is also different from the existing works on task assignment and scheduling [22]–[25], because the mediator does not try to assign/schedule tasks but only informs the teams about when they are expected to lend/borrow robots.

Although robots can be considered as shared resources in our approach since they are borrowed/lent between teams, our work is different from the existing approaches on resource allocation in a multi-agent time-constrained domain [26]–[28], because the mediator does not require any information about local plans, ordering constraints on actions, or causal links, to decide for resource allocation. Although robot exchanges modifies the teams, our work is different from the works on team formation [29], [30] as well, because our method does not aim for deciding how or when to join teams. Finally, although each team utilizes feasibility checks, like collision checks, our work is different from various works on multi-robot motion planning [31]–[33], because our method also considers task planning.

III. ANSWER SET PROGRAMMING

The idea of Answer Set Programming (ASP) [7] is to represent a problem as a "program" whose models (called "answer sets" [34]) correspond to the solutions. The answer sets for the given program can then be computed by special implemented systems called ASP solvers, such as CLASP [8], which has recently won first places at various automated reasoning competitions. Due to the continuous improvement of the ASP solvers and highly expressive representation language of ASP, ASP has been applied to a wide range of areas, including industrial applications [35]–[37].

Authorized licensed use limited to: Universita degli Studi di Genova. Downloaded on June 13,2024 at 12:11:59 UTC from IEEE Xplore. Restrictions apply.

In this study, we use ASP for modeling action domains and combinatorial search problems, and an ASP solver as an automated reasoner. We consider ASP programs (i.e., nondisjunctive HEX programs [38]) that consists of rules

$$\mathit{Head} \leftarrow A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n$$

where $n \geq m \geq 0$, *Head* is an atom or $\bot$, and each A_i is an atom or an external atom. A rule is called a *fact* if $m = n = 0$ and a *constraint* if *Head* is $\bot$.

An external atom is an expression of the form $\&g[y_1, \ldots, y_k](x_1, \ldots, x_l)$ where $y_1, \ldots, y_k$ and $x_1, \ldots, x_l$ are two lists of terms (called input and output lists, respectively), and $\&g$ is an external predicate name. Intuitively, an external atom provides a way for deciding the truth value of an output tuple depending on the extension of a set of input predicates. External atoms allow us to embed results of external computations into ASP programs. They are usually implemented in a programming language of the user's choice, like C++. For instance, the following rule

$$\begin{aligned} \bot \leftarrow\ & \mathit{at}(r, x_1, y_1, t), \mathit{goto}(r, x_2, y_2, t), \\ & \mathit{not}\ \&\mathit{path\_exists}[x_1, y_1, x_2, y_2]() \end{aligned} \tag{1}$$

is used to express that, at any step t of the plan, a robot r cannot move from (x_1, y_1) to (x_2, y_2) if there is no collision-free trajectory between them. Here collision check is done by the external predicate *&path_exists* implemented in C++, utilizing the bidirectional RRT (Rapidly Exploring Random Trees) [39] as in the OMPL [40] library.

In ASP, we use special constructs to express cardinality constraints (e.g., a team needs at least two robots at time step t) or optimization statements (e.g., a plan is optimized by minimizing the total cost of actions). For instance, the following expression

$$\#\mathit{minimize}\ [\ \mathit{cost}(r, c, t) : \mathit{robot}(r) = c\ ] \tag{2}$$

is used to minimize the sum of all costs c of robotic actions performed in a plan, where costs of actions performed by robot r at time step t are defined by atoms of the form $\mathit{cost}(r, c, t)$.

IV. HYBRID REASONING FOR A TEAM OF HETEROGENEOUS ROBOTS

We find optimal feasible plans for a team of heterogeneous robots, using ASP as follows.

A. Heterogeneous Robots

Suppose that in a cognitive factory there are two types of worker robots: wet robots (that are liquid resistant), and dry robots. Wet robots can do all the actions, such as painting and stamping, whereas dry robots can only do stamping. In the presence of such heterogeneous robots with different capabilities, preconditions of actions should be specified accordingly. For instance, consider the action of a robot r painting a product b at time step t. We can describe the effects of this action in ASP by the rules:

$$\mathit{painted}(b, t+1) \leftarrow \mathit{paint}(r, b, t)$$

and its precondition that dry robots cannot paint, by the rules:

$$\leftarrow \mathit{paint}(r, b, t), \mathit{dry}(r).$$

In this way, we can specify which actions cannot be executed by which sort of robots.

B. Hybrid Reasoning

There are four different methods of integrating an external computation (e.g., feasibility checks or object detection) in an action domain description for hybrid reasoning, as described in [6]: (i) Pre-computation (PRE): external computations are done for all possible cases in advance and then this information (e.g., maintained as a set of facts) is embedded in an action domain description via external atoms, (ii) Interleaved computation (INT): external computations are directly embedded in an action domain description via external atoms so that external computations are done when they are needed during the search for a plan, (iii) Replanning (REPL): external computations for feasibility checks are done after a plan is computed and if the plan is found infeasible then a new plan is computed, (iv) Guided replanning (GREPL): similar to the previous strategy of replanning but a new plan is computed subject to the constraints obtained from previous feasibility checks. If the robotic application involves various external computations, then we can construct different levels of integration by deciding for an appropriate combination of methods for them, e.g., as in [5].

