Autonomous Robots (2019) 43:213–238
https://doi.org/10.1007/s10514-018-9721-x

Finding optimal feasible global plans for multiple teams of heterogeneous robots using hybrid reasoning: an application to cognitive factories

Zeynep G. Saribatur[1] · Volkan Patoglu[2] · Esra Erdem[2]

Received: 30 August 2016 / Accepted: 20 March 2018 / Published online: 30 March 2018

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 modeling each team's workspace taking into account capabilities of heterogeneous robots, embedding continuous external computations into discrete symbolic representation and reasoning, not only optimizing the makespans of local plans but also minimizing the total cost of robotic actions. To find an optimal global plan, we propose a semi-distributed approach that does not require exchange of information between teams but yet achieves on an optimal coordination of teams that can help each other. We prove that the optimal coordination problem is NP-complete, and describe a solution using automated reasoners. We experimentally evaluate our methods, and show their applications on a cognitive factory with dynamic simulations and a physical implementation.

Keywords AI reasoning methods · Optimal global planning · Hybrid reasoning · Coordination of multiple teams · Intelligent and flexible manufacturing

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 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 (Zaeh et al. 2009; Erdem et al. 2012b) 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.

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

This work is partially supported by Scientific and Technological Research Council of Turkey (TUBITAK) Grant 111E116. Z. G. Saribatur's work was carried out during her graduate studies at Sabancı University.

✉ Esra Erdem
esraerdem@sabanciuniv.edu

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria

[2] Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey

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 et al. 1995; Erdem et al. 2013).

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 (Erdem et al. 2011, 2016b) 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 (Erdem et al. 2012b) 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 an RGB-D camera into the domain description, while we rely on guided replanning to account for possible robot-robot collisions (Erdem et al. 2016b). 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 communication 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 (Erdem et al. 2013). 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) (Brewka et al. 2011) 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), 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.

This article is a significant extension of our paper (Saribatur et al. 2014) presented at 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, with complexity, correctness and completeness results, algorithms, and extensive experimental evaluations.

2 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 (de Weerdt and Clement 2009) 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 (Shoham and Tennenholtz 1995; ter Mors et al. 2004) or putting restrictions on local plans to be able to merge them or synchronize them into a global plan (Yang et al. 1992; Foulser et al. 1992; Stuart 1985; Georgeff 1988), or by exchanging information between teams about their partial plans or goals (ter Mors et al. 2004; Decker and Lesser 1994; Alami et al. 1998). 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. Indeed, each team communicates with the mediator, but by only answering the mediator's yes/no questions; so the teams do not have to share private information with each other. 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); so no social law or restriction is enforced on the order of robotic actions for local plans. Moreover, we do not assume that all teams are in the same workspace, or all robots are of the same sort. Unlike these related work, 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.

The mediator in our approach is a neutral coordinator like in Ehtamo et al. (1999); however, it does not negotiate with the teams or try to resolve conflicts as in Dovier et al. (2009) but simply gathers information from them to find an optimal global solution. Unlike in the existing works on task assignment and scheduling (Tan and Khoshnevis 2000; Hooker 2005; Luo et al. 2013; Gombolay et al. 2013), the mediator does not try to assign/schedule tasks but only informs the teams about when they are expected to lend/borrow robots.

In our approach, the teams do not communicate with each other, and the mediator does not know anything about the teams's goals, tasks or workspaces; this aids privacy in some applications. In that sense, our work has similar motivations with studies on privacy preserving multi-agent planning (Brafman and Domshlak 2008; Torreño et al. 2012, 2015; Bonisoli et al. 2014; Maliah et al. 2014; Brafman 2015). In these related work, there is usually some classification of actions as private or public actions and/or some communication of partial plans of public actions. In our approach, all teams' actions are private.

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 (Sycara et al. 1991; Chevaleyre et al. 2006; Lin 2011), 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 (Nair et al. 2002; Gaston and desJardins 2008) as well, because our method does not aim for deciding how or when to join teams. Although each team utilizes feasibility checks, like collision checks, our work is different from various works on multi-robot motion planning (LaValle and Hutchinson 1998; Desaraju and How 2012; Turpin et al. 2013), because our method also considers task planning. Also, it is not specific to path finding as in Velagapudi et al. (2010).

Another important aspect of our approach to finding optimal global plans, and its difference from the related work discussed above, is its hybrid nature. Integration of task planning with motion planning has been studied by various researchers in the recent years. Some of these related work integrates task planning with feasibility checks at the representation level (Caldiran et al. 2009; Hertle et al. 2012; Erdem et al. 2011, 2012a; Havur et al. 2013, 2014; Gaschler et al. 2013) and some at the search level (Gravot et al. 2005; Hauser and Latombe 2009; Plaku and Hager 2010; Wolfe et al. 2010; Kaelbling and Lozano-Pérez 2013; Srivastava et al. 2014; Lagriffoul et al. 2014; Dantam et al. 2016). At the search level, the integration takes advantage of a search algorithm to incrementally build a task plan, while checking its kinematic/geometric feasibility at each step by a motion planner; all these approaches use different methods to utilize the information from the task-level to guide and narrow the search in the configuration space. In this way, the task planner helps the search process during motion planning. At the representation level, the integration is done via a general interface using "external atoms" [in the spirit of semantic attachments in theorem proving (Weyhrauch et al. 1978)] in the representation of the robotic domain. So, instead of guiding the task planner at the search level by manipulating its search algorithm directly, the motion planner guides the task planner at the representation level by means of external atoms. A detailed discussion about these works can be found in Erdem et al. (2016b). Our approach to hybrid planning in this study is at the representation level using ASP, like in Erdem et al. (2012a) and Havur et al. (2014).

Note that the related work on integration at search level focus on hybrid planning for a single robot. Most of the related work on integration at representation level, except for our earlier studies, also focus on hybrid planning for a single robot; our earlier studies (Erdem et al. 2012a) consider hybrid planning for a team of multiple robots. Unlike these related work, we focus on hybrid planning with multiple teams of heterogeneous robots.

This article significantly extends our earlier works (Erdem et al. 2013; Saribatur et al. 2014). Indeed, Saribatur et al. (2014) extends our earlier study (Erdem et al. 2013) by considering heterogeneous capabilities of robots and generalizes the formal framework (including definitions, mathemati-

cal modeling and formalizations) accordingly. This article further extends (Saribatur et al. 2014) with complexity, correctness and completeness results, algorithms, and extensive experimental evaluations.

3 Answer set programming

The idea of Answer Set Programming (ASP) (Brewka et al. 2011) is to represent a problem as a "program" whose models [called "answer sets" (Gelfond and Lifschitz 1991)] 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 (Gebser et al. 2007), 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 (Nogueira et al. 2001; Tiihonen et al. 2003; Ricca et al. 2012) and robotics (Erdem et al. 2016a).

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 (Eiter et al. 2005)] that consists of rules

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

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

$$\bot \leftarrow at(r, loc_1, t), goto(r, loc_2, t), \\ not\ \&path\_exists[loc_1, loc_2]() \quad (1)$$

is used to express that, at any step t of the plan, a robot r cannot move from loc_1 to loc_2 if there is no collision-free trajectory between them. Here collision check is done by the external predicate *&path_exists*, which can, for instance, implemented in C++, utilizing the bidirectional RRT (Rapidly Exploring Random Trees) (Kuffner and LaValle 2000a) as in the OMPL (Sucan et al. 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).

Recall that a plan P of makespan n is a sequence $\langle a_0, a_1, \ldots, a_{n-1} \rangle$ of n actions. Each action a_i may consist of a single action, e.g.,

$$a_2 = \{goto(R_1, A, 2)\},$$

which describes R_1 going to location A at time step 2, or a set of actions executed concurrently, e.g.,

$$a_5 = \{goto(R_1, A, 5), goto(R_2, B, 5)\},$$

which describes R_1 going to location A while R_2 going to location B at time step 5.

For instance, suppose that the cost of a robot r performing an action is defined by a cost function *cost* that maps every robot r to a nonnegative integer c. Let us denote by $rob_P(t)$ the set of robots that are involved in executing an action at time step t of a plan P. Assume that costs c of actions performed by robot r at time step t in a plan are defined by atoms of the form $cost(r, c, t)$. We can ensure that the sum of all costs of robotic actions performed in a plan P whose makespan is n, i.e.,

$$\sum_{0 \leq t < n, r \in rob_P(t)} cost(r),$$

is less than some given number k by the following constraint:

$$\leftarrow \#sum\ \{c, r, t : cost(r, c, t)\} \geq k.$$

We can minimize the total cost function by the following optimization statement:

$$\#minimize\ \{c, r, t : cost(r, c, t)\}. \quad (2)$$

4 Hybrid reasoning for a team of heterogeneous robots

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

4.1 Actuation actions

To compute a task plan by an ASP solver, actions of a robot are described by rules. For instance, the following rule

describes a direct effect of a "goto" action of a mobile robot r navigating to a location loc in the cognitive factory floor at time step t:

$$at(r, loc, t+1) \leftarrow goto(r, loc, t).$$

It expresses that, after r goes to loc, the location of r changes to loc at the next time step $t+1$.

The following constraint expresses a precondition of "goto" action (i.e., a robot cannot goto a location if it is already there):

$$\leftarrow goto(r, loc, t), at(r, loc, t).$$

In a similar way, we can describe indirect effects, state constraints and transition constraints in ASP (Erdem et al. 2016a).

4.2 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:

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

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

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

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

4.3 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 Erdem et al. (2016b): (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 Erdem et al. (2011).

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 loc 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 loc occupied by an obstacle at any time step t:

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

where $\&obstacleAt$ is an external predicate whose value is determined by an object detection algorithm (Duff et al. 2013) using Point Cloud Library over data obtained by an RGB-D camera. We can also express a transition constraint to avoid collisions of robots with obstacles while the robots move from one location loc_1 to another loc_2 along grid lines or diagonally to follow straight trajectories:

$$\leftarrow at(r, loc_1, t), goto(r, loc_2, t), \\ \&collision[r, loc_1, loc_2]().$$

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

Note that, since we view the environment as a grid and consider polygonal holonomic robots navigating along grid lines or diagonally to follow straight trajectories, the collision checks are complete. In that sense, adding the constraint above does not cause the hybrid planning to become incomplete (see Sect. 4.5 for further discussions on completeness).

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 $goto(r, loc, t)$ (describing which robot is moving where at time step t) and of the form $at(r, loc, t)$ (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 R_1 and R_2 cannot cross each other diagonally between locations A and B, then we can add the following constraint to the planning problem

$$\leftarrow at(R_1, A, t), at(R_2, B, t), \\ at(R_1, B, t+1), at(R_2, A, t+1)$$

to guide the ASP solver to find a new plan where the robots R_1 and R_2 do not exchange their locations A to B, respectively, at any time t.

As noted above, since the polygonal holonomic robots navigate along grid lines or diagonally to follow straight trajectories, the collision checks are complete in this domain. In that sense, adding the constraint above does not cause the hybrid planning to become incomplete (see Sect. 4.5).

4.4 Optimal planning

As explained in the previous subsections, once the actions and 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 Sect. 3) with respect to different optimization functions.

To find a shortest plan $P = \langle a_0, a_1, \ldots, a_{n-1} \rangle$, whose length is n (i.e., a plan with minimum makespan n), we can perform a linear search on the makespan n between a lower bound and an upper bound (e.g., $n = 0, 1, \ldots, k$ for some large integer k) as suggested in Trejo et al. (2001). Alternatively, we can minimize the time step to reach a goal:

$$\#minimize\ \{t : goal(t)\}$$

where $goal(t)$ expresses that the goal conditions are satisfied at time step 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 loc_1 to another location loc_2 by an external function *time_estimate*:

$$cost(r, c, t) \leftarrow at(r, loc_1, t), goto(r, loc_2, t), \\ \&time\_estimate[r, loc_1, loc_2](c).$$

The external function *time_estimate* estimates the duration in terms of a continuous trajectory for r between loc_1 and loc_2 computed by a motion planner, as in Erdem et al. (2012a).

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 estimates of certain processes).

4.5 Discussion: completeness and optimality

The proposed hybrid planning approach inherits its completeness and optimality properties from both the high-level reasoning and the low-level feasibility checks.

4.5.1 Completeness

The main high-level reasoning task involved in hybrid planning is task planning. The completeness analysis of task planning algorithms is usually done with respect to a parameterized task planning problem, where an upper bound k is specified as an input on the makespan (i.e., the plan length) or the total cost of actions. A task planning algorithm that is designed to compute a task plan (to reach a goal state from the initial state) whose makespan is at most k is (k-)complete, if the algorithm computes such a task plan (if one exists) or it finds out that there is no such task plan.

There are various types of feasibility checks, for instance, to ensure stability, graspability or collision-free motion of robots. The most commonly employed low-level feasibility check involved in hybrid planning is motion planning that checks for the existence of a collision-free trajectory for a navigating robot to move to a location to another or a manipulator to reach a target item. The completeness of motion planning algorithms can be characterized into complete (Canny 1988), resolution complete (Latombe 1991) and probabilistically complete methods (Kavraki et al. 1996; Kuffner and LaValle 2000b). In the latter two cases, the com-

pleteness analysis is performed under the assumption that the resolution or sample size approaches infinity (i.e., that the algorithm may not terminate at all), making it hard to analyze the completeness of the implemented motion planning algorithms that terminate when they reach the specified upper bound on time, resolution size or sample size.

Like in task planning, we propose to analyze the completeness of such underlying low-level feasibility checks with respect to a parameterized motion planning problem, where an upper bound m is specified as an input to specify the resolution size (if the motion planning algorithm is cell-decomposition based) or the sample size (if the motion planning algorithm is deterministic sampling based). A motion planning algorithm to compute a motion plan (to reach a goal configuration from the initial configuration) over a finite configuration space with a resolution or sample size of at most m is (m-)complete, if the algorithm computes a motion plan in that finite configuration space (if a motion plan exists in that finite configuration space) or it finds out that there is no motion plan in that finite configuration space. Note that as the parameter m approaches infinity, (m-)completeness leads to resolution or probabilistic completeness.

Having defined the completeness of task planning algorithms and motion planning algorithms with respect to parameterized versions of task planning problem and motion planning problem, we can analyze the completeness of our hybrid planning algorithm more clearly, in the spirit of the proof of Theorem 1 of Dantam et al. (2016).

(i) The ASP-based high-level reasoning approach used for task planning in our hybrid planning method is k-complete (Giunchiglia et al. 2004, Proposition 11) (i.e., all task plans whose makespans are less than a given upper bound can be computed, if there exists such a task plan; otherwise, it terminates with no plan). Moreover, this approach can generate optimal task plans with respect to a given action cost function (i.e., task plans with minimum total cost of actions).

(ii) If the motion planning algorithm used for feasibility checks is m-resolution complete, then our hybrid planner can compute all task plans whose makespans are at most k and whose feasibility are verified with respect to a resolution size of at most m. In this case, we say that our hybrid planner is km-resolution complete under the basic connectivity assumption.

(iii) If the motion planning algorithm used for feasibility checks is m-probabilistically complete, then our hybrid planner can compute all task plans whose makespans are at most k and whose feasibility are verified with respect to a deterministic sampling size of at most m. In this case, we say that our hybrid planner is km-probabilistically complete under the basic connectivity assumption.

As the values of the parameters k and m change, the size of the combined search space changes, and thus the completeness of the hybrid planning algorithm.

Note that, in general, the basic connectivity assumption, as defined in Dantam et al. (2016), is required to guarantee completeness of hybrid planning, as it ensures that collision-free trajectories computed for consecutive actions of the task plan are compatible with each other.

In the application of our hybrid planning approach to cognitive factories, we view the environment as a grid and consider polygonal holonomic robots navigating along grid lines or diagonally to follow straight trajectories so that the robots do not collide with static obstacles or with each other. Therefore, the feasibility checks can be performed simply with complete algorithms. On the other hand, since the environment is viewed as a grid, the makespan for task plans is considered a large enough number so as to make ASP-based task planning complete. Furthermore, basic connectivity is trivially satisfied. Then, our hybrid planning algorithm with a large enough upper bound on makespan becomes complete.

4.5.2 Optimality

Our hybrid planning approach considers optimality in terms of minimizing the makespan or the total cost of actions, provided that an upper bound k is given on the makespan or the total cost of actions. The ASP-based task planning is guaranteed to find an optimal task plan, whose makespan or the total cost is at most k, with respect to these optimization functions, if such a task plan exists. Therefore, if the feasibility checks are m-complete, then the hybrid planning approach is optimal with respect to the upper bounds k and m.

In our application of the hybrid planning approach to cognitive factories, since the hybrid planning algorithm is complete, optimal hybrid plans are computed.

5 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. (2013):

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. (2013): 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.

5.1 Querying teams

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. (2013) 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.

5.2 Inferring knowledge about robot transfers

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 $Lenders_x \subset Lenders$ of lender teams that can lend robots of type x and sets $Borrowers_x \subset Borrowers$ of borrower teams that needs to borrow robots of type x, where $x \leq n$ ($Lenders, Borrowers \subset Teams$): 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.

5.3 Coordination of teams

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

$$Lend\_earliest_{m,x} : Lenders_x \mapsto \{0, \ldots, \bar{l}\}$$
$$Borrow\_latest_{m,x} : Borrowers_x \mapsto \{0, \ldots, \bar{l}\}$$

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:

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

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

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

(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 $j \in Borrowers_x$, there are some lender teams $i_1, \ldots, i_s \in Lenders_x$, $x \leq n$, 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,

– $Delay(i_1, j, x) = t_1, \ldots, Delay(i_s, j, x) = t_s$ 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

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

(b) For every lender team $i \in Lenders_x$, for all borrower teams $j_1, \ldots, j_s \in Borrowers_x$, $x \leq n$, 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, there is a positive integer $m \leq \overline{m}_x$ such that

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

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 $Lend\_earliest_{m,x}$ and $Borrow\_latest_{m,x}$ for every positive integer $m \leq \overline{m}_x$, $x \leq n$.
OUTPUT: A $n\overline{ml}$*-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, FINDCOLLABORATION_n is NP-hard even when the robots are homogeneous (assuming that P $\neq$ NP).

Example 1 Consider five teams of robots, where Teams 1 and 2 are lenders and Teams 3, 4 and 5 are borrowers. Suppose that there are two types of robots (i.e., $n = 2$). Take $\bar{l} = 8, \overline{m}_1 = 3, \overline{m}_2 = 3$. The lenders' answers to questions of the form Q2 ("Can you complete your task in $\bar{l}$ steps, if you lend m robots of type x 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 of type x after step l?") are summarized in Fig. 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.

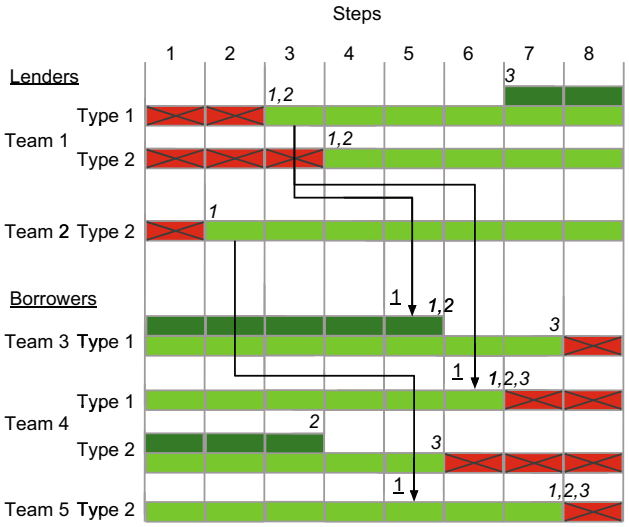

Fig. 1 A summary of teams' answers to queries Q2 and Q3 in Example 1

According to these answers, Team 1 can lend at most 2 robots of type 1 after step 3, or 3 robots after step 7. Team 2 can only lend 1 robot of type 2 after step 2. Team 3 needs to borrow at least 1 robot of type 1 before step 5, or 3 robots of type 1 before step 7. Team 4 needs to borrow at least 1 robot of type 1 before step 6, 2 robots of type 2 before step 3, or 3 robots of type 2 before step 5. Team 5 needs to borrow at least 1 robot before step 7.

Suppose that the delay function is $Delay(i, j, x) = |i - j|$. Then an $n\overline{ml}$-collaboration f exists between the lenders (i.e., Teams 1 and 2) and the borrowers (i.e, Teams 3–5) with at most $\overline{m} = \max\{\overline{m}_x\} = \max\{3, 3\} = 3$ robot transfers, with $n = 2$ types of robots, and within at most $\bar{l} = 8$ steps, relative to the delay function *Delay*:

$$f(1, 3, 1) = (3, 1),\ f(1, 4, 1) = (3, 1),\ f(2, 5, 2) = (2, 1).$$

Indeed, f satisfies the conditions (a) and (b) of Definition 1:

(a) For Team 3, Team 1 can lend 1 robot of type 1 at time step 3 (i.e., $f(1, 3, 1) = (3, 1)$). Then, even with the delay of $3 - 1 = 2$ units, Team 3 can finish its task in less than 8 steps:

$$\max\{3 + 2\} = 5 \leq 5 = Borrow\_latest_{1,1}(3).$$

For Team 4, Team 1 can lend 1 robot of type 1 at time step 3 (i.e., $f(1, 4, 1) = (3, 1)$). Then, even with the delay of $4 - 1 = 3$ units, Team 4 can finish its task in less than 8 steps:

$\max\{3+3\} = 6 \leq 6 = Borrow\_latest_{1,1}(4)$.

For Team 5, Team 2 can lend 1 robot of type 2 at time step 2 (i.e., $f(2,5,2) = (2,1)$). Then, even with the delay of $5-2=3$ units, Team 5 can finish its task in less than 8 steps:

$\max\{2+3\} = 5 \leq 7 = Borrow\_latest_{1,2}(5)$.

(b) Team 1 can lend 2 robots of type 1 at time step 3, 1 robot to Team 3 and 1 robot to Team 4:

$f(1,3,1) = (3,1)$, $f(1,4,1) = (3,1)$,

and finish its task in less than 8 steps:

$Lend\_earliest_{2,1}(1) = 3 \leq 3 = \min\{3,3\}$.

Similarly, Team 2 can lend 1 robot of type 2 to Team 5 at time step 2:

$f(2,5,2) = (2,1)$,

and finish its task in less than 8 steps:

$Lend\_earliest_{1,2}(2) = 2 \leq \min\{2\}$.

5.4 Finding a team coordination is hard

As expected, finding a team coordination as described in the previous section is intractable but not harder than the one studied in Erdem et al. (2013) (assuming that P $\neq$ NP):

Proposition 1 *Existence of an $n\overline{m}\overline{l}$-collaboration (i.e., the decision version of* FINDCOLLABORATION_*n) is NP-complete.*

To prove this proposition, we will need the following proposition [essentially, Proposition 1 of Erdem et al. (2013)]:

Proposition 2 *Existence of an $n\overline{m}\overline{l}$-collaboration where $n = 1$ is NP-complete.*

Proof *(Proposition 2)* 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_1 problem instance with a lender teams and b borrower teams with $\overline{l} = 2a$ and $\overline{m}$ 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. □

Proof *(Proposition 1)* We prove the membership as in the proof of Proposition 2. We prove the hardness by a polynomial-time reduction from FINDCOLLABORATION_1, which is an NP-complete problem by Proposition 2: consider one type of robots in FINDCOLLABORATION_*n*. □

5.5 Finding a coordination of teams in ASP

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

We model FINDCOLLABORATION_*n*, subject to conditions C1–C3, in ASP. The input is represented by a set of facts, using atoms of the forms $delay(i,j,l)$, $lend\_earliest(i,m,l,x)$, and $borrow\_latest(j,m,l,x)$ where $1 \leq x \leq n$, $i \in Lenders_x$, $j \in Borrowers_x$, $m \leq \overline{m}$, $l \leq \overline{l}$.

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

$$
\begin{aligned}
&condition\_borrower(j,x) \leftarrow \\
&\quad borrow\_latest(j,m,l,x), \\
&\quad \#sum\{u,i,l_1 : f(i,j,l_1,u,x), \\
&\qquad\qquad i \in Lenders_x, l_1 \leq \overline{l}, u \leq \overline{m}\} \geq m, \\
&\quad \#max\{l_1+t,i,u : f(i,j,l_1,u,x), delay(i,j,t), \\
&\qquad\qquad i \in Lenders_x, l_1 \leq \overline{l}, u \leq \overline{m}\} \leq l
\end{aligned}
$$

where $1 \leq x \leq n$, $j \in Borrowers_x$, $l \leq \overline{l}$, $m \leq \overline{m}$. 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:

$$
\begin{aligned}
&condition\_lender(i,x) \leftarrow \\
&\quad lend\_earliest(i,m,l,x), \\
&\quad \#sum\{u,j,l_1 : f(i,j,l_1,u,x), \\
&\qquad\qquad j \in Borrowers_x, l_1 \leq \overline{l}, u \leq \overline{m}\} \leq m \\
&\quad \#min\{l_1,i,u : f(i,j,l_1,u,x), \\
&\qquad\qquad j \in Borrowers_x, l_1 \leq \overline{l}, u \leq \overline{m}\} \geq l
\end{aligned}
$$

where $1 \leq x \leq n$, $i \in Lenders_x$, $l \leq \overline{l}$, $m \leq \overline{m}$.

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

$$
\begin{aligned}
&\{f(i,j,l,u,x) : l \leq \overline{l}, u \leq \overline{m}\}1 \leftarrow \\
&\quad (1 \leq x \leq n, i \in Lenders_x, j \in Borrowers_x)
\end{aligned}
$$

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

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

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

$$\leftarrow not\ condition\_borrower(j,x), fB(j,x) \quad (j \in Borrowers_x, 1 \le x \le n)$$
$$\leftarrow not\ condition\_lender(i,x) \quad (i \in Lenders_x, 1 \le x \le n).$$

5.6 Decoupling plans for an optimal global plan

Once a coordination of the teams is found for an optimal value of $\bar{l}$, the necessary information of which team lends to (or borrow from) which other team and when is passed to each team. Then each team computes an optimal local plan whose length is less than or equal to $\bar{l}$. The optimal global plan is found by decoupling the optimal local plans.

In this section, we discuss the correctness of our approach.

Proposition 3 *Under C1, C2 and C3, for every $\bar{l}, \overline{m}$ and n, there is an $n\overline{ml}$-collaboration of lenders and borrowers if and only if there is a collaboration of lenders and borrowers.*

Proof ($\Rightarrow$) Assume that there is an $n\overline{ml}$-collaboration of lenders and borrowers under C1, C2 and C3. The $n\overline{ml}$-collaboration tells which lender should lend how many number of robots of which type to which borrower at which step. Since the workspaces of the teams are separate and the teams execute their local plans according to the $n\overline{ml}$-collaboration there is no collision of robots during the execution of the plans. Therefore the $n\overline{ml}$-collaboration is a collaboration of lenders and borrowers.

($\Leftarrow$) Assume that there is a collaboration of lenders and borrowers for some $\bar{l}, \overline{m}$ and n under C1, C2 and C3. Since this collaboration satisfies C2 and C3, it means that lending/borrowing is done in a single batch and the teams lend/borrow robots of the same type.

Let team i be a lender that lends $m_1, \ldots, m_r$ robots of type x to teams $j_1, \ldots, j_r$ at steps $l_1, \ldots, l_r$, respectively. Team i can finish its task in at most $\bar{l}$ steps while it lends $m_1 + \cdots + m_r = m$ robots. So if $Lend\_earliest_{m,x}(i) = l$, then $l \le \min\{l_1, \ldots, l_r\}$ will hold. Similarly, for a borrower team j that borrows $m_1, \ldots, m_r$ robots of type x from teams $i_1, \ldots, i_r$ at steps $l_1, \ldots, l_r$, we can conclude that if $Borrow\_latest_{m,x}(j) = l$ for $m = m_1 + \cdots + m_r$, then $l \ge \max\{l_1, \ldots, l_r\}$. Also if there is delay between teams we can determine it by taking the time difference of the step l' that team i lends robots to team j and the step l'' that team j borrows robots from team i.

Therefore if there is a collaboration then it satisfies the conditions for an $n\overline{ml}$-collaboration. So there is an $n\overline{ml}$-collaboration under C1, C2 and C3. □

By incrementing $\bar{l}$ one by one until k, we can find the optimal value for the plan length. After finding an optimal plan length for the teams, each team computes its local plan regarding the $n\overline{ml}$-collaboration.

Proposition 4 *The union of local plans is a global plan.*

Proof Each team computes its local plan with the information of at which step how many robots they are suppose to lend/borrow, stated by the $n\overline{ml}$-collaboration. Since the constraints in the domain description rules out the actions which would result in collisions, during the execution of the local plans there won't be any conflicts.

By taking the union of the local plans, each team will concurrently perform their actions at each step. At the lending steps, the robots that are lent will move to the bench, which is a separate area from the workspaces. At the borrowing steps, the robots will move from the bench to the borrower's workspace. The delay between two workspaces are also considered, so at the borrowing step for the team, the robots will be available at the bench to enter the workspace.

Since the workspaces of the teams are separate, during the execution of the plans there won't be any conflicts between teams. Therefore the union of the local plans can be determined as the global plan of the teams. □

5.7 Algorithm for finding an optimal global plan

We introduce an algorithm (Algorithm 1) to find an optimal global plan of r teams, given the action domain description $\mathcal{D}$, maximum plan length k, number of types of robots n, maximum number of robots m_x that can be transferred for type x ($x \le n$), and the planning problems $\mathcal{P}_1, \ldots, \mathcal{P}_r$ for each team with initial states and goals.

To find an optimal global plan, first the roles of each team (lender or borrower) are identified by asking queries of the form Q1 (Algorithm 2). After that, by gathering yes/no answers to queries of the forms Q2 and Q3 from every lender/borrower team, the earliest lend times and the latest borrow times are inferred (Algorithms 3 and 4). Then, based on these earliest and latest times of robot transfers, and considering delays of robot transfers, an optimal coordination among the teams is computed using ASP as described in Sect. 5.5. Once such a coordination is found for an optimal global plan of length, necessary information about this coordination is conveyed to each lender or borrower as constraints. After that, each team computes its own optimal local

hybrid plan (as described in Sect. 4) whose length is at most $\bar{l}$, and which satisfies the coordination constraints conveyed to by the mediator. Finally, these optimal local hybrid plans are decoupled for an optimal global plan as described in Sect. 5.6. According to Algorithm 1, linear search is used to find the optimal value of $\bar{l}$.

Algorithm 1 FIND_OPTIMAL_GLOBAL_PLAN

Input: An action domain description $\mathcal{D}$, positive integers k, n and $m_x, x \leq n$, r planning problems $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ (one for each team) with initial states $s_1, s_2, \ldots, s_r$ and goal states $g_1, g_2, \ldots, g_r$, and a transportation delay t_d
Output: A tuple consisting of, for each team i, a plan $P[i]$ of length at most $\bar{l} \leq k$, team role $role[i]$, lending/borrowing constraints for each team $C[i]$

$planfound := false$;
$\bar{l} := 0$;
while $\neg planfound \wedge \bar{l} < k$ **do**
  $\bar{l} = \bar{l} + 1$;
  $role \leftarrow$ DETERMINEROLES($\mathcal{D}, \mathcal{P}_1, \ldots, \mathcal{P}_r, \bar{l}$);
  if all teams are Borrowers **then**
  //There isn't a lender team to help the borrower teams
    continue;
  end if
  if all teams are Lenders **then**
  //Every team can complete on its own
    $planfound = true$;
    continue;
  end if
  for all teams i **do**
    if $role[i] = Lender$ **then**
      $lend_{m,x}[i] \leftarrow$
        GATHERANSWERS_LEND($\mathcal{D}, \mathcal{P}_i, \bar{l}, n, m_1, \ldots, m_n$);
    else
      $borrow_{m,x}[i] \leftarrow$
        GATHERANSWERS_BORROW($\mathcal{D}, \mathcal{P}_i, \bar{l}, n, m_1, \ldots, m_n$);
    end if
  end for
  If FIND_COORDINATION($role, \bar{l}, n, m_1, ..., m_n,$
      $lend_{m,x}, borrow_{m,x}, t_d$) **then**
    $C \leftarrow$ determine constraints from the coordination;
    $planfound = true$;
end while
for all teams i **do**
  $P[i] \leftarrow$ FIND_LOCAL_PLAN($\mathcal{D}_i, \bar{l}, \mathcal{P}_i, C_i$);
end for

In Algorithm 1, the function DETERMINEROLES (described in Algorithm 2) determines the role of each team for length $\bar{l}$ given the action domain description $\mathcal{D}$ and the planning problems of each team. This function asks each team i for a plan of length $\bar{l}$ with domain $\mathcal{D}$ and planning problem $\mathcal{P}_i$. If the team can find a plan, then there is a possibility that the team may be able to lend some of its robots and still reach its goal state in $\bar{l}$ steps, so it is identified as a lender. Otherwise, it is identified as a borrower.

The function GATHERANSWERS_LEND (described in Algorithm 3) asks lender teams queries of the form Q2 for every $x \leq n, m \leq m_x$ and $l \leq \bar{l}$, by updating the domain description $\mathcal{D}$ with $\mathcal{D}_{lend,x}$ which is a description of the lending action and additional constraints on only allowing the teams to lend robots of type x only. The constraints are given to avoid undesired lending actions of other type of robots. After collecting the answers, the earliest lend time for each lender team (i.e., the minimum time step l that the team can lend robots, while completing its tasks within $\bar{l}$ steps) is found.

Algorithm 2 DETERMINEROLES

Input: An action domain description $\mathcal{D}$, r planning problems $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$ (one for each team), positive integer $\bar{l}$
Output: team roles, $role$

for all teams i **do**
  Ask for a plan of length $\bar{l}$ with domain $\mathcal{D}$ and planning problem $\mathcal{P}_i$
  if answer=success **then**
    $role[i] = Lender$;
  else
    $role[i] = Borrower$;
  end if
end for

Similarly, for borrowers, GATHERANSWERS_BORROW (described in Algorithm 4) asks queries of the form Q3 for every $x \leq n, m \leq m_x$. This function updates the domain description $\mathcal{D}$ now with $\mathcal{D}_{borrow,x}$ which is a description of the borrowing action and additional constraints on only allowing the teams to borrow robots of type x only. The constraints are to avoid undesired borrowing actions of other type of robots. After the answers are collected, the latest borrow time for each borrower team (i.e., the maximum time step l at which the team borrows robots, to complete its tasks within $\bar{l}$ steps) is found.

Algorithm 3 GATHERANSWERS_LEND

Input: An action domain description $\mathcal{D}$, planning problem $\mathcal{P}$, positive integers $\bar{l}, n$ and $m_x, x \leq n$
Output: $lend_{m,x}$ for each $m \leq m_x, x \leq n$

$L_{1,1}, \ldots, L_{m_1,1}, \ldots, L_{1,n} \ldots, L_{m_n,n} \leftarrow$ empty sets for lend times of each number of robot type;
for all robot types x ≤ n **do**
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{lend,x}$;
  for all m ≤ m_x **do**
    for all l ≤ $\bar{l}$ **do**
      $C \leftarrow$ the constraint of lending m robots of type x before step l;
      Ask for a plan of length $\bar{l}$ with domain $\mathcal{D}$ and planning problem $\mathcal{P} \cup C$;
      if answer=success **then**
        $L_{m,x} = L_{m,x} \cup \{l\}$;
      end if
    end for
    $lend_{m,x} = min\{L_{m,x}\}$;
  end for
end for

Once the answers are collected and the earliest lend times and latest borrow times are identified, the function FIND_COORDINATION tries to find an $nm\bar{l}$-collaboration. If a coordination of the teams is found, the constraints of each

Algorithm 4 GATHERANSWERS_BORROW

Input: An action domain description $\mathcal{D}$, planning problem $\mathcal{P}$, positive integers $\bar{l}$, n and $m_x, x \leq n$

Output: $borrow_{m,x}$ for each $m \leq m_x, x \leq n$

$B_{1,1}, \ldots, B_{m_1,1}, \ldots, B_{1,n} \ldots, B_{m_n,n} \leftarrow$ empty sets for borrow times of each number of robot type;

for all robot types x ≤ n **do**
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{borrow,x}$;
  for all m ≤ m_x **do**
    for all l ≤ $\bar{l}$ **do**
      $C \leftarrow$ the constraint of borrowing m robots of type x after step l;
      Ask for a plan of length $\bar{l}$ with domain $\mathcal{D}$ and planning problem $\mathcal{P} \cup C$;
      if answer=success **then**
        $B_{m,x} = B_{m,x} \cup \{l\}$;
      end if
    end for
    $borrow_{m,x} = max\{B_{m,x}\}$;
  end for
end for

team describing how many robots of which type they are supposed to lend to (or borrow from) which team at which step are determined. To find local plans of each team, first, the domain descriptions are updated according to the constraints of the team. If a team has constraints for lending or borrowing robots of type x, the domain description $\mathcal{D}$ is updated with $\mathcal{D}_{lend,x}$ or $\mathcal{D}_{borrow,x}$, respectively. Then, the teams are asked for a local plan given this constraints from the coordination. The plans gathered from the teams forms the optimal global plan.

6 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 Fig. 2, 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 these 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 Erdem et al. (2012b), 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 Sect. 4. 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 Sect. 5, 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.

Fig. 2 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

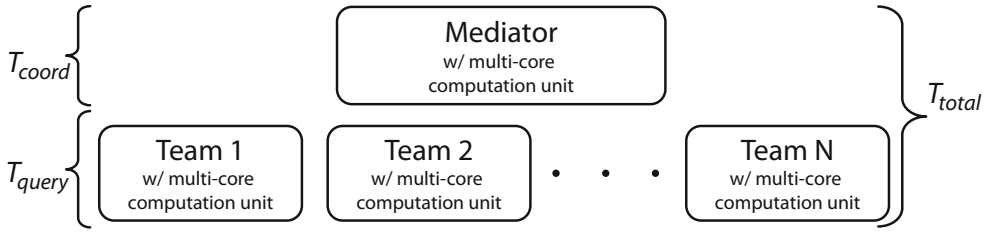

Fig. 3 Parallelization of queries

We have tested this cognitive factory with dynamic simulations (with all three different optimizations of local feasible plans) using OpenRave (Diankov 2010), 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 Fig. 2. 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.

7 Experimental evaluation

We have investigated the scalability and usefulness of the proposed optimal global planning method by means of some experiments over the cognitive toy factory domain.

7.1 Setting

We have performed our experiments on a Linux server with 16 Intel E5-2665 CPU cores (32 threads) with 2.4GHz and 64GB memory. Our experiments never used more than 300MB.

Algorithms 1, 2, 3 and 4 are implemented in Python. The ASP solver Clasp version 2.1.3 (with gringo version 3.0.5), with `configuration=handy` as the command line option, is used for answering queries. To solve the collaboration problem, we also use Clasp.

First of all, it is important to observe that, since each team is equipped with its own computation unit and the queries are designed in a way that teams do not need to communicate with each other, queries for teams can be trivially parallelized as depicted in Fig. 3. 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. 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 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.

7.2 T_{query}: querying teams

To study the scalability of querying teams, we have performed experiments both for homogeneous teams and for heterogeneous teams. In the homogeneous case, we have considered two sets of scenarios where workspaces are 7×3 grid cells, as shown in Table 1.

The team sizes in these scenarios 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. In the first set of scenarios, each team has a different size. Scenarios 1–3 (resp. Scenarios 4 and 5) are built incrementally. For example, in Scenario 2 there are three teams with 1, 2, 4 worker robots, respectively, and the total number of robots in this scenario, including the charger robots, is 11, whereas in Scenario 3 an additional team with 6 worker robots and 3 charger robots is added. In the second set of scenarios, some teams have the same size. Here also Scenarios 1_b–3_b (resp. Scenarios 4_b –6_b) are built incrementally, but considering teams of same sizes. For instance, in Scenario 1_b there are three teams with 1, 1, 2 worker robots, whereas in Scenario 2_b there are four teams with 1, 1, 2, 2 worker robots, the numbers of teams are increased by adding new teams of the same size.

We have also considered the heterogeneous cases ($n = 2$) of these scenarios, where the number of a worker robot of one type is computed by dividing the total number of workers by n. For example, the heterogeneous case of Scenario 4 is two teams with 1 wet, 1 dry worker robots and 2 wet, 2 dry worker robots, respectively.

Table 1 Scenarios

Scenario	Teams	Worker robots		Charger robots	Total robots
		Homogeneous	Heterogeneous (w:wet, d:dry)		
1	2	1, 2	1 w, 1 w 1 d	1, 1	5
2	3	1, 2, 4	1 w, 1 w 1 d, 2 w 2 d	1, 1, 2	11
3	4	1, 2, 4, 6	1 w, 1 w 1 d, 2 w 2 d, 3 w 3 d	1, 1, 2, 3	20
4	2	2, 4	1 w 1 d, 2 w 2 d	1, 2	9
5	3	2, 4, 6	1 w 1 d, 2 w 2 d, 3 w 3 d	1, 2, 3	18
1_b	3	1, 1, 2	1 w, 1 w, 1 w 1 d	1, 1, 1	7
2_b	4	1, 1, 2, 2	1 w, 1 w, 1 w 1 d, 1 w 1 d	1, 1, 1, 1	10
3_b	5	1, 1, 2, 2, 2	1 w, 1 w, 1 w 1 d, 1 w 1 d, 1 w 1 d	1, 1, 1, 1, 1	13
4_b	3	2, 2, 4	1 w 1 d, 1 w 1 d, 2 w 2 d	1, 1, 2	12
5_b	4	2, 2, 4, 4	1 w 1 d, 1 w 1 d, 2 w 2 d, 2 w 2 d	1, 1, 2, 2	18
6_b	5	2, 2, 4, 4, 4	1 w 1 d, 1 w 1 d, 2 w 2 d, 2 w 2 d, 2 w 2 d	1, 1, 2, 2, 2	24

In these experiments, in order to find the optimal global plan length for the teams in each scenario, we have considered queries of the form Q1–Q3 as described in Sect. 5.1. Total number of queries in each scenario corresponds to the total number of queries answered by all the teams until a coordination for an optimal length is found. The computation time T_{query} in each scenario is computed by the time it takes for the slowest team to answer all the queries.

We have analyzed the results to better understand how the computation time (CPU seconds) for answering queries and the optimal global plan length are affected by a change in the team size, the number of teams, the number of orders, the maximum number of robot exchange between teams, and when heterogeneous robots with varying capabilities are considered. Although the experimental evaluation of different sorts of hybrid reasoning has been studied in a companion paper (Erdem et al. 2016b), we have also conducted some experiments to observe the effect of hybrid reasoning.

7.2.1 Changing the maximum number of robot transfers

For comparing the effect of the maximum number of robot exchanges between teams, we have considered scenarios in which there are slightly larger teams, Scenarios 3 and 5, and Scenarios 1_b and 4_b, since we want the possibility of exchanging more robots between teams. The results of query answering are shown in Table 2.

We can observe from Table 2 that, as more number of robots are allowed to be exchanged between teams, the teams become more collaborative and this results in shorter optimal global plans. For example, in Scenario 3, when the maximum number of robot exchanges between teams is limited to 1, the optimal global plan is found to be of length 30 and the computation (T_{query}) takes 795 s (with homogeneous robots). When the maximum number of robot exchanges is set to 2, the optimal global plan length decreases to 20 and the computation time decreases to 30.6 s. In Scenario 5, the optimal global plan length continues to decrease when a maximum of 3 robot transfers are allowed. We can also observe that most of the computation time is spent for negative answers. Similarly, in Scenarios 1_b and 4_b, increasing the maximum number of robot transfers shortens the plan lengths, while the computation times increase slightly.

7.2.2 Changing the workspace sizes and the team sizes

To analyze the effects of changing the workspace size (i.e., number of grid cells) and the team size (i.e., the total number of robots in each team) on the computational efficiency (T_{query}) and the quality of plans (i.e., makespans), we have performed some experiments.

First, to understand the role of workspace size, we have generated two instances for each of Scenarios 2 and 4 of Table 1, with different sizes of workspaces (with $5 \times 3 = 15$ and $7 \times 3 = 21$ grid cells). In these instances, the maximum makespan is $k = 50$. Table 3 shows the results of query answering. Consider the first instance of Scenario 2 where the maximum number of robot exchanges is 2 (m =2), the size of the order is 4, and the workspace consists of 15 cells. The optimal global plan length for this instance is 24. In this instance, the team of two homogeneous robots answers 93 queries (21 of them positively and 72 of them negatively) in 9.17 (CPU) seconds (2.55 s for positive answers, 6.92 s for negative answers), under multi-threading. It takes 45.0 s for the team of 3 robots to answer 94 queries, whereas it takes 47.4 s for the team of 6 robots to answer 95 queries. Therefore, query answering for an optimal global plan in this instance takes 47.4 s. Now, consider the second instance of Scenario 2, where the workspace size is increased from 15 cells to 21 cells. The optimal global plan length increases to

Table 2 Robot transfers versus computation time and plan quality: multiple teams, where workspaces are of the same size 7×3 and the maximum makespan is 50

Scenario	Worker robots #	Charger robots #	Order #	Max. robot exchange #	Homogeneous T_{query} sec	Heterogeneous T_{query} sec	Optimal plan length
3	1, 2, 4, 6	1, 1, 2, 3	5	1	795 (51.2 + 744)	245 (17.4 + 228)	30
				2	30.6 (28.7 + 1.92)	34.7 (32.9 + 1.79)	20
5	2, 4, 6	1, 2, 3	6	1	179 (12.2 + 167)	136 (18.4 + 118)	25
				2	640 (11.9 + 628)	263 (9.48 + 253)	23
				3	261 (9.31 + 251)	174 (11.8 + 163)	21
1_b	1, 1, 2	1, 1, 1	4	1	250 (117 + 132)	140 (68.6 + 71.2)	39
				2	390 (205 + 184)	137 (67.0 + 70.1)	35\39*
4_b	2, 2, 4	1, 1, 2	4	1	41.2 (11.5 + 29.7)	34.6 (34.1 + 0.52)	25
				2	79.1 (35.3 + 43.7)	57.8 (39.2 + 18.6)	23

*Optimal plan lengths are different for the homogeneous case and the heterogeneous case

Table 3 Team size and workspace size versus computation time and plan quality for homogeneous robot teams

	Number of robots	Total grid cells	Optimal plan length	Total queries	Time to answer all queries (secs)
Scenario 2	2	15	24	93 (21 + 72)	9.17 (2.55 + 6.92)
m = 2	3			94 (39 + 55)	45.0 (13.0 + 32.0)
Order = 4	6			95 (74 + 21)	47.4 (24.2 + 23.2)
Scenario 2	2	21	25	86 (20 + 66)	20.9 (4.71 + 16.2)
m = 2	3			87 (39 + 48)	105 (23.9 + 80.6)
Order = 4	6			85 (71 + 14)	102 (57.2 + 44.4)
Scenario 4	3	15	19	29 (5 + 24)	2.78 (0.65 + 3.98)
m = 2	6			29 (20 + 9)	6.09 (3.98 + 2.11)
Order = 5					
Scenario 4	3	21	20	22 (5 + 17)	5.84 (1.75 + 4.08)
m = 2	6			22 (14 + 8)	15.9 (8.51 + 7.40)
Order = 5					

25. Here, the team of 2 robots answers 86 queries 20.9 s, the team of 3 robots answers 87 queries 105 s, the team of 6 robots answers 85 queries 102 s. Therefore, query answering for an optimal global plan in this instance (with a larger workspace) takes 105 s. There are several interesting observations over these results: as the workspace increases, the optimal global plan length increases (since the teams need to navigate a bit more), the number of queries, in particular, the number of queries that are answered negatively decreases (since the teams can now navigate more comfortably, complete their tasks, and thus are more willing to help each other), and the computation time generally increases (since the problem size increases). There is another interesting observation if we look at each team's query answering in each scenario instance: the larger teams take more time to answer about the same number of queries compared to smaller teams. This is due to that larger the team larger the search space for each query.

To better understand the role of team size and robot transfers, we have generated four instances for each of Scenarios 1_b, 2_b, 4_b and 5_b of Table 1, with 1 or 2 maximum robot transfers, homogeneous or heterogeneous . The team sizes also vary in these scenarios. In Scenarios 1_b and 4_b there are three teams, but Scenario 4_b doubles the number of worker robots in the teams. Similarly, Scenarios 2_b and 5_b have four teams, but Scenario 5_b doubles the number of worker robots and the number of charger robots in the teams. Table 4 shows the results of query answering in these scenarios. For instance, consider the case where the maximum number of robot transfers is 1. Under these conditions, for the

Table 4 Team size and robot transfers versus computation time and plan quality: workspace size is 7×3, the order number is 4, and the maximum makespan is 50

Scenario	Max. robot exchange	Worker robots	Charger robots	Homogeneous		Heterogeneous		Optimal plan
	#	#	#	Total queries #	Time sec	Total queries #	Time sec	length
1_b	1	1*	1	96 (41 + 55)	32.1 (16.5 + 15.5)	167 (81 + 86)	58.6 (30.9 + 27.7)	39
		2	1	95 (49 + 46)	250 (118 + 132)	168 (78 + 90)	140 (68.6 + 71.1)	
	2	1*	1	119 (45 + 74)	36.6 (14.3 + 22.3)	271 (118 + 153)	112 (47.1 + 65.0)	35\ 39**
		2	1	115 (56 + 59)	390 (205 + 185)	168 (78 + 90)	137 (67.0 + 70.1)	
4_b	1	2*	1	48 (22 + 26)	41.2 (11.5 + 29.7)	85 (43 + 42)	25.6 (9.49 + 16.1)	25
		4	2	43 (41 + 2)	25.5 (24.9 + 0.59)	75 (73 + 2)	34.6 (34.1 + 0.52)	
	2	2*	1	70 (33 + 37)	36.2 (9.90 + 26.3)	131 (66 + 65)	33.7 (13.3 + 20.4)	23
		4	2	67 (53 + 14)	79.1 (35.3 + 43.7)	127 (92 + 35)	57.8 (39.2 + 18.6)	
2_b	2	1*	1	69 (23 + 46)	13.7 (5.71 + 7.98)	122 (45 + 77)	30.9 (12.0 + 18.0)	30
		2*	1	63 (26 + 37)	239 (112 + 127)	75 (25 + 50)	92.2 (37.8 + 54.4)	
	3	1*	1	93 (31 + 62)	25.1 (9.42 + 15.7)	170 (60 + 110)	50.4 (17.2 + 33.2)	30
		2*	1	63 (26 + 37)	252 (118 + 134)	75 (25 + 50)	85.6 (35.1 + 50.5)	
5_b	2	2*	1	11 (4 + 7)	1.37 (0.68 + 0.68)	19 (8 + 11)	2.25 (1.13 + 1.11)	17
		4*	2	11 (7 + 4)	4.14 (2.5 + 1.63)	19 (12 + 7)	5.67 (3.59 + 2.08)	
	3	2*	1	15 (5 + 10)	2.84 (1.12 + 1.72)	27 (10 + 17)	5.32 (2.10 + 3.23)	17
		4*	2	14 (9 + 5)	6.8 (4.11 + 2.69)	19 (12 + 7)	6.24 (3.99 + 2.54)	

*Since teams with same size give similar results, only one representative team is presented
**Optimal plan lengths are different for the homogeneous case and the heterogeneous case

homogeneous case, the optimal plan length decreases from 39 in Scenario 1_b to 25 in Scenario 4_b, since larger teams are able to finish their tasks quickly. This results in less number of queries and the computation time becomes smaller: it takes at most 250 s to answer 95 queries in Scenario 1_b, whereas it takes at most 41.2 s to answer 48 queries in Scenario 4_b. We can observe also in Scenarios 2_b and 5_b that as the team size increases the optimal plan length decreases, since there are more robots that can be used for the tasks. Similar observations hold for the heterogenous instances.

Now consider the case where the maximum number of robot transfers is 2: so the teams can help each other even more. Since the number of queries increases due to the increase of maximum robot transfers, the computation time increases in all of the scenarios. Also, the increase in the number of robot transfer leads the teams to help each other more. In Scenarios 1_b and 4_b, for instance, we can see it affecting the plan quality: for the homogenous case, the plan lengths decrease from 39 to 35 in Scenario 1_b, and from 25 to 23 in Scenario 4_b.

Additionally, it can be observed that the total number of queries increase when heterogeneity is considered. In many cases, this increase also leads to an increase in computation time. Meanwhile, the increase in the number of worker robots decreases the total querying time (i.e., the slowest querying time among the teams), since it becomes easier to answer the queries when there are robots with restricted capabilities. If it is known that a robot cannot do a certain task, there is no need to look further and this decreases the computation time.

7.2.3 Changing the number of teams

To analyze the effects of changing the number of teams on the computational efficiency (T_{query}) and the quality of plans (i.e., makespans), we have considered three sets of scenarios, Scenarios 2, 4 and 5, Scenarios 1_b, 2_b and 3_b, and Scenarios 4_b, 5_b and 6_b of Table 1. These scenarios have different number of teams but same values for other parameters such as the maximum number of robot exchange or the number of orders. The results of query answering in these scenarios are shown in Table 5.

Consider Scenarios 2, 4 and 5: Scenario 4 has 2 teams, Scenario 2 has 3 teams with an additional team having a smaller size than the other teams, and Scenario 5 has 3 teams with an additional team having a larger size than the other teams. We can observe from Table 5 that both the computation time (T_{query}) and the optimal plan length increase when a team of a smaller size is included. For example, an optimal global

Table 5 Number of teams versus computation time and plan quality: multiple teams with different size, where workspaces are of the same size 7×3, the maximum number of robot transfers is 2, and the maximum makespan is 50

Scenario	Team size	Order	Homogeneous		Heterogeneous		Optimal plan
	#	#	Total queries #	Time sec	Total queries #	Time sec	length
4	3	4	11 (4 + 7)	1.47 (0.72 + 0.75)	19 (8 + 11)	2.70 (1.35 + 1.35)	17
2 teams	6	4	11 (7 + 4)	4.51 (2.71 + 1.80)	19 (12 + 7)	6.98 (4.53 + 2.45)	
2	2	4	86 (20 + 66)	20.6 (4.68 + 15.9)	162 (41 + 121)	36.1 (9.08 + 27.0)	25
3 teams	3	4	87 (39 + 48)	106 (24.6 + 81.4)	159 (75 + 84)	62.7 (21.1 + 41.6)	
	6	4	85 (71 + 14)	102 (57.5 + 44.9)	163 (122 + 41)	87.0 (64.0 + 22.9)	
5	3	4	27 (4 + 23)	3.92 (0.69 + 3.23)	51 (8 + 43)	7.23 (1.23 + 6.01)	17
3 teams	6	4	11 (7 + 4)	4.31 (2.62 + 1.69)	19 (12 + 7)	5.78 (3.67 + 2.11)	
	9	4	11 (11 + 0)	13.5 (13.5 + 0)	19 (19 + 0)	19.7 (19.7 + 0)	
1_b	2	3	99 (38 + 61)	12.2 (4.71 + 7.48)	183 (78 + 105)	23.8 (9.59 + 14.2)	30\31**
3 teams*	3	3	100 (51 + 49)	33.4 (16.1 + 17.3)	112 (52 + 60)	15.2 (7.45 + 7.78)	
2_b	2	3	44 (14 + 30)	3.94 (1.56 + 2.39)	78 (29 + 49)	7.91 (3.15 + 4.75)	24
4 teams*	3	3	39 (16 + 23)	12.8 (5.26 + 7.51)	45 (19 + 26)	7.45 (3.56 + 3.89)	
3_b	2	3	44 (14 + 30)	4.07 (1.60 + 2.47)	78 (29 + 49)	8.01 (3.22 + 4.79)	24
5 teams*	3	3	39 (16 + 23)	12.7 (5.27 + 7.49)	45 (19 + 26)	7.55 (3.58 + 3.98)	
4_b	3	5	71 (30 + 41)	71.4 (18.7 + 52.7)	150 (68 + 82)	98.2 (34.6 + 63.6)	25\ 26**
3 teams*	6	5	70 (49 + 21)	246 (58.8 + 187)	150 (103 + 47)	156 (83.8 + 72.5)	
5_b	3	5	22 (5 + 17)	5.93 (1.75 + 4.18)	38 (10 + 28)	8.40 (3.13 + 5.26)	20
4 teams*	6	5	22 (14 + 8)	16.5 (8.87 + 7.61)	39 (25 + 14)	18.1 (10.6 + 7.50)	
6_b	3	5	22 (5 + 17)	6.63 (2.00 + 4.63)	38 (10 + 28)	9.08 (3.40 + 5.69)	20
5 teams*	6	5	22 (14 + 8)	17.8 (9.46 + 8.31)	39 (25 + 14)	20.4 (12.0 + 8.40)	

*Since teams with same size give similar results, only one representative team is presented
**Optimal plan lengths are different for the homogeneous case and the heterogeneous case

plan length for Scenario 4 is 17, whereas for Scenario 2 it is 25. It takes 4.51 s for the teams to answer all queries in Scenario 4, and 106 s for Scenario 2, when the robots are homogeneous. Such an increase in the optimal plan length and the total querying time is due to that a longer plan is needed for the smaller team to complete its task. We can also observe that, when a larger team is included (as in Scenario 5), the querying time increases slightly (with respect to Scenario 4) even when the optimal plan length remains the same. This increase is due to more computation time spent for queries about a larger team.

Table 5 also shows the results of increasing the number of teams by adding teams of the same size. We can observe that there can be a decrease in the optimal global plan length when the additional team is not of a smaller size, and that it can be used to help the other teams to reach a shorter plan. For example, in Scenarios 1_b and 2_b, there is a decrease in the plan length from 30 to 24. This also results in the decrease of total number of queries and hence the decrease in the computation times (T_{query}): we can observe that it takes at most 33.4 s to complete query answering for Scenario 1_b, whereas it takes at most 12.8 s for Scenario 2_b, when the robots are homogeneous. However, there is not always a decrease in the optimal global plan length, as we can observe from Scenario 3_b, since a coordination for a shorter length of plan may not be found in the setting. Also the computation time stays the same, since the additional team has the same size, and it does not take a longer time for the team to answer all the queries. Similar results can be observed for Scenarios 4_b, 5_b and 6_b which have larger sized teams than in Scenarios 1_b, 2_b and 3_b.

Additionally, it can be observed that the optimal global plan length increases in certain cases when heterogeneity is considered, since the existence of robots with restricted capabilities may cause delays for the accomplishment of some tasks (check for instance Scenario 1_b, when the teams can help each other by borrowing at most 1 robot or at most 2 robots).

Table 6 Order numbers versus computation time and plan quality: multiple teams, where workspaces are of the same size 7×3 and the maximum makespan is 50

Scenario	Worker robots #	Charger robots #	Max. robot exchange #	Order #	Homogeneous T_{query} sec	Heterogeneous T_{query} sec	Optimal plan length
1	1,2	1,1	1	3	8.50 (3.98 + 4.52)	7.57 (3.60 + 3.97)	24
				4	166 (62.9 + 103)	91.4 (37.6 + 53.8)	30
				5	5233 (1878 + 3354)	1141 (455 + 685)	36
2	1,2,4	1,1,2	2	3	36.4 (23.7 + 12.6)	41.0 (31.9 + 9.04)	21
				4	106 (24.6 + 81.8)	85.3 (62.4 + 22.9)	25
				5	1892 (241 + 1650)	534 (159 + 374)	30
3	1,2,4,6	1,1,2,3	3	3	13.3 (13.3 + 0)	20.8 (19.5 + 1.33)	15
				4	19.0 (19.0 + 0)	29.5 (27.9 + 1.63)	17
				5	61.8 (59.9 + 1.90)	84.0 (77.2 + 6.81)	20

Table 7 Order numbers versus computation time and plan quality: multiple teams (using hybrid reasoning), where workspaces are of the same size 7×3 and the maximum makespan is 50

Scenario	Worker robots #	Charger robots #	Max. robot exchange #	Order #	Homogeneous T_{query} sec	Heterogeneous T_{query} sec	Optimal plan length
1	1,2	1,1	1	3	26.6 (10.5 + 16.0)	15.3 (6.82 + 8.51)	24
				4	990 (234 + 756)	313 (113 + 200)	31
2	1,2,4	1,1,2	2	3	69.9 (51.0 + 18.8)	73.7 (57.0 + 16.7)	22
				4	317 (131 + 186)	158 (119 + 38.3)	26
3	1,2,4,6	1,1,2,3	3	3	22.4 (22.4 + 0)	45.5 (32.5 + 12.9)	15
				4	39.9 (38.5 + 1.36)	108 (71.8 + 35.9)	17

7.2.4 Changing the number of orders

To analyze the effects of changing the number of orders/toys on the computational efficiency and the quality of plans, we have considered the scenarios in Table 1. Since the results are similar in two sets of scenarios, Table 6 shows the results of query answering for Scenarios 1–3.

We can observe from Table 6 that, for teams with smaller sizes, both the total time of query answering (T_{query}) and the optimal global plan length increase considerably as the size of the order increases. For example, in Scenario 1, the computation takes 8.50 s when the order is 3 toys, whereas it takes 5233 s when the order is 5 toys. This is due to the increase in work load for each team, and that it takes more time to find a plan utilizing a small number of robots. Also it can be observed that the optimal plan length increases in considerable amount, even if only one more box is added to the order of the teams: the optimal plan length is 24 when the order is 3 toys, whereas the plan length is 30 when the order is 4.

When there are larger teams in a scenario, the increase in the computation time becomes less severe. For example, in Scenario 2, the computation takes 36.4 s when the order is 3 toys, whereas it takes 1892 s when the order is 5 toys. Also, as there are more teams that can help the smaller teams in Scenario 2, the optimal global plan length is found to be 21. As expected, the increase in the number of toys increases the optimal global plan length. Including larger teams, as in Scenario 3, results in shorter plan lengths, and smaller computation times, since queries ask for shorter plans.

Lastly, we can observe from Table 6 that larger teams are able to react more efficiently to new orders. For instance, in Scenario 3, the optimal global plan length increases by 5 steps when 2 more toys are added to the order; in Scenario 1, the optimal plan length increases by 8 steps.

7.2.5 Usefulness of hybrid reasoning

To investigate the usefulness of hybrid reasoning for query answering, we have performed some experiments consider-

Table 8 Hybrid reasoning versus computation time and plan quality for answering Q1: one team, where workspaces are of the same size 7×3 and the maximum makespan is 35

Team size	Order	W/o hybrid			W/ hybrid		
	#	Plan length ($\bar{l}$) #	Answer	Time sec	Plan length ($\bar{l}$) #	Answer	Time sec
2 w, 1 ch	3	21	y	0.19	22	y	0.44
		20	n	0.46	21	n	0.86
	4	25	y	2.46	26	y	2.72
		24	n	2.42	25	n	7.71
	5	30	y	8.39	31	y	72.1
		29	n	61.3	30	n	155
4 w, 2 ch	3	15	y	0.22	15	y	0.32
		14	n	0.21	14	n	0.3
	4	17	y	0.33	17	y	0.66
		16	n	0.32	16	n	0.39
	5	19	y	0.57	19	y	0.75
		18	n	0.7	18	n	0.8
6 w, 3 ch	3	14	y	0.69	14	y	1.13
		13	n	0.6	13	n	0.9
	4	15	y	0.79	16	y	2.16
		14	n	0.67	15	n	1.3
	5	17	y	1.04	17	y	16.5 (6)*
		16	n	0.88	16	n	1.15

*The number in the parentheses denotes the number of replanning until a feasible plan is found

ing Scenarios 1–3 of Table 1. Table 7 shows the results of these experiments.

We can make the following observations from Table 7 (with hybrid reasoning) and Table 6 (without hybrid reasoning). The computation time increases in all scenarios since finding a feasible plan is time consuming. For instance, for Scenario 1 with heterogeneous robots where the number of orders is 3, it takes 7.57 s without collision checks and 15.7 s with collision checks. Also, we can observe an increase in the optimal plan length for some scenarios with hybrid reasoning, since the optimal global plan found without hybrid reasoning may not feasible. For example, Scenario 1 with 4 orders has an optimal plan length of 30 without hybrid reasoning whereas the plan length becomes 31 when hybrid reasoning is included.

We have also conducted some experiments to further study the effect of hybrid reasoning on different sorts of queries. In particular, we have considered queries of type Q1, to find the minimum plan length ($\bar{l}$) for a team to complete its task, and of type Q2, to find the earliest lending time (l) of a team for the plan length $\bar{l}$ found by query Q1. Tables 8 and 9 show the results of these experiments for querying a team. In each table, we can make the following observations. First, the teams with smaller sizes are able to complete their task within a longer plan, whereas the teams with larger sizes can effectively utilize collaborations and complete their task within a shorter plan. For teams with smaller sizes, the computation time for answering a query negatively usually takes longer than answering a query positively. For example, consider the team with 2 workers and 1 charger with order 5. Without hybrid reasoning, it can find a plan of length 30 in 8.39 s, whereas it takes 61.3 s to find that no plan exists for length 29. This difference also exists for the case with hybrid reasoning: The team finds a plan of length 31 in 72.1 s, whereas it takes 155 s to find that no plan exists for length 30. These observations are due to that a negative answer requires checking a larger search space. As also observed in Table 7, hybrid reasoning slightly increases the computation time, and sometimes the optimal plan length. Apart from these common observations from Tables 8 and 9, we can observe that the increase in computation time due to hybrid reasoning is larger for Q2 for many instances; this may be due to the additional constraints considered in Q2.

7.3 T_{coord}: coordination of teams

To study the scalability of our method for finding a coordination of teams, 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 and 16) and the maximum number of robots that can be transferred ($\overline{m} = 2, 4$).

Table 9 Hybrid reasoning versus computation time and plan quality for answering Q2: one team, workspaces are of the same size 7×3 and the maximum makespan is 35

Team size	Order	Robot exchange	W/o hybrid				W/hybrid			
	#	#	Plan length ($\bar{l}$) #	l #	Answer	Time sec	Plan length ($\bar{l}$) #	l #	Answer	Time sec
2 w, 1 ch	3	1	21	20	y	0.49	22	22	y	0.66
				19	n	0.92		21	n	2.86
		2	21	21	y	0.28	22	–	–	–
				20	n	0.43		22	n	2.38
	4	1	25	24	y	5.26	26	25	y	15.9
				23	n	9.06		24	n	37.3
		2	25	25	y	2.81	26	26	y	7.36
				24	n	5.66		25	n	15.8
	5	1	30	30	y	13.9	31	31	y	245
				29	n	206		30	n	t/o
		2	30	–	–	–	31	–	–	–
				30	n	153		31	n	349
4 w, 2 ch	3	1	15	1	y	0.27	15	1	y	0.47
				–	–	–		–	–	–
		2	15	10	y	0.33	15	13	y	0.52
				9	n	0.31		12	n	0.64
		3	15	13	y	0.35	15	15	y	0.51
				12	n	0.39		14	n	0.71
		4	15	14	y	0.28	15	–	–	–
				13	n	0.3		15	n	0.65
	4	1	17	1	y	0.33	17	1	y	0.43
				–	–	–		–	–	–
		2	17	12	y	0.53	17	15	y	0.78
				11	n	0.68		14	n	1.7
		3	17	15	y	0.77	17	17	y	2.57 (4)*
				14	n	0.87		16	n	1.64
		4	17	16	y	0.42	17	–	–	–
				15	n	0.56		17	n	1.68
	5	1	19	1	y	0.59	19	1	y	0.52
				–	–	–		–	–	–
		2	19	13	y	1.3	19	17	y	2.27
				12	n	1.3		16	n	7.37
		3	19	17	y	2.21	19	19	y	3.48 (2)*
				16	n	4.47		18	n	5.85
		4	19	18	y	1.21	19	–	–	–
				17	n	1.34		19	n	5.23

Table 9 continued

Team size	Order	Robot exchange	W/o hybrid				W/hybrid			
	#	#	Plan length ($\bar{l}$) #	l #	Answer	Time sec	Plan length ($\bar{l}$) #	l #	Answer	Time sec
6 w, 3 ch	3	1	14	1	y	0.84	14	9	y	4.76 (3)*
				–	–	–		8	n	2.56
		2	14	9	y	0.94	14	12	y	13.9 (6)*
				8	n	1.08		11	n	4.25
		3	14	10	y	0.9	14	14	y	1.84
				9	n	1.2		13	n	6.84
		4	14	11	y	0.86	14	–	–	–
				10	n	1.13		14	n	8.27
	4	1	15	11	y	0.95	16	9	y	2.61
				10	n	1.37		8	n	11.3
		2	15	12	y	1.03	16	12	y	8.91
				11	n	1.63		11	n	16.2
		3	15	13	y	1.00	16	15	y	6.81
				12	n	2.86		14	n	134
		4	15	13	y	1.00	16	16	y	4.30
				12	n	1.35		15	n	112
	5	1	17	10	y	1.50	17	–	–	–
				9	n	7.00		17	n	16.7
		2	17	12	y	1.61	17	–	–	–
				11	n	7.86		17	n	15.2
		3	17	13	y	2.44	17	–	–	–
				12	n	3.28		17	n	19.3
		4	17	14	y	1.97	17	–	–	–
				13	n	6.22		17	n	20.4

*The number in the parentheses denotes the number of replanning until a feasible plan is found

Table 10 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

Table 10 shows the results of coordination; each reported CPU time is the average for at least $\overline{m} \times n$ 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 s. 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.

7.4 Usefulness of collaborations of teams

Our approach for finding an optimal global plan allows for collaborations of teams. To better understand the usefulness of these collaborations on the plan quality (in terms of optimality of makespans), some more experiments have been performed. Table 11 shows the results of these experiments for the scenarios of Table 1 with homogeneous robots and without hybrid reasoning. As expected, it can be observed that collaborations improve the plan quality. For instance, for Scenario 5, an optimal global plan with a coordination

Table 11 Plan quality versus collaboration between teams

Scenario	Teams	Workspace (grid cells)	Worker robots	Total robots	Order (toys)	Optimal global plan (W/collaboration)	Optimal global plan (W/o collaboration)
	#	#	#	#	#	length	length
1	2	15	1,2	5	6	30	34
2	3	15	1,2,3	9	9	25	34
3	4	15	1,2,3,4	15	12	21	34
4	2	24	2,4	8	8	20	29
5	3	24	2,4,6	18	12	18	29
6	4	24	2,4,6,8	30	16	18	29
1_b	3	21	1,1,2	7	9	30	31
2_b	4	21	1,1,2,2	10	16	30	39
3_b	5	21	1,1,2,2,2	13	25	46	46
4_b	3	21	2,2,4	12	12	23	25
5_b	4	21	2,2,4,4	18	20	20	30
6_b	5	21	2,2,4,4,4	24	30	25	35

has 18 steps; whereas an optimal global plan without any collaborations has 29 steps.

7.5 Discussion

We can summarize the results of these experiments, where we systematically investigate the role of each parameter on the computational efficiency and plan quality, as follows.

- About robot transfers: As more number of robots are allowed to be exchanged between teams, the teams become more collaborative and this results in shorter optimal global plans.
- About the workspace size: As the workspace increases, the optimal global plan length increases (since the teams need to navigate a bit more), the number of queries, in particular, the number of queries that are answered negatively decreases (since the teams can now navigate more comfortably, complete their tasks, and thus are more willing to help each other), and the computation time for querying generally increases (since the problem size increases).
- About the sizes of teams in a group of multiple teams: In general, larger teams are able to finish a task more quickly than smaller teams, since there are more robots that work on the task. This results in less number of queries and the computation time for querying becomes smaller.
- About extending a group of multiple teams by a new team: When a team of smaller size is included in the group of multiple teams, both the querying time and the optimal global plan length increase since a longer plan is needed for the smaller team to complete its task. When a larger team is included in the group, the querying time increases slightly even when the optimal plan length remains the same: more computation time is spent for queries about a larger team since the size of search space increases. There can be a decrease in the optimal global plan length when the additional team is not of a smaller size, and it can be used to help the other teams to reach a shorter plan. This also results in the decrease of total number of queries and hence the decrease in the computation times.
- About heterogeneity of teams: The optimal global plan length increases in certain cases when heterogeneous robots are considered, since the existence of robots with restricted capabilities may cause delays for the accomplishment of some tasks. On the other hand, the total number of queries increase when heterogeneity is considered. In many cases, this increase also leads to an increase in computation time. Meanwhile, the increase in the number of robots with restricted capabilities decreases total query time: If it is known that a robot cannot do a certain task, there is no need to look further and this decreases the computation time.
- About the order size (i.e., the amount of task to be completed): For teams with smaller sizes, both the total time of query answering and the optimal global plan length increase considerably as the size of the order increases. This is due to the increase in work load for each team, and that it takes more time to find a plan utilizing a small number of robots. When there are larger teams in a scenario,

the increase in the computation time for query answering becomes less severe. As expected, the increase in the order increases the optimal global plan length. Including larger teams results in shorter plan lengths, and thus less computation times. Essentially, larger teams are able to react more efficiently to new orders.
- About hybrid reasoning: The computation time increases since finding a feasible plan is time consuming. Also, the optimal plan length increases for some scenarios with hybrid reasoning, since the optimal global plan found without hybrid reasoning may not feasible.
- About finding a coordination, once the queries are answered: The computation time for coordination 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. 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.
- About parallelization and scalability: Parallelization of query answering helps with the scalability of our semi-distributed approach to find an optimal global plan. Even though both query answering and coordination problem are NP-complete, an optimal global hybrid plan can be computed within minutes for cognitive factories with many reasonably sized teams that help each other.

8 Conclusion

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. Furthermore, in this semi-distributed approach, the teams do not need to know about each others' tasks, workspaces and goals, and the mediator is neutral and does not need to know about the teams' tasks, workspaces and goals. This enables privacy-preserving planning of multiple teams of robots in critical tasks where the amount of communication is restricted.

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

Alami, R., Ingrand, F., & Qutub, S. (1998). A scheme for coordinating multi-robots planning activities and plans execution. In *Proceedings of ECAI*.

Bonisoli, A., Gerevini, A. E., Saetti, A., & Serina, I. (2014). A privacy-preserving model for the multi-agent propositional planning problem. In *Proceedings of ECAI* (pp. 973–974).

Brafman, R. I. (2015). A privacy preserving algorithm for multi-agent planning and search. In *Proceedings of IJCAI* (pp. 1530–1536).

Brafman, R. I., & Domshlak, C. (2008). From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of ICAPS* (pp. 28–35).

Brewka, G., Eiter, T., & Truszczynski, M. (2011). Answer set programming at a glance. *Communications of the ACM*, *54*(12), 92–103.

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

Canny, J. F. (1988). *The complexity of robot motion planning*. Cambridge, MA: MIT Press.

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., et al. (2006). Issues in multiagent resource allocation. *Informatica*, *30*(1), 3–31.

Dantam, N. T., Kingston, Z. K., Chaudhuri, S., & Kavraki, L. E. (2016). Incremental task and motion planning: A constraint-based approach. In *Proceedings of RSS*.

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

de Weerdt, M., & Clement, B. (2009). Introduction to planning in multiagent systems. *Multiagent and Grid Systems*, *5*, 345–355.

Decker, K., & Lesser, V. (1994). Designing a family of coordination algorithms. In *Proceedings of DAI* (pp. 65–84).

Desaraju, V. R., & How, J. P. (2012). Decentralized path planning for multi-agent teams with complex constraints. *Autonomous Robots*, *32*(4), 385–403.

Diankov, R. (2010). Automated construction of robotic manipulation programs. Ph.D. thesis, Carnegie Mellon University, Robotics Institute.

Dovier, A., Formisano, A., & Pontelli, E. (2009). An empirical study of constraint logic programming and answer set programming solutions of combinatorial problems. *Journal of Experimental and Theoretical Artificial Intelligence*, *21*(2), 79–121.

Duff, D. J., Erdem, E., & Patoglu, V. (2013). Integration of 3D object recognition and planning for robotic manipulation: A preliminary report. In *Proceedings ICLP 2013 workshop on knowledge representation and reasoning in robotics*.

Ehtamo, H., Hamalainen, R. P., Heiskanen, P., Teich, J., Verkama, M., & Zionts, S. (1999). Generating pareto solutions in a two-party setting: Constraint proposal methods. *Management Science*, *45*(12), 1697–1709.

Eiter, T., Ianni, G., Schindlauer, R., & Tompits, H. (2005). A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proceedings of IJCAI* (pp. 90–96).

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

Erdem, E., Gelfond, M., & Leone, N. (2016a). Applications of answer set programming. *AI Magazine*, *37*(3), 53–68.

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

Erdem, E., Haspalamutgil, K., Patoglu, V., & Uras, T. (2012b). Causality-based planning and diagnostic reasoning for cognitive factories. In *Proceedings of IEEE international conference on emerging technologies and factory automation (ETFA)*.

Erdem, E., Patoglu, V., Saribatur, Z. G., Schüller, P., & Uras, T. (2013). Finding optimal plans for multiple teams of robots through a mediator: A logic-based approach. *Theory and Practice of Logic Programming*, *13*(4–5), 831–846.

Erdem, E., Patoglu, V., & Schüller, P. (2016b). A systematic analysis of levels of integration between high-level task planning and low-level feasibility checks. *AI Communications*, *29*(2), 319–349.

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

Finger, J. (1986). Exploiting constraints in design synthesis. Ph.D. thesis, Stanford University.

Foulser, D., Li, M., & Yang, