

Finding optimal plans for multiple teams of robots through a mediator: A logic-based approach

ESRA ERDEM¹, VOLKAN PATOGLU¹,
ZEYNEP G. SARIBATUR¹, PETER SCHÜLLER¹
and TANSEL URAS²

¹*Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey*

²*Department of Computer Science, University of Southern California, Los Angeles, USA*

submitted 10 April 2013; revised 23 June 2013; accepted 5 July 2013

Abstract

We study the problem of finding optimal plans for multiple teams of robots through a mediator, where each team is given a task to complete in its workspace on its own and where teams are allowed to transfer robots between each other, subject to the following constraints: 1) teams (and the mediator) do not know about each other's workspace or tasks (e.g., for privacy purposes); 2) every team can lend or borrow robots, but not both (e.g., transportation/calibration of robots between/for different workspaces is usually costly). We present a mathematical definition of this problem and analyze its computational complexity. We introduce a novel, logic-based method to solve this problem, utilizing action languages and answer set programming for representation, and the state-of-the-art ASP solvers for reasoning. We show the applicability and usefulness of our approach by experiments on various scenarios of responsive and energy-efficient cognitive factories.

KEYWORDS: answer set programming, decoupled planning, cognitive robotics

1 Introduction

As conventional manufacturing and assembly systems fall short of responding to constantly rising market demands for customized and variant-rich products in a cost effective manner within short delivery times, new approaches for automated fabrication of customized products become crucial for enhancing productivity, ensuring competitiveness and *economic sustainability* in the manufacturing sector. Along these lines, reconfigurable and flexible manufacturing systems have been deployed over the last decade. Cognitive factories (Beetz *et al.* 2007; Zaeh *et al.* 2009; Zaeh *et al.* 2012) are a further step in this direction aimed towards highly flexible and typically small to medium size manufacturing plants that can produce a very large variety of customized products even in low quantities. Rapidly responding to changing customer needs and customization requests, cognitive factories can

demonstrate the flexibility of human workshops, while maintaining cost-effectiveness of mass production systems.

Cognitive factories endow manufacturing system with high-level reasoning capabilities in the style of cognitive robotics, such that these systems become capable of planning their own actions. By utilizing sophisticated planning and decision-making algorithms, cognitive factories can efficiently allocate their resources for daily/weekly/monthly work load and ensure production of variant-rich products to guarantee pressing delivery deadlines.

One of the key challenges in cognitive factories is the coordination between multiple teams of robots to achieve overall shortest delivery time for a given manufacturing order. Minimizing the delivery lead time for a customized order not only leads to a more cost-effective process by reducing contribution of factory overhead per order, but also preserves energy resources and decreases negative environmental impacts by efficient use of facility infrastructure, such as HVAC (heating, ventilation, and air conditioning) and lighting.

With this motivation, we consider multiple teams of robots, where each team is given a feasible task to complete in its workspace on its own, and where teams are allowed to transfer robots between each other. The goal is to find an optimal overall plan for all teams so that all tasks can be completed as soon as possible, subject to the following constraints:

- 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.

Note that a lender can lend robots to more than one team; and a borrower can borrow robots from more than one team. Note also that each team may have robots that cannot be transferred; the above problem considers transferrable robots only and assumes that capabilities of transferrable robots are indifferent.

We introduce a novel method to find an optimal global plan for all teams, with at most k steps, subject to constraints C1 and C2, and the presence of a mediator who does not belong to any team and who does not know anything about teams' workspaces, tasks and goals. Our method consists of two phases: finding a coordination of the teams and then an optimal global plan.

In the first phase, for a nonnegative integer $\bar{l} \leq k$ denoting the length of a global plan: 1) The mediator asks yes/no questions to every team (in any order), to identify whether a team can complete its task in \bar{l} steps, while lending/borrowing how many robots to/from other teams and when. 2) Once answers to these questions are collected, the mediator tries to find a coordination of the teams (i.e., which team should lend how many robots to which other team, and when), subject to the constraints C1 and C2 above. The optimal value for \bar{l} can be found by a linear search between 1 and k .

In the second phase, after some coordination of teams is found for an optimal value of \bar{l} , the mediator informs each team how many robots it is expected to lend to (or borrow from) which other team and when. Taking this information into account, each team computes an optimal local plan (whose length is less than or equal to \bar{l}) to complete its task. An optimal global plan for all teams is the union of all optimal local plans.

Note that the mediator cannot find a global plan on its own since it does not know about teams' workspace, tasks, plans, actions, goals, etc.. In fact, a centralized approach to compute a global plan is in most cases not scalable due to large domain description that formalizes all workspaces and teams. Also note that teams do not communicate with each other. Otherwise, the number of queries (and the number of rounds of exchanging messages) would increase substantially, leading to a more time-consuming process to find an optimal global plan.

Both phases involve solving computational problems that are intractable, since finding plans of length \bar{l} possibly with temporal constraints is NP-complete (Turner 2002), and answering each query in the first phase is a planning problem with temporal constraints, and thus NP-complete (Turner 2002). We prove that finding a coordination of the teams for a global plan with at most \bar{l} steps is also NP-complete (see Proposition 1 in Section 3). In the first phase, each team answers queries that are relevant to its workspace, task, goals only, and independently of other teams; therefore, queries can be answered in parallel. In the second phase, each team computes an optimal local plan on its own; therefore, optimal local plans can be computed in parallel as well.

We propose to solve the planning problems with temporal constraints (and thus answering queries posed by the mediator), and the coordination problem using answer set programming (ASP) (Marek and Truszczyński 1999; Niemelä 1999; Lifschitz 2002; Lifschitz 2008; Brewka *et al.* 2011). To solve planning problems, first we represent action domains (i.e., workspaces) and planning problems (i.e., queries) in the input language of CCALC (McCain and Turner 1997), which allows representation of dynamic domains in a subset of the expressive action description language $\mathcal{C}+$ (Giunchiglia *et al.* 2004), and allows teams to solve planning problems with temporal constraints in a variation of the action query language \mathcal{Q} (Gelfond and Lifschitz 1998). Then CCALC's input is transformed into ASP using the tool CPLUS2ASP (Casolary and Lee 2011). After that, we can use state-of-the-art ASP solvers, like CLASP (Gebser *et al.* 2007), to compute plans. To solve the coordination problem using ASP solvers, we formulate the coordination problem in the representation language of ASP.

We show the usefulness and applicability of our approach by experiments on various scenarios of responsive energy-efficient cognitive factories.

2 Automating reasoning for a team of robots

Our goal is to find an optimal global plan with at most k steps, where at most \bar{m} robots can be transferred between teams. To find a coordination of teams for an

optimal global plan, 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 $m \leq \bar{m}$:

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 before step l ?

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

Once such a coordination is found from the answers of these queries, the mediator informs each team how many robots it is expected to lend to (or borrow from) which other team and when. Then each team computes an optimal local plan to complete its own task, taking into account the relevant information of transfer of robots from/to it. Therefore, each team of robots performs two kinds of reasoning tasks: answering queries of form Q1, Q2 and Q3, and finding optimal plans with complex goals to complete its tasks. Although optimal plans can be found by some existing classical planners, queries mentioned above cannot be answered by them directly. Also in many application scenarios for cognitive factories, actions of robots are concurrent (e.g., several robots working on different parts of an order at the same time) and there are ramifications and/or delayed effects of actions (e.g., after painting, a box gets dried after a while).

Due to the representation challenges mentioned above (about concurrent actions, ramifications, etc.), we represent workspaces (as action domains) and queries (as planning problems) in the input language of CCALC (McCain and Turner 1997), which allows representation of dynamic domains in a subset of the expressive action description language $\mathcal{C}+$ (Giunchiglia et al. 2004), and allows teams to answer various sorts of queries in an elaboration tolerant way (without having to modify the domain description) in a variation of the action query language \mathcal{Q} (Gelfond and Lifschitz 1998). Also that the language $\mathcal{C}+$ has been used in various sophisticated real-world robotic applications (Erdem and Patoglu 2012), where discrete high-level reasoning is integrated tightly with continuous geometric reasoning and low-level controls (Erdem et al. 2011; Aker et al. 2012; Havur et al. 2013), is advantageous for multi-robot cognitive factory applications that have motivated our studies in this paper. We refer the reader to (Giunchiglia et al. 2004) for the syntax and semantics of $\mathcal{C}+$, and some examples.

In the action query language of CCALC, an *atomic query* is one of the two forms, F **holds at** t or A **occurs at** t , where F is a fluent formula, A is an action formula, and t is a time step. A *query* is a propositional combination of atomic queries.

Suppose that F and G are fluent formulas denoting an initial state and goal conditions respectively. We can express the question “can you complete the task specified by the initial state F and the goal conditions G in k steps?”, with a query of the form

$$F \text{ holds at } 0 \wedge G \text{ holds at } k.$$

Note that this query describes the problem of finding a plan of length k .

Suppose that the action formula $giveRobot(w)$ describes that the team lends the robot w . We can express the question “can you complete your task specified by the initial state F and the goal conditions G in k steps, while also lending m robots

before step k' ?", with a query of the form

$$F \text{ holds at } 0 \wedge G \text{ holds at } k \wedge \exists T, W_1, \dots, W_m : \\ T < k' \wedge W_1 < W_2 < \dots < W_m \wedge \bigwedge_{i=1}^m \text{giveRobot}(W_i) \text{ occurs at } T.$$

Given an action description and a query in the input language of CCALC, we can use either state-of-the-art parallel SAT solvers (like MANYSAT (Hamadi *et al.* 2009)) or state-of-the-art ASP solvers (like CLASP (Gebser *et al.* 2007)) to find an answer to the query. After some experimental evaluations comparing these two approaches (summarized in Appendix A), we have decided to use the ASP solver CLASP (with the grounder GRINGO (Gebser *et al.* 2011)) to find answers to queries and planning problems. CCALC's input can be transformed into an ASP program in the input language of CLASP, using the tool CPLUS2ASP (Casolary and Lee 2011). If CLASP finds an answer set (Gelfond and Lifschitz 1998) for the ASP program then the query is answered affirmatively; otherwise, the query is answered negatively.

Note that since each team answers queries that are relevant to its workspace, task, goals only, and independently of other teams, queries can be answered by the teams in parallel.

3 Coordination of teams

From teams' answers to yes/no questions (of the forms Q1–Q3) 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 a set *Lenders* of lender teams and a set *Borrowers* of borrower teams ($Lenders, Borrowers \subset Teams$): If a team answers “no” to question Q1, and “yes” to question Q3 for some l and m , then it is a borrower. If a team answers “yes” to question Q1 and answers “yes” to question Q2 for some l and m , then it is a lender. Otherwise, it is neither a lender nor a borrower.
- 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) robots, 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.

To precisely define this problem, let us introduce some notation. For every lender team $i \in Lenders$, positive integer $m \leq \bar{m}$ and nonnegative integer $l \leq \bar{l}$, we denote by atoms of the form $lend(i, m, l)$ that the lender team i can lend m robots before time step l . Similarly, for every borrower team $i \in Borrowers$, we denote by atoms of the form $borrow(i, m, l)$ that the borrower team i needs m robots before time step l .

To identify the earliest lend times and latest borrow times, we introduce a collection of partial functions for lenders and borrowers:

$$\begin{aligned} \text{Lend_earliest}_m &: \text{Lenders} \mapsto \{0, \dots, \bar{l}\} \\ \text{Borrow_latest}_m &: \text{Borrowers} \mapsto \{0, \dots, \bar{l}\} \end{aligned}$$

where Lend_earliest_m returns the earliest step that a lender can lend m robots and Borrow_latest_m returns the latest step that a borrower needs to borrow m robots:

$$\begin{aligned} \text{Lend_earliest}_m(i) &= \arg \min_l \{ \text{lend}(i, m, l) = 1 \} \\ \text{Borrow_latest}_m(j) &= \arg \max_l \{ \text{borrow}(j, m, l) = 1 \}. \end{aligned}$$

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. Such a delay time is defined by a function:

$$\text{Delay} : \text{Lenders} \times \text{Borrowers} \mapsto \{0, \dots, \bar{l}\}.$$

Now we can define when a set of lender teams can collaborate with a set of borrower teams:

Definition 1 A $\bar{m}\bar{l}$ -collaboration between Lenders and Borrowers with at most \bar{m} robot transfers and within at most \bar{l} steps, relative to Delay, is a partial function

$$f : \text{Lenders} \times \text{Borrowers} \rightarrow \{0, \dots, \bar{l}\} \times \{0, \dots, \bar{m}\}$$

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

(a) For every borrower team $j \in \text{Borrowers}$, there are some lender teams $i_1, \dots, i_s \in \text{Lenders}$ where

- $f(i_1, j) = (l_1, u_1), \dots, f(i_s, j) = (l_s, u_s)$ for some time steps $l_1, \dots, l_s \leq \bar{l}$ and some positive integers $u_1, \dots, u_s \leq \bar{m}$, and
- $\text{Delay}(i_1, j) = t_1, \dots, \text{Delay}(i_s, j) = t_s$ for some time steps $t_1, \dots, t_s \leq \bar{l}$;

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

$$\begin{aligned} \max\{l_1 + t_1, \dots, l_s + t_s\} &\leq \text{Borrow_latest}_m(j) \\ m &\leq \sum_{k=1}^s u_k. \end{aligned}$$

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

$$\begin{aligned} \text{Lend_earliest}_m(i) &\leq \min\{l_1, \dots, l_s\} \\ m &\geq \sum_{k=1}^s u_k. \end{aligned}$$

Condition (a) ensures that a borrower team does not borrow fewer robots than it needs. Condition (b) ensures that a lender team does not lend more robots than it

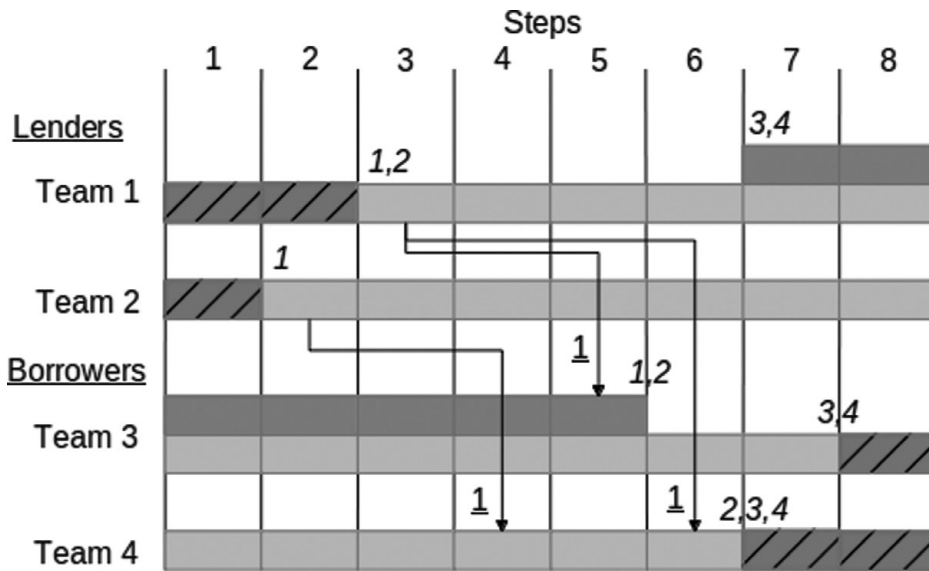


Fig. 1. A summary of teams' answers to queries.

can. These two conditions entail the existence of a lender team that can lend robots when a borrower team needs them.

Example 1 Consider four teams of robots, where Teams 1 and 2 are lenders and Teams 3 and 4 are borrowers. Take $\bar{l}=8$ and $\bar{m}=4$. The lenders' answers to questions of the form $Q2$ ("Can you complete your task in \bar{l} steps, if you lend m robots before step l ?") and the borrowers' answers to questions of the form $Q3$ ("Can you complete your task in \bar{l} steps, if you borrow m robots after step l ?") are summarized in Figure 1. The affirmative (resp., negative) answers to questions for time step l are denoted by green/solid (resp., red/hatched); the number m of robots that can be lent or needs to be borrowed are denoted above the rows. According to these answers, Team 1 can lend 2 robots after step 3 or 4 robots after step 7, Team 2 can lend 1 robot after step 2, Team 3 needs to borrow 1 robot before step 5 or 3 robots before step 7, and Team 4 needs to borrow 2 robots before step 6.

Suppose the delay time is $\text{Delay}(i, j) = |i - j|$. We can show that a $\bar{m}\bar{l}$ -collaboration f exists: $f(1, 3) = (3, 1)$, $f(1, 4) = (3, 1)$, $f(2, 4) = (2, 1)$. Indeed, f satisfies the conditions stated in Def. 1. Condition (a): for Team 3, $f(1, 3) = (3, 1)$, and there exists $m = 1 \leq 1$ s.t. $\text{Borrow_latest}_1(3) = 5 \geq 3 + 2$. So Team 3 can finish its task in 8 steps if it borrows 1 robot before step 5. Similarly, for Team 4, $f(1, 4) = (3, 1)$ and $f(2, 4) = (2, 1)$, and there exists $m = 2 \leq 1 + 1$ such that $\text{Borrow_latest}_2(4) = 6 \geq \max\{3 + 3, 2 + 2\}$. Condition (b): for Team 1, $f(1, 3) = (3, 1)$ and $f(1, 4) = (3, 1)$, and there exists $m = 2 \geq 1 + 1$ s.t. $\text{Lend_earliest}_2(1) = 3 \leq 3$. Similarly, for Team 2, $f(2, 4) = (2, 1)$, and there exists $m = 1 \geq 1$ such that $\text{Lend_earliest}_1(2) = 2 \leq 2$.

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

FINDCOLLABORATION

INPUT: For a set *Lenders* of lender teams, a set *Borrowers* of borrower teams, positive integers \bar{l} and \bar{m} : a delay function *Delay* and a collection of functions *Lend_earliest_m* and *Borrow_latest_m* for every positive integer m ($m \leq \bar{m}$).

OUTPUT: A \bar{m} -collaboration between *Lenders* and *Borrowers* with at most \bar{m} robot transfers and within at most \bar{l} steps, relative to *Delay*.

As expected, this problem is intractable:

Proposition 1 *The decision version of FINDCOLLABORATION (i.e., existence of a \bar{m} -collaboration) is NP-complete.*

Intuitively the membership proof is established by guessing and checking f in polynomial time. The hardness proof relies on a polynomial-time reduction from a 3SAT instance F with a atoms and b clauses, to a FINDCOLLABORATION problem instance with a lender teams and b borrower teams with $\bar{l} = 2a$ and \bar{m} defined over the number of occurrences of literals in F , and with no delays. Basically, we associate each atom with two time steps (denoting true resp. false); for each clause we define a borrower that can complete its work in $2a$ steps if it can borrow enough robots for at least one time step corresponding to a literal in the clause. We create lenders that can give the required numbers of robots either early (atom is true) or late (atom is false). We configure the number of robots associated with each literal such that a borrower's requirements can only be satisfied by the correct literals. The detailed proof is contained in Appendix B.

4 Finding a coordination of teams in ASP

Deciding whether a program in ASP has an answer set is NP-complete (Dantsin et al. 2001); therefore, ASP is suitable for solving FINDCOLLABORATION problem. We formalize FINDCOLLABORATION in ASP as follows.

The input is represented by a set of facts, using atoms of the forms *delay*(i, j, l), *lend_earliest*(i, m, l), and *borrow_latest*(j, m, l) where $i \in \text{Lenders}$, $j \in \text{Borrowers}$, $m \leq \bar{m}$, $l \leq \bar{l}$.

To formalize conditions (a) and (b) of Def. 1, we introduce atoms of the forms *condition_borrower*(j) and *condition_lender*(i). Condition (a) is defined as follows:

$$\begin{aligned} \text{condition_borrower}(j) \leftarrow & \text{sum}\{\langle u : f(i, j, l_2, u), i \in \text{Lenders}, l_2 \leq \bar{l} \rangle\} \geq m, \\ & \max\{\langle l_1 + t : f(i, j, l_1, u), \text{delay}(i, j, t), i \in \text{Lenders}, u \leq \bar{m} \rangle\} \leq l, \\ & \text{borrow_latest}(j, m, l). \end{aligned}$$

where $j \in \text{Borrowers}$, $l \leq \bar{l}$, $m \leq \bar{m}$. The second line of the rule above computes the number m of robots lent to the borrower team j ; the third line computes the latest time step l that team j borrows a robot; and the last line describes that team j needs m robots by step l . Similarly, we define condition (b).

Next, we introduce atoms of the form $f(i, j, l, u)$ (describing $f(i, j) = (l, u)$). We define an \bar{m} -collaboration f , by first “generating” partial functions f :

$$\{f(i, j, l, u) : l \leq \bar{l}, u \leq \bar{m}\} \leftarrow (i \in \text{Lenders}, j \in \text{Borrowers})$$

and then “eliminating” the ones that do not satisfy conditions (a) and (b) of Def. 1:

$$\begin{aligned} &\leftarrow \text{not condition_borrower}(j) && (j \in \text{Borrowers}), \\ &\leftarrow \text{not condition_lender}(i) && (i \in \text{Lenders}). \end{aligned}$$

With the ASP formulation of FINDCOLLABORATION above, an ASP solver can find an \bar{m} -collaboration.

5 Finding an optimal plan for multiple teams

Once a coordination of teams is found using an ASP solver for an optimal global plan with $\bar{l} \leq k$ steps, the mediator informs each team how many robots it is expected to lend/borrow to/from which team and when, along with the optimal plan length \bar{l} . Taking this information into account, each team computes an optimal local plan with at most \bar{l} steps using an ASP solver, to complete its task, as described in Section 2. The union of these optimal local plans gives us an optimal global plan.

In a naive approach, every team answers $O(\bar{m} \cdot k^2)$ queries, within this overall algorithm. We can improve it by applying binary search between 1 and \bar{l} to find the earliest lend times and the latest borrow times, and between 1 and k to find the optimal value for \bar{l} . With this improvement, every team answers $O(\bar{m} \cdot \log(k)^2)$ queries.

On the other hand, the computation time to answer a query drastically increases as the plan length increases (due to inherent hardness of planning (Erol *et al.* 1995; Turner 2002)). In such cases, as suggested by Trejo *et al.* (Trejo *et al.* 2001), it is not a good idea to apply binary search to find the optimal value \bar{l} for a global plan. Therefore, a better approach might be to use linear search to find the optimal value \bar{l} for a global plan, and binary search to find optimal values for lending/borrowing times. This approach leads to more number of queries (i.e., every team answers $O(\bar{m} \cdot k \cdot \log(k))$ queries) but less amount of computation times as verified by experiments (Table A2 in Appendix A).

Note that since each team computes an optimal local plan on its own, optimal local plans can be computed in parallel as well.

6 Experimental evaluation

We investigated the scalability and usefulness of the proposed planning approach (e.g., in terms of quality of solutions) by means of some experiments.

We performed some experiments in a variation of the Painting Factory domain described in Erdem *et al.* (2012). In this domain, a set of boxes must be manufactured within a given time. To manufacture a box it has to undergo various stages of painting, waxing, and stamping, obeying certain time constraints. There are two types of robots: *worker robots* operate on boxes, they can configure themselves

Table 1. Experimental results for six scenarios

Scenario	Teams	Workspace (grid cells)		Worker Robots		Total Robots	Order (boxes)	Questions (total)	Answering Questions (average time)	Finding Collaboration (average time)	Optimal Global Plan (with collaboration)	Optimal Global Plan (without collaboration)
	#	#	#	#	#	#	#	sec	sec	length	length	
1	2	15	1,2	5	6	212	3.96	< 0.1	30	34		
2	3	15	1,2,3	9	9	437	3.92	< 0.1	25	34		
3	4	15	1,2,3,4	15	12	525	1.82	< 0.1	21	34		
4	2	24	2,4	8	8	127	4.76	< 0.1	20	29		
5	3	24	2,4,6	18	12	171	5.37	< 0.1	18	29		
6	4	24	2,4,6,8	30	16	293	79.96	< 0.1	18	29		

for different stages of process, 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. We assembled teams of different sizes in this domain, so that exchanging worker robots between teams can reduce the time that is necessary to produce the requested amount of boxes.

For our experimental scenarios (see Table 1), we considered team workspaces of $5 \times 3 = 15$ (resp., $8 \times 3 = 24$) grid cells. We varied the number of teams, the number of robots in each team, and the number of boxes that must be manufactured by each team. In each team, for every two worker robots, there is one charger robot. As an example we discuss Scenario 5: three teams must manufacture 12 boxes. Each has a $8 \times 3 = 24$ workspace. The teams consist of 2, 4, resp., 6 worker robots. Accordingly, the teams have 1, 2, resp., 3 charger robots; this yields a total of 18 robots in Scenario 5. The sizes of the workspaces in these instances are reasonable considering real manufacturing processes, since every work cell in a real factory typically is of modest size with 3–12 operators (in our case 2–9 robots per workspace). The number of work cells in a factory ranges drastically from micro factories to large manufacturing plants; with the utilization of parallelization to answer queries, the number of workspaces can be increased further.

We performed experiments on a Linux server with 32 Intel® E5-2665 CPU cores with 2.4GHz and 64GB memory (note that our experiments never use more than 300MB). The overall algorithm described in Section 5 is implemented in PYTHON. The ASP solver CLASP version 2.1.3 (with GRINGO version 3.0.5) is used for answering queries (Section 2) and to solve the collaboration problem (Section 4).

Table 1 shows the results for six scenarios of varying size, averaged over three runs. For each scenario, we report the total number of questions answered by the teams, the average CPU time to answer a query and to find a coordination of teams, the length of an optimal global plan with/without collaborations of teams. For instance, for Scenario 5, a total of 171 queries are answered by the teams; average time to answer a query is 5.37 seconds; and finding a coordination of teams takes

less than a second. An optimal global plan with such a coordination has 18 steps; whereas an optimal global plan without any collaborations has 29 steps.

We can observe from the table that finding a coordination function by the mediator takes a negligible amount of time. The majority of the total computation time is spent for the teams to answer questions. As the problem size increases, the size of the ASP program gets larger, making it hard for CLASP to find an answer. Since teams' query answering can be parallelized, scalability of the approach to factories with many workspaces seems plausible.

We can also observe the tradeoff between the optimal global plan length and the total computation time, with and without team collaborations. For instance, for Scenario 2, if we allow collaborations of teams, then we can find an optimal global plan of length 25, in about 33 minutes; otherwise, we can find an optimal global plan of length 34 in about 5 minutes. This computational cost is negligible compared to 27% decrease in process length resulting in large cost savings for the manufacturing industry; time gains achieved by such a decrease in process length will help economic sustainability under low quantity orders, and result also in better customer satisfaction.

7 Related work

The most related work to ours is on decoupling plans of multiple agents to coordinate their actions (M. M. de Weerd 2009), and can be summarized in three parts:

Coordination before planning: These methods coordinate the agents before they even begin to plan. Some of them introduce *social laws* the agents must follow (Shoham and Tennenholtz 1995; ter Mors *et al.* 2004). These laws restrict the actions of agents and can be used to reduce planning and coordination time (e.g., if everyone drives on the right side of the road, no coordination with oncoming cars is required).

Coordination during planning: In these methods, agents find plans for themselves while sharing information about them, and adapt their plans accordingly to avoid conflicts. Partial Global Planning (PGP) framework (Durfée and Lesser 1987) and its extension, Generalized PGP (Decker and Lesser 1994), are examples of these sorts of methods. In these approaches, agents share their plans using a specialized plan representation. Coordination is achieved as follows: if an agent informs a second agent of its own plan, the second agent merges this information into its own partial global plan. The second agent then tries to improve the global plan. If it can, the improved plan is shown to the other agents who can accept/reject/modify it. Another example of this approach is the Plan Merging Paradigm (Alami *et al.* 1998), where each robot incrementally builds and executes its own plan taking into account the multi-robot context. There are also coordination methods where the agents exchange subgoals with auctions (van der Krogt *et al.* 2005).

Coordination after planning: These methods use plan merging. Given the individual plans of all agents, plan merging constructs a joint plan for all agents. Georgeff (1988) proposes a plan-synchronization process starting with individual plans. Stuart (1985) uses propositional temporal logic to guarantee that only feasible states of the environment can be reached. Introducing restrictions on individual plans (as in

coordination before planning) can be used to ensure efficient merging (Foulser *et al.* 1992; Yang *et al.* 1992). Another approach to merging plans is to use A^* search with a smart cost-based heuristic (Ephrati and Rosenschein 1993). There are also task allocation methods that assign the roles of agents for the execution of a given plan (Hunsberger and Grosz 2000).

Our method is different from these related work in the following ways: 1) In our approach, the teams do not communicate with each other, but with a mediator. 2) The communication is not done by passing information about plans or durations of actions: the teams answer the mediator's yes/no questions, ensuring that the teams do not have to share private information (e.g., workspace, tasks, goals) with the mediator; once a coordination is found, then the mediator informs each team about how many robots it should lend/borrow and when, so the teams do not know about which other teams lend/borrow robots. 3) Like the related work, each team computes its own an optimal local plan to complete its task taking into account some extra information; but this extra information is not about other teams (e.g., their plans, actions, tasks). 4) 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. 5) Such a coordination is found iteratively where each iteration involves individual teams' solving various planning problems with complex goals to answer mediator's questions; so determining a feasible coordination goes hand-in-hand with planning. 6) Our method assumes that a team cannot be both a lender or borrower, to ensure a small number of costly transfers of robots between teams; on the other hand, we do not assume that all teams are in the same workspace. Note that once a coordination of teams is found, then an optimal global plan is computed by combining the local plans (as in related work where coordination is done before/after planning).

Our work is more about team work to find a (optimal) global plan, like the related work discussed above, where teams are determined in advance (and in our case costly transfers of robots are not desired), rather than team formation (to decide how or when to join teams) (Nair *et al.* 2002; Gaston and desJardins 2008).

Our work is also different from the existing approaches on resource allocation in a multi-agent time-constrained domain Sycara *et al.* (1991); Chevaleyre *et al.* (2006); Lin (2011) due to the second item above, because in our method no information is required about plans, ordering constraints on actions, or causal links.

It is important to note here that, the mediator in our approach is a neutral coordinator like in Ehtamo *et al.* (1999), though it does not negotiate with the teams but simply gathers information to achieve a 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.

Distributed planning for multiple agents with the help of a supervisor, has been studied using action languages (Dovier *et al.* 2013) and logic programming (Kowalski and Sadri 2013) as well. There are essential differences between these works and the proposed approach: differences in the expressivity of the languages (e.g., the action language in (Dovier *et al.* 2013) does not allow representation of ramifications, but on the other hand it includes formulas to represent communication messages); the

role of the supervisor (in both of the related works, the role of the supervisor is to resolve conflicts; in our work, it is to decide for efficient use of resources); communication of teams and supervisor (in the related approaches, agents can communicate and negotiate between each other, and the supervisor collects teams' to-be-executed actions to check for conflicts);

8 Discussion

We have introduced a novel method to find an optimal global plan for multiple teams of robots, by means of determining a coordination of teams based on their answers to yes/no questions that do not convey private information about their workspace, tasks, robots, plans, actions, goals, etc.. We have defined the problem of determining a coordination, and proved its intractability. Using the state-of-the-art ASP solvers, we have evaluated the usefulness of our approach in a cognitive factory setting, and observed a promising decrease in the total process time, which is important for larger cost savings and better customer satisfaction towards economic sustainability.

Lessons learnt The generic nature of this method allows other reasoners and solvers to be used for planning, query answering, or coordination finding. We have used CCALC; because 1) workspaces we consider in a cognitive factory involve concurrency and ramifications (which can be easily formalized in the input language of CCALC), and require external computations in continuous space of robots' configurations (e.g., to check collisions) as in Erdem *et al.* (2011); Aker *et al.* (2012); Havur *et al.* (2013); and 2) planning problems and queries we consider involve complex goals and conditions. Also, CCALC can be used with a wide range of reasoners, like SAT solvers and ASP solvers.

We have used ASP for finding a coordination of teams, since it provides a concise provably correct description of the problem and the computation times with the state-of-the-art ASP solvers are quite good.

The fact that in our approach each team performs its own computations about completing its own task leads to a highly modular structure of computation and allows computation of local plans and answering queries in parallel.

In addition to the strengths of using these logic-based formalisms from the point of view of representation and efficient reasoning, it is also important that these formalisms are actually being used for challenging robotic applications; making it easier to apply our approach to robotic domains, like in cognitive factories (Erdem *et al.* 2012).

Future work Our approach can be extended in several ways. If the mediator is allowed to know about the basic tasks and the sorts of transferrable robots, then it can ask questions like “Can your team complete its task in k steps, while also lending a robot that can carry a heavy box, before step k' ?”. Teams can answer such queries because the background knowledge that associates tasks with robots can be embedded in action descriptions and queries, as in Erdem and Patoglu (2012); Aker

et al. (2012; Erdem et al. (2012). The ASP formulation for finding a coordination can be slightly modified by adding relevant constraints.

Our algorithm to find an optimal global plan can be embedded in an execution monitoring framework. When the plan fails during execution, our algorithm can be called to find an optimal global plan. For subsequent replans, we can reuse the information (e.g., roles of teams and bounds for each team) from the previously computed plan.

Acknowledgements

This work is partially supported by TUBITAK Grant 111E116. Peter Schueller is supported by TUBITAK 2216 Fellowship. Tansel Uras' work is carried out while studying at Sabancı University.

References

- 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*.
- ALAMI, R., INGRAND, F. AND QUTUB, S. 1998. A scheme for coordinating multi-robots planning activities and plans execution. In *Proc. of ECAI*, 617–621.
- 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. *Communications of the ACM* 54, 12, 92–103.
- CASOLARY, M. AND LEE, J. 2011. Representing the language of the causal calculator in answer set programming. In *Proc. of ICLP (Technical Communications)*, 51–61.
- CHEVALEYRE, Y., DUNNE, P. E., ENDRIS, U., LANG, J., LEMAÎTRE, M., MAUDET, N., PADGET, J. A., PHELPS, S., RODRÍGUEZ-AGUILAR, J. A. AND SOUSA, P. 2006. Issues in multiagent resource allocation. *Informatica* 30, 1, 3–31.
- DANTSIN, E., EITER, T., GOTTLÖB, G. AND VORONKOV, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys* 33, 3, 374–425.
- DECKER, K. AND LESSER, V. 1994. Designing a family of coordination algorithms. In *Proc. of DAI*, 65–84.
- DOVIER, A., FORMISANO, A. AND PONTELLI, E. 2013. Autonomous agents coordination: Action languages meet clp() and linda. *TPLP* 13, 2, 149–173.
- DURFEE, E. H. AND LESSER, V. R. 1987. Planning coordinated actions in dynamic domains. In *Proc. of the DARPA Knowledge-Based Planning Workshop*, 18.1–18.10.
- EHTAMO, H., HAMALAINEN, R. P., HEISKANEN, P., TEICH, J., VERKAMA, M. AND ZIONTS, S. 1999. Generating pareto solutions in a two-party setting: Constraint proposal methods. *Management Science* 45, 12, 1697–1709.
- EPHRATI, E. AND ROSENSCHEIN, J. S. 1993. Multi-agent planning as the process of merging distributed sub-plans. In *Proc. of DAI*, 115–129.
- 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., 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 International Conference on Emerging Technologies and Factory Automation (ETFA)*.
- ERDEM, E. AND PATOGLU, V. 2012. Applications of action languages in cognitive robotics. In *Correct Reasoning* 229–246.
- EROL, K., NAU, D. S. AND SUBRAHMANYAN, V. S. 1995. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence* 76, 1–2, 75–88.
- FOULSER, D., LI, M. AND YANG, Q. 1992. Theory and algorithms for plan merging. *Artificial Intelligence Journal* 57, 143–182.
- GASTON, M. E. AND DESJARDINS, M. 2008. The effect of network structure on dynamic team formation in multi-agent systems. *Computational Intelligence* 24, 2, 122–157.
- GEBSER, M., KAMINSKI, R., KÖNIG, A. AND SCHAUB, T. 2011. Advances in gringo series 3. In *Proc. of LPNMR*, 345–351.
- 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. 1998. Action languages. *Electronic Transactions on Artificial Intelligence* 2, 193–210.
- GEOGHEFF, M. P. 1988. Communication and interaction in multi-agent planning. In *Readings in Distributed AI*, 200–204.
- GIUNCHIGLIA, E., LEE, J., LIFSCHITZ, V., MCCAIN, N. AND TURNER, H. 2004. Nonmonotonic causal theories. *AIJ* 153, 49–104.
- HAMADI, Y., JABBOUR, S. AND SAIS, L. 2009. Control-based clause sharing in parallel sat solving. In *Proc. of IJCAI*, 499–504.
- 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*.
- HUNSBERGER, L. AND GROSZ, B. J. 2000. A combinatorial auction for collaborative planning. In *Proc. of ICMAS*, 151–158.
- KOWALSKI, R. A. AND SADRI, F. 2013. Towards a logic-based unifying framework for computing. *CoRR abs/1301.6905*.
- 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*, MIT Press, 1594–1597.
- LIN, S.-H. 2011. Coordinating time-constrained multi-agent resource sharing with fault detection. In *Proc. of IEEM*, 1000–1004.
- M. M. DE WEERDT, B. J. C. 2009. Introduction to planning in multiagent systems. *Multiagent and Grid Systems* 5, 345–355.
- 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.
- MCCAIN, N. AND TURNER, H. 1997. Causal theories of action and change. In *Proc. of AAAI/IAAI*, 460–465.
- NAIR, R., TAMBE, M. AND MARSELLA, S. 2002. Team formation for reformation in multiagent domains like robocuprescue. In *Proc. of RoboCup*, 150–161.
- NIEMELÄ, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25, 241–273.
- SHOHAM, Y. AND TENNENHOLTZ, M. 1995. On social laws for artificial agent societies:off-line design. *Artificial Intelligence* 73, 231–252.

- STUART, C. 1985. An implementation of a multi-agent plan synchronizer. In *Proc. of IJCAI*, 1031–1033.
- SYCARA, K. P., ROTH, S. P., SADEH, N. M. AND FOX, M. S. 1991. Resource allocation in distributed factory scheduling. *IEEE Expert* 6, 1, 29–40.
- TER MORS, A., VALK, J. AND WITTEVEEN, C. 2004. Coordinating autonomous planners. In *Proc. of IC-AI*, 795–801.
- 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.
- TURNER, H. 2002. Polynomial-length planning spans the polynomial hierarchy. In *Proc. of JELIA*, 111–124.
- VAN DER KROGT, R., ROOS, N., DE WEERDT, M. AND WITTEVEEN, C. 2005. Multiagent planning through plan repair. In *Proc. of AAMAS*, 1337–1338.
- YANG, Q., NAU, D. S. AND HENDLER, J. 1992. Merging separately generated plans with restricted interactions. *Computational Intelligence* 8, 648–676.
- 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 Reconfigurable Manufacturing Systems*, 355–371.
- ZAEH, M., OSTGATHE, M., GEIGER, F. AND REINHART, G. 2012. Adaptive job control in the cognitive factory. In *Enabling Manufacturing Competitiveness and Economic Sustainability*, H. A. ElMaraghy, Ed. Springer, Berlin Heidelberg, 10–17.