Consider, for instance, a workspace in a cognitive factory, with some obstacles. With the method PRE for integration: An object detection algorithm can be used to identify all locations l occupied by these obstacles, and the results of this external computation can be embedded into the formulation of a state constraint expressing that a robot r cannot be at a location l occupied by an obstacle at any time step t:

$$\leftarrow \mathit{at}(r, l, t), \&\mathit{obstacleAt}[l]()$$

where *&obstacleAt* is an external predicate whose value is determined by an object detection algorithm [41] using Point Cloud Library over data obtained by a Kinect RGB-D camera. We can also express a transition constraint to avoid collisions of robots with obstacles while the robots move from one location l_1 to another l_2:

$$\leftarrow \mathit{at}(r, l_1, t), \mathit{goto}(r, l_2, t), \&\mathit{collision}[r, l_1, l_2]()$$

where the external predicate *collision* checks for such collisions using Open Dynamics Engine (ODE).

With the method INT: We can interleave collision checks of robots with static obstacles, into automated reasoning so that checks are done as needed, as described in the previous section (see constraint (1)). We can also interleave collision checks of robots with movable obstacles:

$$\leftarrow \mathit{not}\ \&\mathit{path\_exists\_dynamic}[\mathit{goto}, \mathit{at}]().$$

The external predicate *path_exists_dynamic* gets as input the set of atoms of the form $\mathit{goto}(r, l, t)$ (describing which robot is moving where at time step t) and of the form $\mathit{at}(r, l, t)$ (describing the locations of all robots at time step t). It

Authorized licensed use limited to: Universita degli Studi di Genova. Downloaded on June 13,2024 at 12:11:59 UTC from IEEE Xplore. Restrictions apply.

returns true if and only if, for each time step and each robot, there is a collision-free motion plan from the location given by at at step t to the location given by $goto$ at step t.

With the method GREPL: After computing a plan, we can check for collisions of robots, using the external computations provided by ODE. If the plan is found infeasible, then we can identify which actions c are being executed at which state s when a collision occurs. Based on this information, we can ask for a new plan which does not involve execution of c at s, by adding a constraint to the planning problem description. For instance, if it is found by collision checks that two robots $R1$ and $R2$ cannot cross each other diagonally between locations A and B, then we can add the following constraint to the planning problem

$$\leftarrow \mathit{at}(R1, A, t), \mathit{at}(R2, B, t), \\ \mathit{at}(R1, B, t+1), \mathit{at}(R2, A, t+1)$$

to guide the ASP solver to find a new plan where the robots $R1$ and $R2$ do not exchange their locations A to B, respectively, at any time t.

C. Optimal Planning

As explained in the two previous subsections, once the capabilities of heterogeneous robots are described, with the hybrid reasoning methods, we can find feasible plans for a team of heterogeneous robots. To find optimal feasible plans, we model optimization problems in ASP (as described in Section III) with respect to different optimization functions.

To find a shortest plan (i.e., with minimum makespan), we can perform a linear search between a lower bound and an upper bound, as suggested in [42]. Alternatively, we can minimize the time step to reach a goal:

$$\#\mathit{minimize}\ [\ \mathit{goal}(t) = t\ ].$$

Once we find the length of a shortest plan, we can further optimize the plan, e.g., by trying to minimize the total cost of actions in a shortest plan. For instance, if the cost of every robotic action is defined as 1, then we can minimize the number of actions by the constraint (2). This may be useful for eliminating redundant actions executed in parallel with necessary actions.

Action costs can be defined as functions that depend on time. For instance, by defining the cost of all actions executed at time step t as $t+1$, we can ensure execution of actions as early as possible.

We can also define the cost of an action by estimating its duration or by the distance traveled. For instance, the following rule estimates the duration of the action of a robot r moving from location l_1 to another location l_2 by an external function *time_estimate*:

$$\mathit{cost}(r, c, t) \leftarrow \mathit{at}(r, l_1, t), \mathit{goto}(r, l_2, t), \\ \&\mathit{time\_estimate}[r, l_1, l_2](c).$$

The external function *time_estimate* estimates the duration in terms of a continuous trajectory for r between l_1 and l_2 computed by a motion planner, as in [43].

Note that, in a cognitive factory, finding local plans with shortest makespans helps minimizing the delivery lead time for a customized order. Further optimizations help improving robustness (e.g., by ensuring that critical processes are executed as early as possible), and conserve energy and achieve cost-effectiveness (e.g., by minimizing the total energy consumption of certain processes).

V. COORDINATION OF MULTIPLE TEAMS FOR AN OPTIMAL GLOBAL PLAN

We consider multiple teams of n types of robots, where each team is given a feasible task to complete in its workspace on its own using hybrid reasoning as described above, and where teams are allowed to transfer robots between each other. The goal is to find an optimal feasible global plan for all teams so that all tasks can be completed as soon as possible within at most k steps, where at most $\overline{m}_x$ robots of type x can be transferred between any two teams, and subject to the following constraints as in [4]:

- C1 Teams do not know about each other's workspace or tasks (e.g., for the purpose of privacy in micro manufacturing plants that specialize on prototyping pre-release products).
- C2 Lending/borrowing robots between workspaces back and forth is not desired (e.g., transportation of robots is usually costly, also, since tasks may be different in workspaces, robots need some tuning). Also, for similar reasons, robots can be transferred between two teams in a single batch.

To find a coordination of teams for such an optimal feasible global plan, we consider a semi-distributed approach as in [4]: a mediator gathers information from the teams to find a coordination for an optimal global plan (with minimum makespan); and then necessary information about this coordination is passed to each team so that they can utilize this information as part of their hybrid reasoning to find optimal feasible local plans. The mediator does not know anything about the workspaces of teams. To extend this approach to heterogeneous robots, we generalize the formal framework as follows.

The mediator asks yes/no questions of the following three forms to every team (in any order), for every $\overline{l} \leq k$, $l \leq \overline{l}$ and $m \leq \overline{m}_x$, $x \leq n$:

- Q1 Can you complete your task in $\overline{l}$ steps?
- Q2 Can you complete your task in $\overline{l}$ steps, if you lend m robots of type x before step l?
- Q3 Can you complete your task in $\overline{l}$ steps, if you borrow m robots of type x after step l?

These questions are more general than the ones in [4] due to consideration of heterogeneous robots; computing an answer for each question is still NP-hard. These questions can be further generalized, considering different combinations of types of robots that are lent or borrowed. We do not consider such generalizations, for computational efficiency purposes. Therefore,

- C3 Teams can borrow/lend robots of the same sort.

From teams' answers to these yes/no questions posed by the mediator, the following can be inferred:

Authorized licensed use limited to: Universita degli Studi di Genova. Downloaded on June 13,2024 at 12:11:59 UTC from IEEE Xplore. Restrictions apply.

- If there is a team that answers "no" to every question, then there is no overall plan of length $\bar{l}$ where every team completes its own tasks.
- Otherwise, we can identify sets $\textit{Lenders}_x \subset \textit{Lenders}$ of lender teams that can lend robots of type x and sets $\textit{Borrowers}_x \subset \textit{Borrowers}$ of borrower teams that needs to borrow robots of type x, where $x \leq n$ ($\textit{Lenders}, \textit{Borrowers} \subset \textit{Teams}$): If a team answers no to question Q1 and "yes" to question Q3 for some l, m and x, then it is a borrower for robot type x. If a team answers "yes" to question Q1 and "yes" to question Q2 for some l, m and x, then it is a lender for robot type x.
- For every lender (resp., borrower) team, from its answers to queries Q2 (resp., Q3), we can identify the earliest (resp., latest) time it can lend (resp., borrow) m robots of type x, $x \leq n$, in order to complete its tasks in $\bar{l}$ steps.

For every $\bar{l} \leq k$, these inferences can be used to decide whether lenders and borrowers can collaborate with each other, so that every team completes its task in $\bar{l}$ steps as follows.

First, let us identify the earliest lend times and latest borrow times by a collection of partial functions:

$$\textit{Lend\_earliest}_{m,x} : \textit{Lenders}_x \mapsto \{0, \ldots, \bar{l}\}$$
$$\textit{Borrow\_latest}_{m,x} : \textit{Borrowers}_x \mapsto \{0, \ldots, \bar{l}\}$$

Usually transferring robots from one team to another team takes some time, not only due to transportation but also due to calibration of the robots for a different workspace. Let us define such a delay time by a function:

$$\textit{Delay} : \textit{Lenders} \times \textit{Borrowers} \times \{1, \ldots, n\} \mapsto \{0, \ldots, \bar{l}\}.$$

Next, let us define when a set of lender teams can collaborate with a set of borrower teams.

Definition 1: An $n\overline{m}\bar{l}$*-collaboration* between *Lenders* and *Borrowers* with at most $\overline{m} = \max\{\overline{m}_x\}$ robot transfers, with n types of robots, and within at most $\bar{l}$ steps, relative to *Delay*, is a partial function

$$f : \textit{Lenders} \times \textit{Borrowers} \times \{1, \ldots, n\} \mapsto \{0, \ldots, \bar{l}\} \times \{0, \ldots, \overline{m}\}$$

(where $f(i, j, x) = (l, u)$ denotes that team i lends u robots of type x to team j at time step l) such that the following hold:

(a) For every borrower team $j \in \textit{Borrowers}_x$, there are some lender teams $i_1, \ldots, i_s \in \textit{Lenders}_x$, $x \leq n$, where the following two conditions hold:
- $f(i_1, j, x) = (l_1, u_1), \ldots, f(i_s, j, x) = (l_s, u_s)$ for some time steps $l_1, \ldots, l_s \leq \bar{l}$, some positive integers $u_1, \ldots, u_s \leq \overline{m}_x$, and some type x,
- $\textit{Delay}(i_1, j, x) = t_1, \ldots, \textit{Delay}(i_s, j, x) = t_s$ for some time steps $t_1, \ldots, t_s \leq \bar{l}$;

and there is a positive integer $m \leq \overline{m}_x$ such that

$$\max\{l_1 + t_1, \ldots, l_s + t_s\} \leq \textit{Borrow\_latest}_{m,x}(j)$$
$$m \leq \textstyle\sum_{k=1}^{s} u_k.$$

(b) For every lender team $i \in \textit{Lenders}_x$, for all borrower teams $j_1, \ldots, j_s \in \textit{Borrowers}_x$, $x \leq n$, such that $f(i, j_1, x) = (l_1, u_1), \ldots, f(i, j_s, x) = (l_s, u_s)$ for some time steps $l_1, \ldots, l_s \leq \bar{l}$, some positive integers $u_1, \ldots, u_s \leq \overline{m}_x$, and some type x, and there is a positive integer $m \leq \overline{m}_x$ such that

$$\textit{Lend\_earliest}_{m,x}(i) \leq \min\{l_1, \ldots, l_s\}$$
$$m \geq \textstyle\sum_{k=1}^{s} u_k.$$

Condition (a), which ensures that a borrower team does not borrow fewer robots than it needs, and Condition (b), which ensures that a lender team does not lend more robots than it can, together entail the existence of a lender team that can lend robots when a borrower team needs them.

Now we are ready to precisely describe the computational problem of finding a coordination of multiple teams of heterogeneous robots, to complete all the tasks as soon as possible in at most $\bar{l}$ steps where at most $\overline{m}$ robots can be relocated:

FINDCOLLABORATION_n
INPUT: For a set *Lenders* of lender teams, a set *Borrowers* of borrower teams, positive integers n, $\bar{l}$ and $\overline{m}_x$, $x \leq n$: a delay function *Delay* and a collection of functions $\textit{Lend\_earliest}_{m,x}$ and $\textit{Borrow\_latest}_{m,x}$ for every positive integer $m \leq \overline{m}_x$, $x \leq n$.
OUTPUT: A $n\overline{m}\bar{l}$*-collaboration* between *Lenders* and *Borrowers* with at most $\overline{m} = \max\{\overline{m}_x\}$ robot transfers, with at most n types of robots, and within at most $\bar{l}$ steps, relative to *Delay*.

As expected, this problem is also intractable but not harder than the one studied in [4] (assuming that P $\neq$ NP):

Proposition 1: The decision version of FINDCOLLABORATION_n (i.e., existence of a $n\overline{m}\bar{l}$-collaboration) is NP-complete.

Proof: We prove the membership as in the proof of Proposition 1 of [4]. We prove the hardness by a polynomial-time reduction from FINDCOLLABORATION, which is an NP-complete problem [4]: consider one type of robots in FINDCOLLABORATION_n. ■

Since the computational problem is intractable, ASP is suitable for solving it: Deciding whether a program in ASP has an answer set is NP-complete [44].

We model FINDCOLLABORATION_n in ASP as follows. The input is represented by a set of facts, using atoms of the forms $\textit{delay}(i, j, l)$, $\textit{lend\_earliest}(i, m, l, x)$, and $\textit{borrow\_latest}(j, m, l, x)$ where $1 \leq x \leq n$, $i \in \textit{Lenders}_x$, $j \in \textit{Borrowers}_x$, $m \leq \overline{m}$, $l \leq \bar{l}$.

Condition (a) is defined for each borrower j as follows:

$$\begin{aligned}
&\textit{condition_borrower}(j, x) \leftarrow \\
&\quad \textit{borrow_latest}(j, m, l, x), \\
&\quad \textit{sum}\langle\{u : f(i, j, l_1, u, x), \\
&\qquad\qquad i \in \textit{Lenders}_x, l_1 \leq \bar{l}, u \leq \overline{m}\}\rangle \geq m, \\
&\quad \textit{max}\langle\{l_1 + t : f(i, j, l_1, u, x), \textit{delay}(i, j, t), \\
&\qquad\qquad i \in \textit{Lenders}_x, l_1 \leq \bar{l}, u \leq \overline{m}\}\rangle \leq l
\end{aligned}$$

where $1 \leq x \leq n$, $j \in \textit{Borrowers}_x$, $l \leq \bar{l}$, $m \leq \overline{m}$. The second line of the rule above describes that team j needs m robots

Authorized licensed use limited to: Universita degli Studi di Genova. Downloaded on June 13,2024 at 12:11:59 UTC from IEEE Xplore. Restrictions apply.

of type x by step l. The third and fourth lines express that the number of robots lent to the borrower team j is at least m; the last two lines express the latest time step l that team j borrows a robot of type x. Similarly, we define condition (b).

We define an $\overline{nml}$-collaboration f, by atoms of the form $f(i,j,l,u,x)$ (describing $f(i,j,x)=(l,u)$), by first "generating" partial functions f:

$$\{f(i,j,l,u,x) : l \leq \bar{l}, u \leq \overline{m}\}1 \leftarrow \\ (1 \leq x \leq n, i \in \textit{Lenders}_x, j \in \textit{Borrowers}_x)$$

and then ensuring that the borrowers can borrow exactly one type of robot and that the lenders can lend at most one type of robots:

$$\leftarrow \textit{not } 1\{fB(j,x) : 1 \leq x \leq n\}1 \qquad (j \in \textit{Borrowers}) \\ \leftarrow 2\{fL(i,x) : 1 \leq x \leq n\} \qquad (i \in \textit{Lenders})$$

where $fB(j,x)$ and $fL(i,x)$ are projections of f onto j,x and i,x. Finally, we "eliminate" the partial functions that do not satisfy conditions (a) and (b) of Def. 1:

$$\leftarrow \textit{not condition\_borrower}(j,x), fB(j,x) \\ (j \in \textit{Borrowers}_x, 1 \leq x \leq n) \\ \leftarrow \textit{not condition\_lender}(i,x) \ \ (i \in \textit{Lenders}_x, 1 \leq x \leq n).$$

With the ASP formulation above, an ASP solver can find an $\overline{nml}$-collaboration.

VI. CASE STUDY: A COGNITIVE TOY FACTORY

We consider a toy factory with two teams of robots, where each team is located in a separate workspace collectively working toward completion of an assigned task. In particular, Team 1 manufactures nutcracker toy soldiers through the sequential stages of cutting, carving and assembling, while Team 2 processes them by going through stages of painting in black, painting in color, and stamping. Each workspace is depicted as a grid, as shown in Figure 1, contains an assembly line along the north wall to move the toys and a pit stop area where the worker robots can change their end-effectors. Each workspace also includes static obstacles.

The teams are heterogeneous, as each team is composed of three types robots with different capabilities. Worker robots operate on toys, they can configure themselves for different stages of processes, and they can be exchanged between teams; charger robots maintain the batteries of workers and monitor team's plan, and cannot be exchanged between teams. Worker robots are further categorized into two based on their liquid resistance. In particular, wet (liquid resistant) robots can perform every stage of the processes involved in manufacturing and painting of the toys, while dry (non-liquid resistant) robots cannot be employed in painting and cutting stages, since there processes involve liquids. All robots are holonomic and can move from any grid to another one following straight paths.

Note that this cognitive factory is different from the cognitive painting factory presented in [2], not only due to different robotic tasks but also due to heterogeneous worker robots and obstacles within workspace.

The teams act as autonomous cognitive agents; therefore, each team finds its own hybrid plan to complete its own designated task, as described in Section IV. We use the ASP solver CLASP to compute feasible local plans with minimum makespans. We consider three forms of optimization to further improve these local plans: (i) To minimize the total number of robotic actions, we define cost of each action as 1; (ii) To ensure that actions in a team are executed as early as possible, postponing idle time of robots to the end of plan execution, we define cost of each action as the step size t; (iii) To minimize fuel consumption for robots, we define costs of move actions as the distance traversed, while we keep the cost of all other actions as 1.

In this cognitive toy factory, teams can help each other: at any step, a team can lend several of its worker robots through their pit stop such that after a transportation delay the worker robots show up in the pit stop of a borrowing team. Following the methodology detailed in Section V, given the initial state of each workspace and the designated tasks for each team (e.g., how many toys to process), an optimal global plan (with minimum makespan) is computed for all teams to complete their tasks.

We have tested this cognitive factory with dynamic simulations (with all three different optimizations of local feasible plans) using OPENRAVE [45], and with an augmented reality physical implementation utilizing KuKa youBots and Lego NXT robots controlled over Robot Operating System (ROS). A snapshot of the physical implementation is shown in Figure 1. Videos of the demonstrations are available at `http://cogrobo.sabanciuniv.edu/?p=748`.

The physical implementation is valuable in that, it demonstrates (i) feasibility of the computed plans for execution under closed-loop control with real robots, (ii) concurrent actions of multiple robots to collaboratively achieve common goals within a team, and (iii) collaboration between teams via properly timed robot exchanges. The dynamic simulations are valuable in that, they show the effects of further optimizations over local plans with minimum makespans.

VII. EXPERIMENTAL EVALUATION

We have investigated the scalability of the proposed planning approach using the Cognitive Toy Factory domain described in the previous section. We have performed experiments on a Linux server with 16 Intel E5-2665 CPU cores (32 threads) with 2.4GHz and 64GB memory (note that our experiments never use more than 300MB). The ASP solver Clasp version 2.1.3 (with Gringo version 3.0.5) is used for answering queries.

First of all, it is important to observe that, since each team is equipped with its own computation unit, queries for teams can be trivially parallelized as depicted in Figure 2. Thanks to this parallelization, the total time required to find a coordination for an optimal global plan through a mediator, consists of the time it takes for the mediator to compute a coordination plus the time it takes for the slowest team to answer all queries asked by the mediator: $T_{total} = T_{coord} + T_{query}$. Furthermore, each team can additionally

Authorized licensed use limited to: Universita degli Studi di Genova. Downloaded on June 13,2024 at 12:11:59 UTC from IEEE Xplore. Restrictions apply.

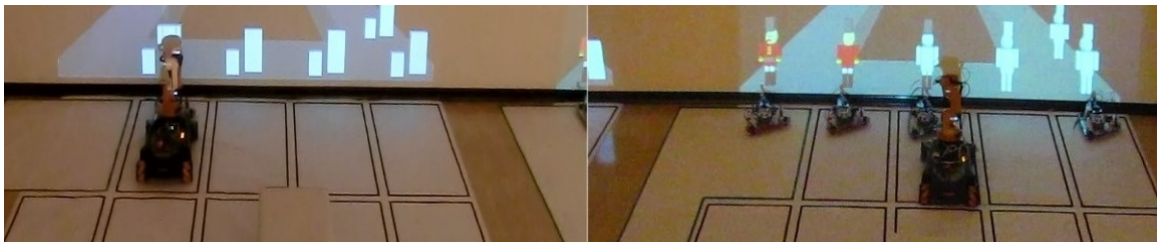

Fig. 1: Snapshot during the execution of optimal global plan by two teams utilizing Kuka youBots and Lego NXT robots. Videos of the physical implementation and dynamic simulations are available at: `http://cogrobo.sabanciuniv.edu/?p=748`

TABLE I: Experimental results for T_{query}

Number of robots	Total grid cells	Plan length	Total queries	Time to answer all queries (secs)
2	15	34	151	5.3
3	15	34	149	12.8
4	15	34	97	13.9
6	15	34	128	15.3
3	24	29	67	14.0
6	24	29	54	16.8
9	24	29	50	42.0

take advantage of multiple core/threads that may exists in its computation unit, to process multiple queries simultaneously. Therefore, we can analyze the scalability of the proposed approach by investigating T_{coord} and T_{query}, respectively.

To study the scalability of T_{query}, we have generated two sets of instances that vary over two different sizes of workspaces (with $5 \times 3 = 15$ and $8 \times 3 = 24$ grid cells), with the maximum makespan of $k = 34, 29$ respectively. The team sizes in these instances are kept reasonable (2–9 robots per workspace) considering real manufacturing processes, since every work cell in a real factory typically is of modest size with 3–12 operators. Since the experimental evaluation of different sorts of hybrid reasoning is extensively studied in a companion paper [6], query answering in this setting does not involve hybrid reasoning. Table I shows the results of query answering, averaged over three runs. We can observe from this table that the total time for a team of 2 robots to answer all 151 queries (essentially planning problems) is 5.3 seconds, under multi-threading. The computation time increases slightly as the number of robots per team and the size of the workspace increase.

To study the scalability of T_{coord}, we have generated two sets of instances, one with homogeneous worker robots ($n = 1$) and one with heterogeneous worker robots ($n = 2, 4$), that vary over the number of teams (ranging between 2–16) and the maximum number of robots that can be transferred ($\bar{m} = 2, 4$). Table II shows the results of coordination; each

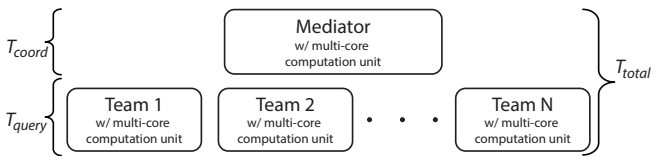

Fig. 2: Two sorts of parallelization are utilized: i) mediator asks sets of queries to teams in parallel and ii) each team takes advantage of multiple cores in its computation unit to process multiple queries simultaneously.

reported CPU time is the average for at least $\bar{m} \times n$ instances. We can observe from this figure that the computation time increases slightly as the factory involves more heterogeneous robots and more number of teams. These results can be observed more clearly, as the number of transferred robots also increases. When the factory involves 16 teams with heterogeneous robots, and the number of robots lent/borrowed between any two teams is at most 4, the computation time is still less than 15 seconds. Intuitively, involving more heterogeneous robots (resp. requiring more robot transfers between teams) makes the coordination problem harder since different capabilities of the robots (resp. more combinations of robot transfers) have to be considered while searching for a coordination.

TABLE II: CPU time in seconds for T_{coord}

number of teams	$\bar{m}$ = 2		$\bar{m}$ = 4	
	n = 1	$n > 1$	n = 1	$n > 1$
2	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	0.005
4	$< 10^{-6}$	0.014	0.023	0.245
8	0.005	0.061	2.92	3.61
16	0.080	0.432	6.63	14.9

Both results of experiments, for T_{query} and T_{coord}, are quite promising from the point of view of scalability, considering that both query answering and coordination are essentially NP-hard (Section V). Indeed, these results suggest that, for a cognitive factory with 16 workspaces each with 9 robots (so the factory involves 144 robots), and with upto 32 robot transfers between teams, the total time to find a coordination for an optimal global plan with minimum makespan (less than 29) would be around a minute ($T_{total} = 42 + 15 = 57$ seconds) using a desktop workstation.

VIII. DISCUSSION

We have proposed a hybrid reasoning method for finding local feasible plans with minimum makespans, for a team of heterogeneous robots trying to compete their tasks in a factory workspace as soon as possible. This method considers various capabilities of heterogeneous robots, embeds feasibility checks into task planning, and further optimizes these plans by minimizing total plan cost with respect to a given action cost.

We have also introduced a semi-distributed approach for finding a coordination of multiple teams of heterogeneous robots to help each other in a cognitive factory, to be able to complete all their tasks as early as possible. This method considers hybrid reasoning to compute feasible global plans, as well as transfers of heterogeneous robots between teams.

Authorized licensed use limited to: Universita degli Studi di Genova. Downloaded on June 13,2024 at 12:11:59 UTC from IEEE Xplore. Restrictions apply.

We have applied these methods to a cognitive toy factory using computational methods of answer set programming, and illustrated these applications by dynamic simulations and a physical implementation. Furthermore, we have showed the scalability of these methods by experimental evaluations in this domain.

During these studies, we have found the flexibility of (i) performing different forms of optimizations to improve plans, and (ii) combining different methods of integration of high-level reasoning with low-level feasibility checks, beneficial in the dynamic environment of a cognitive factory. We have also observed the computational benefits of our semi-distributed approach to find an optimal global plan: the computational effort is divided among the teams, and thus can be parallelized.

REFERENCES

[1] M. Zaeh, M. Beetz, K. Shea, G. Reinhart, K. Bender, C. Lau, M. Ostgathe, W. Vogl, M. Wiesbeck, M. Engelhard, C. Ertelt, T. Rühr, M. Friedrich, and S. Herle, "The cognitive factory," in *Changeable and Reconf. Manufacturing Systems*, 2009, pp. 355–371.

[2] E. Erdem, K. Haspalamutgil, V. Patoglu, and T. Uras, "Causality-based planning and diagnostic reasoning for cognitive factories," in *Proc. of IEEE Int. Conf. Emerging Tech. & Factory Automation (ETFA)*, 2012.

[3] K. Erol, D. S. Nau, and V. S. Subrahmanian, "Complexity, decidability and undecidability results for domain-independent planning," *Artif. Intell.*, vol. 76, no. 1–2, pp. 75–88, 1995.

[4] E. Erdem, V. Patoglu, Z. G. Saribatur, P. Schüller, and T. Uras, "Finding optimal plans for multiple teams of robots through a mediator: A logic-based approach," *Theory and Practice of Logic Programming*, vol. 13, no. 4–5, pp. 831–846, 2013.

[5] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *Proc. of ICRA*, 2011.

[6] E. Erdem, V. Patoglu, and P. Schüller, "A systematic analysis of levels of integration between high-level task planning and low-level feasibility checks," in *Proc. of RCRA*, 2014.

[7] G. Brewka, T. Eiter, and M. Truszczynski, "Answer set programming at a glance," *Commun. ACM*, vol. 54, no. 12, pp. 92–103, 2011.

[8] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "clasp: A conflict-driven answer set solver," in *Proc. of LPNMR*, 2007.

[9] J. McCarthy and P. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," in *Machine Intell.*, B. Meltzer and D. Michie, Eds. Edinburgh Univ. Press, 1969, vol. 4, pp. 463–502.

[10] J. McCarthy, "Circumscription—a form of non-monotonic reasoning," *Artif. Intell.*, vol. 13, pp. 27–39,171–172, 1980.

[11] J. Finger, "Exploiting constraints in design synthesis," Ph.D. dissertation, Stanford University, 1986.

[12] B. J. C. M. M. de Weerdt, "Introduction to planning in multiagent systems," *Multiagent and Grid Systems*, vol. 5, pp. 345–355, 2009.

[13] Y. Shoham and M. Tennenholtz, "On social laws for artificial agent societies:off-line design," *Artif. Intell.*, vol. 73, pp. 231–252, 1995.

[14] A. ter Mors, J. Valk, and C. Witteveen, "Coordinating autonomous planners," in *Proc. of IC-AI*, 2004, pp. 795–801.

[15] Q. Yang, D. S. Nau, and J. Hendler, "Merging separately generated plans with restricted interactions," *Comput. Intell.*, vol. 8, pp. 648–676, 1992.

[16] D. Foulser, M. Li, and Q. Yang, "Theory and algorithms for plan merging," *Artif. Intell.*, vol. 57, pp. 143–182, 1992.

[17] C. Stuart, "An implementation of a multi-agent plan synchronizer," in *Proc. of IJCAI*, 1985, pp. 1031–1033.

[18] M. P. Georgeff, "Communication and interaction in multi-agent planning," in *Proc. of DAI*, 1988, pp. 200–204.

[19] K. Decker and V. Lesser, "Designing a family of coordination algorithms," in *Proc. of DAI*, 1994, pp. 65–84.

[20] R. Alami, F. Ingrand, and S. Qutub, "A scheme for coordinating multi-robots planning activities and plans execution," in *Proc. of ECAI*, 1998.

[21] H. Ehtamo, R. P. Hamalainen, P. Heiskanen, J. Teich, M. Verkama, and S. Zionts, "Generating pareto solutions in a two-party setting: Constraint proposal methods," *Management Science*, vol. 45, no. 12, pp. 1697–1709, 1999.

[22] W. Tan and B. Khoshnevis, "Integration of process planning and scheduling – a review," *J. Intell. Manuf.*, vol. 11, no. 1, pp. 51–63, 2000.

[23] J. Hooker, "A hybrid method for the planning and scheduling," *Constraints*, vol. 10, no. 4, pp. 385–401, 2005.

[24] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithm design for multi-robot task assignment with deadlines for tasks," in *Proc. of ICRA*, 2013.

[25] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of multi-robot teams with temporospatial constraints," in *Proc. of RSS*, 2013.

[26] K. P. Sycara, S. P. Roth, N. M. Sadeh, and M. S. Fox, "Resource allocation in distributed factory scheduling," *IEEE Expert*, vol. 6, no. 1, pp. 29–40, 1991.

[27] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. A. Padget, S. Phelps, J. A. Rodríguez-Aguilar, and P. Sousa, "Issues in multiagent resource allocation," *Informatica*, vol. 30, no. 1, pp. 3–31, 2006.

[28] S.-H. Lin, "Coordinating time-constrained multi-agent resource sharing with fault detection," in *Proc. of IEEM*, 2011, pp. 1000–1004.

[29] R. Nair, M. Tambe, and S. Marsella, "Team formation for reformation in multiagent domains like robocuprescue," in *Proc. of RoboCup*, 2002, pp. 150–161.

[30] M. E. Gaston and M. desJardins, "The effect of network structure on dynamic team formation in multi-agent systems," *Comput. Intell.*, vol. 24, no. 2, pp. 122–157, 2008.

[31] S. LaValle and S. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[32] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Autonomous Robots*, vol. 32, no. 4, pp. 385–403, 2012.

[33] M. Turpin, N. Michael, and V. Kumar, "Concurrent assignment and planning of trajectories for large teams of interchangeable robots," in *Proc. of ICRA*, 2013, pp. 842–848.

[34] M. Gelfond and V. Lifschitz, "Classical negation in logic programs and disjunctive databases," *New Generation Computing*, vol. 9, pp. 365–385, 1991.

[35] M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, and M. Barry, "An A-Prolog decision support system for the space shuttle," in *Proc. of PADL*, 2001, pp. 169–183.

[36] J. Tiihonen, T. Soininen, and R. Sulonen, "A practical tool for mass-customising configurable products," in *Proc. of the International Conference on Engineering Design*, 2003, pp. 1290–1299.

[37] F. Ricca, G. Grasso, M. Alviano, M. Manna, V. Lio, S. Iiritano, and N. Leone, "Team-building with answer set programming in the Gioia-Tauro seaport," *Theory and Practice of Logic Programming*, vol. 12, no. 3, pp. 361–381, 2012.

[38] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits, "A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming," in *Proc. of IJCAI*, 2005, pp. 90–96.

[39] J. Kuffner Jr and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of ICRA*, 2000, pp. 995–1001.

[40] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. & Autom. Mag.*, vol. 19, no. 4, pp. 72–82, 2012.

[41] D. J. Duff, E. Erdem, and V. Patoglu, "Integration of 3D object recognition and planning for robotic manipulation: A preliminary report," in *Proc. ICLP 2013 Workshop on Knowledge Representation and Reasoning in Robotics*, 2013.

[42] R. Trejo, J. Galloway, C. Sachar, V. Kreinovich, C. Baral, and L.-C. Tuan, "From planning to searching for the shortest plan: An optimal transition," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 9, no. 6, pp. 827–837, 2001.

[43] E. Erdem, E. Aker, and V. Patoglu, "Answer set programming for collaborative housekeeping robotics: Representation, reasoning, and execution," *Intell. Service Robotics*, vol. 5, no. 4, pp. 275–291, 2012.

[44] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, "Complexity and expressive power of logic programming," *ACM Computing Surveys*, vol. 33, no. 3, pp. 374–425, 2001.

[45] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, CMU, August 2010.

Authorized licensed use limited to: Universita degli Studi di Genova. Downloaded on June 13,2024 at 12:11:59 UTC from IEEE Xplore. Restrictions apply.