Hybrid Reasoning for Teams of Heterogeneous Robots: Finding an Optimal Feasible Global Plan *

Zeynep G. Saribatur, Esra Erdem, Volkan Patoglu
Faculty of Engineering and Natural Sciences
Sabancı University, İstanbul, Turkey

Abstract

We address two key challenges of domains including teams of heterogeneous robots, hybrid reasoning for each team and finding an optimal global plan for multiple teams, by utilizing state-of-the-art automated reasoners. For hybrid reasoning, we propose (i) representing each team's workspace taking into account capabilities of heterogeneous robots, (ii) combining different methods of integration to embed external computations (e.g., collision checks) into high-level representation and reasoning, (iii) further optimizing local feasible shortest plans by minimizing the total cost of robotic actions, where costs of actions can be defined in various ways. To find a shortest global plan, we propose a semi-distributed approach based on our earlier studies: 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 automated reasoners. As a case study, we have formally represented a cognitive toy factory and showed an application of our hybrid reasoning and coordination approach on this domain. We have performed dynamic simulations and physical implementation of the cognitive toy factory utilizing KuKa youBots and Lego NXT robots.

1 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 (Aker et al. 2011; Aker, Patoglu, and Erdem 2012; Erdem, Aker, and Patoglu 2012) to cognitive factories (Erdem et al. 2012; 2013b). 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 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 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 (Erol, Nau, and Subrahmanian 1995; Erdem et al. 2013b).

For hybrid reasoning, we emphasize several core characteristics of these domains, such as existence of heterogeneous robot 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* plans for each team. The proposed method is based on our earlier works on hybrid planning (Caldiran et al. 2009; Erdem et al. 2011; Havur et al. 2013; Schueller, Patoglu, and Erdem 2013b) 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 in hybrid planning by focusing on heterogeneity of robots, considering existence of static/dynamic obstacles in the domain, utilizing a combination of integration methods for performing feasibility checks, and conducting various optimizations on plans. In particular, for performing feasibility checks, we utilize pre-computation approach to embed information about static obstacles perceived by a Kinect RGBD camera into the domain description, while we rely on guided replanning to account for possible robot-robot collisions (Haspalamutgil et al. 2010; Schueller, Patoglu, and Erdem 2013b; 2013a). In addition to finding a shortest plan, we further optimize the total cost of this plan by considering several objectives: 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 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 major concern in micro manufacturing plants that specialize on prototyping pre-release products, while reduction of communication among teams may be preferable when the means of communication

*This work is supported by Sabancı University IRP Grant IACF09-00643, and TUBITAK Grant 111E116.

is lossy or costly. Furthermore, a semi-distributed approach is advantageous in that it reduces the size of the global domain into manageable pieces; hence, provides a solution methodology that can be parallelized to include a large number of robot teams.

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 (Erdem et al. 2013b). 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) 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 (Gebser et al. 2007). ASP can handle many challenges: concurrency, the frame problem (McCarthy and Hayes 1969), the qualification problem (McCarthy 1980), ramifications (Finger 1986), numeric-valued fluents (Lee and Lifschitz 2003; Erdem and Gabaldon 2005), 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 (collision-free) plans.

Using ASP, we show applicability of our hybrid reasoning and coordination approach through a case study of a cognitive toy factory. Cognitive factory concept (Beetz, Buss, and Wollherr 2007; Zaeh et al. 2009; 2012; Erdem et al. 2012; 2013b) 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.

Contributions of the paper can be summarized as follows: We have extended our earlier studied formal frameworks, as described in detail above. As a case study for these extensions, we have formally represented a cognitive toy factory and showed the applicability and effectiveness of our hybrid reasoning and coordination approach on this domain. We have performed dynamic simulations and physical implementation of the cognitive toy factory utilizing KuKa youBots and Lego NXT robots.

2 Answer Set Programming

Answer Set Programming (ASP) (Marek and Truszczyński 1999; Niemelä 1999; Lifschitz 2002; 2008; Brewka, Eiter, and Truszczynski 2011) is a form of knowledge representation and reasoning paradigm oriented towards solving combinatorial search problems as well as knowledge-intensive problems. The idea is to represent a problem as a "program" whose models (called "answer sets" (Gelfond and Lifschitz 1991; 1988)) correspond to the solutions. The answer sets for the given program can then be computed by special implemented systems called answer set solvers. ASP has a high-level representation language that allows recursive definitions, aggregates, weight constraints, optimization statements, and default negation. ASP also provides efficient solvers, such as CLASP (Gebser et al. 2007), which has recently won first places at various automated reasoning competitions, such as SAT competitions and ASP competitions.

Due to the continuous improvement of the ASP solvers and highly expressive representation language of ASP which is supported by a strong theoretical background that results from a years of intensive research, ASP has been applied fruitfully to a wide range of areas, including industrial applications (Nogueira et al. 2001; Tiihonen, Soininen, and Sulonen 2003; Ricca et al. 2012). ASP has also been used for various robotic applications, such as housekeeping robotics (Aker, Patoglu, and Erdem 2012; Erdem, Aker, and Patoglu 2012), cognitive factories (Erdem et al. 2013b), and multi-path finding (Erdem et al. 2013a).

In this study, we use ASP for high-level representation of action domains and combinatorial search problems, and an ASP solver as an automated reasoner. We consider ASP programs (i.e., nondisjunctive HEX programs (Eiter et al. 2005)) that consists of rules of the form:

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

where $m, n \geq 0$, *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 $\&g[y_1, \ldots, y_k](x_1, \ldots, y_l)$ 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. Therefore, external predicates 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 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) (Kuffner Jr and LaValle 2000) as in the OMPL (Sucan, Moll, and Kavraki 2012) 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

$$\#minimize\ [\ cost(r, c, t) : robot(r) = c\ ] \qquad (2)$$

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 $cost(r, c, t)$.

3 Hybrid Reasoning for a Team of Heterogeneous Robots

To find optimal feasible plans for a team of heterogeneous robots, we consider the expressive representation formalisms and state-of-the-art efficient solvers of ASP as follows.

3.1 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 by the following rules in ASP:

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

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

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

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

3.2 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 (Schueller, Patoglu, and Erdem 2013b; 2013a): (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 predicates, (ii) Interleaved computation (INT): external computations are directly embedded in an action domain description via external predicates 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 (Erdem et al. 2011; Aker, Patoglu, and Erdem 2012).

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:

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

where *&obstacleAt* is an external predicate whose value is determined by an object detection algorithm (Duff, Erdem, and Patoglu 2013) using Point Cloud Library over data obtained by a Kinect RGBD 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:

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

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:

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

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

$$\begin{aligned}\leftarrow\ & at(R1, A, t), at(R2, B, t),\\ & at(R1, B, t + 1), at(R2, A, t + 1)\end{aligned}$$

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.

3.3 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 utilize optimization statements of ASP (as described in Section 2) 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 (Trejo et al. 2001). Alternatively, we can minimize the time step to reach a goal:

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

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*:

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

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 (Erdem, Aker, and Patoglu 2012).

4 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 (Erdem et al. 2013b):

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 (Erdem et al. 2013b): 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 $\bar{l} \leq k$, $l \leq \bar{l}$ and $m \leq \overline{m}_x$, $x \leq n$:

Q1 Can you complete your task in $\bar{l}$ steps?

Q2 Can you complete your task in $\bar{l}$ steps, if you lend m robots of type x before step l?

Q3 Can you complete your task in $\bar{l}$ steps, if you borrow m robots of type x after step l?

These questions are more general than the ones in (Erdem et al. 2013b) 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:

- 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:

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

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*

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

(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 $j \in \mathit{Borrowers}_x$*, there are some lender teams* $i_1,\ldots,i_s \in \mathit{Lenders}_x$*,* $x \le n$*, 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* $l_1,\ldots,l_s \le \bar{l}$*, some positive integers* $u_1,\ldots,u_s \le \overline{m}_x$*, and some type* x*,*
- $\mathit{Delay}(i_1,j,x)=t_1,\ldots,\mathit{Delay}(i_s,j,x)=t_s$ *for some time steps* $t_1,\ldots,t_s \le \bar{l}$*;*

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

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

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

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

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 $\overline{m}_x$, $x \le n$: a delay function *Delay* and a collection of functions $\mathit{Lend\_earliest}_{m,x}$ and $\mathit{Borrow\_latest}_{m,x}$ for every positive integer $m \le \overline{m}_x$, $x \le n$.
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 (Erdem et al. 2013b):

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 (Erdem et al. 2013b). We prove the hardness by a polynomial-time reduction from FINDCOLLABORATION, which is an NP-complete problem (Erdem et al. 2013b): consider one type of robots in FINDCOLLABORATION_n. □

Since the computational problem is intractable, Answer Set Programming (ASP) is suitable for solving it: Deciding whether a program in ASP has an answer set is NP-complete (Dantsin et al. 2001).

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

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

$$\begin{aligned}&\mathit{condition\_borrower}(j,x) \leftarrow\\ &\quad \mathit{borrow\_latest}(j,m,l,x),\\ &\quad \mathit{sum}\langle\{u : f(i,j,l_1,u,x),\\ &\qquad\qquad i \in \mathit{Lenders}_x, l_1 \le \bar{l}, u \le \overline{m}\}\rangle \ge m,\\ &\quad \mathit{max}\langle\{l_1{+}t : f(i,j,l_1,u,x), \mathit{delay}(i,j,t),\\ &\qquad\qquad i \in \mathit{Lenders}_x, l_1 \le \bar{l}, u \le \overline{m}\}\rangle \le l\end{aligned}$$

where $1 \le x \le n$, $j \in \mathit{Borrowers}_x$, $l \le \bar{l}$, $m \le \overline{m}$. The second line of the rule above describes that team j needs m robots 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) as follows:

$$\begin{aligned}&\mathit{condition\_lender}(i,x) \leftarrow\\ &\quad \mathit{lend\_earliest}(i,m,l,x),\\ &\quad \mathit{sum}\langle\{u : f(i,j,l_1,u,x),\\ &\qquad\qquad j \in \mathit{Borrowers}_x, l_1 \le \bar{l}, u \le \overline{m}\}\rangle \le m\\ &\quad \mathit{min}\langle\{l_1 : f(i,j,l_1,u,x),\\ &\qquad\qquad j \in \mathit{Borrowers}_x, l_1 \le \bar{l}, u \le \overline{m}\}\rangle \ge l\end{aligned}$$

where $1 \le x \le n$, $i \in \mathit{Lenders}_x$, $l \le \bar{l}$, $m \le \overline{m}$.

We define an $n\overline{m}\bar{l}$-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:

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

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:

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

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:

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

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

5 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 some 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 following straight paths.

The teams act as autonomous cognitive agents; therefore, each team finds its own hybrid plan to complete its own designated task. For that, we have formalized each workspace in ASP and used the ASP solver CLASP to compute hybrid plans, as discussed Section 3. We have utilized hybrid reasoning, by combining two integration methods: PRE method to integrate perception into planning in order to detect the static obstacles, and GREPL method to integrate robot-robot collision checks into planning. In particular, the external predicate used by the PRE method is determined by an object detection algorithm (Duff, Erdem, and Patoglu 2013) using Point Cloud Library over data obtained by a Kinect RGBD camera, while robot-robot collisions are checked via simulations in ODE. We have implemented three forms of optimization to further improve local feasible shortest plans as described in Section 3: (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 4, 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 is computed for all teams to complete their tasks in a minimum number of steps, while simultaneously minimizing several other cost functions as discussed in Section 3.

We have tested this cognitive toy factory scenario with both dynamic simulations (with all three different optimizations of local feasible plans) using OPENRAVE (Diankov 2010), and with a physical implementation utilizing KuKa youBots and Lego NXT robots controlled over Robot Operating System (ROS). A snapshot of the physical simulation is shown in Figure 1, while a snapshot of the dynamic simulation is shown in Figure 2. Videos of these implementations are available at http://cogrobo.sabanciuniv.edu/?p=748.

6 Discussion

We have proposed the use of state-of-the-art automated reasoners for hybrid reasoning to find an optimal local feasible plan for a team of heterogeneous robots, and for finding an optimal global plan for multiple such teams. For that, we have shown how to take advantage of the expressive representation languages of these reasoners, for describing capabilities of heterogeneous robots, for formalizing different sorts of optimizations, and for embedding results of external computations (e.g., for perception or geometric reasoning) in action domain descriptions. Extending our earlier studies, we have also introduced a formal framework to compute optimal global plans taking into account transfer of heterogeneous robots between teams. We have applied these approaches to a cognitive toy factory using computational methods of answer set programming, and illustrated these applications by dynamic simulations and a physical implementation.

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

Aker, E.; Erdogan, A.; Erdem, E.; and Patoglu, V. 2011. Causal reasoning for planning and coordination of multiple housekeeping robots. In *Proc. of LPNMR*, 311–316.

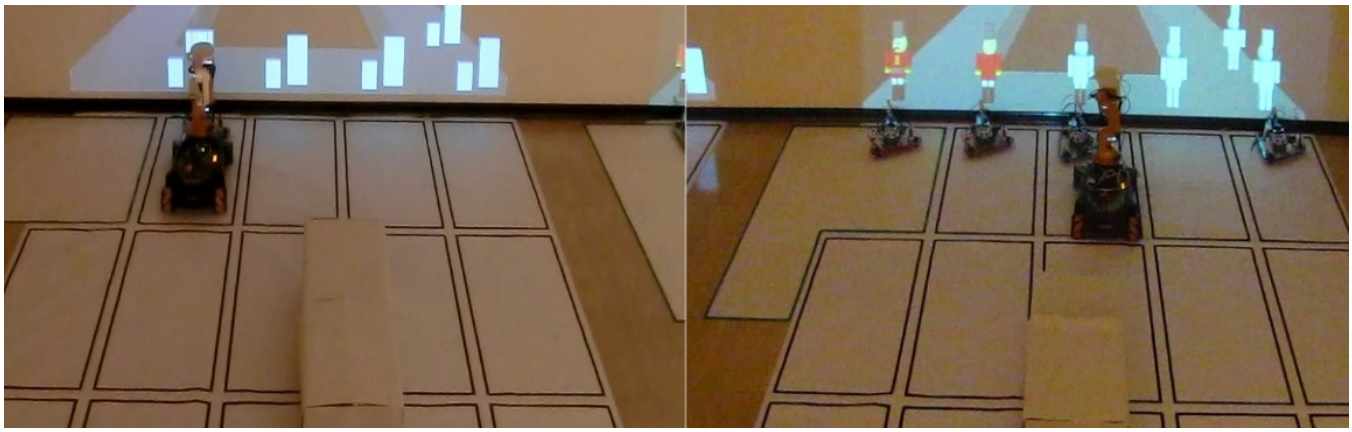

Figure 1: Snapshot during the execution of optimal global plan by two teams utilizing Kuka youBots and Lego NXT robots.

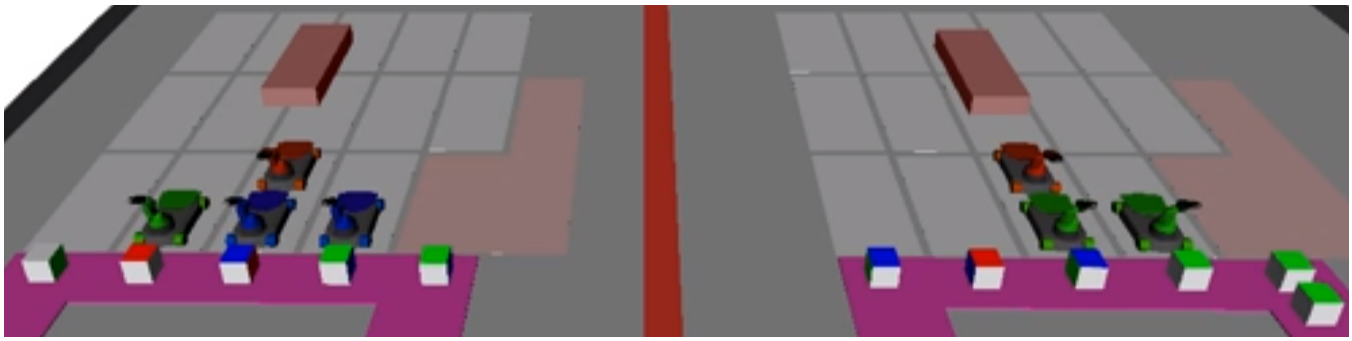

Figure 2: Snapshot during the dynamic simulation of optimal global plan by two teams utilizing Kuka youBots.

Aker, E.; Patoglu, V.; and Erdem, E. 2012. Answer set programming for reasoning with semantic knowledge in collaborative housekeeping robotics. In *Proc. of IFAC SYROCO*.

Beetz, M.; Buss, M.; and Wollherr, D. 2007. CTS - What is the role of artificial intelligence? In *Proc. of KI*, 19–42.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Commun. ACM* 54(12):92–103.

Caldiran, O.; Haspalamutgil, K.; Ok, A.; Palaz, C.; Erdem, E.; and Patoglu, V. 2009. Bridging the gap between high-level reasoning and low-level control. In *Proc. of LPNMR*.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys* 33(3):374–425.

Diankov, R. 2010. *Automated Construction of Robotic Manipulation Programs*. Ph.D. Dissertation, Carnegie Mellon University, Robotics Institute.

Duff, D. J.; Erdem, E.; and Patoglu, V. 2013. 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*.

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

Erdem, E.; Aker, E.; and Patoglu, V. 2012. Answer set programming for collaborative housekeeping robotics: Representation, reasoning, and execution. *Intelligent Service Robotics* 5(4):275–291.

Erdem, E., and Gabaldon, A. 2005. Cumulative effects of concurrent actions on numeric-valued fluents. In *Proc. of AAAI*, 627–632.

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

Erdem, E.; Haspalamutgil, K.; Patoglu, V.; and Uras, T. 2012. Causality-based planning and diagnostic reasoning for cognitive factories. In *Proc. of IEEE Int. Conf. Emerging Technologies and Factory Automation (ETFA)*.

Erdem, E.; Kisa, D. G.; Oztok, U.; and Schuller, P. 2013a. A general formal framework for pathfinding problems with multiple agents. In *Proc. 27th AAAI Conf. on Artificial Intelligence (AAAI)*.

Erdem, E.; Patoglu, V.; Saribatur, Z. G.; Schuller, P.; and Uras, T. 2013b. Finding optimal plans for multiple teams of robots through a mediator: A logic-based approach. *Theory and Practice of Logic Programming*.

Erol, K.; Nau, D. S.; and Subrahmanian, V. S. 1995. Complexity, decidability and undecidability results for domain-independent planning. *Artif. Intell.* 76(1–2):75–88.

Finger, J. 1986. *Exploiting Constraints in Design Synthesis*. Ph.D. Dissertation, Stanford University. PhD thesis.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T.

2007. clasp: A conflict-driven answer set solver. In *Proc. of LPNMR*, 260–265.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP*, 1070–1080. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Haspalamutgil, K.; Palaz, C.; Uras, T.; Erdem, E.; and Patoglu, V. 2010. A tight integration of task and motion planning in an execution monitoring framework. In *Proc. of BTAMP*.

Havur, G.; Haspalamutgil, K.; Palaz, C.; Erdem, E.; and Patoglu, V. 2013. A case study on the tower of hanoi challenge: Representation, reasoning and execution. In *Proc. of ICRA*.

Kuffner Jr, J., and LaValle, S. 2000. RRT-connect: An efficient approach to single-query path planning. In *International Conference on Robotics and Automation (ICRA)*, volume 2, 995–1001. IEEE.

Lee, J., and Lifschitz, V. 2003. Describing additive fluents in action language $\mathcal{C}$+. In *Proc. of IJCAI*.

Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence* 138:39–54.

Lifschitz, V. 2008. What is answer set programming? In *Proc. of AAAI*, 1594–1597. MIT Press.

Marek, V., and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*. Springer Verlag. 375–398.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence*, volume 4. Edinburgh: Edinburgh University Press. 463–502.

McCarthy, J. 1980. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence* 13:27–39,171–172.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25:241–273.

Nogueira, M.; Balduccini, M.; Gelfond, M.; Watson, R.; and Barry, M. 2001. An A-Prolog decision support system for the space shuttle. In *Proc. of PADL*, 169–183. Springer.

Ricca, F.; Grasso, G.; Alviano, M.; Manna, M.; Lio, V.; Iiritano, S.; and Leone, N. 2012. Team-building with answer set programming in the Gioia-Tauro seaport. *Theory and Practice of Logic Programming* 12.

Schueller, P.; Patoglu, V.; and Erdem, E. 2013a. Levels of integration between low-level reasoning and task planning. In *Proc. AAAI 2013 Workshop on Intelligent Robotic Systems*.

Schueller, P.; Patoglu, V.; and Erdem, E. 2013b. A systematic analysis of levels of integration between low-level reasoning and task planning. In *Proc. ICRA 2013 Workshop on Combining Task and Motion Planning*.

Sucan, I. A.; Moll, M.; and Kavraki, L. E. 2012. The open motion planning library. *Robotics & Automation Magazine, IEEE* 19(4):72–82.

Tiihonen, J.; Soininen, T.; and Sulonen, R. 2003. A practical tool for mass-customising configurable products. In *Proc. of the International Conference on Engineering Design*, 1290–1299.

Trejo, R.; Galloway, J.; Sachar, C.; Kreinovich, V.; Baral, C.; and Tuan, L.-C. 2001. From planning to searching for the shortest plan: An optimal transition. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 9(6):827–837.

Zaeh, M.; Beetz, M.; Shea, K.; Reinhart, G.; Bender, K.; Lau, C.; Ostgathe, M.; Vogl, W.; Wiesbeck, M.; Engelhard, M.; Ertelt, C.; Rhr, T.; Friedrich, M.; and Herle, S. 2009. The cognitive factory. In *Changeable and Reconf. Manufacturing Systems*. 355–371.

Zaeh, M.; Ostgathe, M.; Geiger, F.; and Reinhart, G. 2012. Adaptive job control in the cognitive factory. In ElMaraghy, H. A., ed., *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer Berlin Heidelberg. 10–17.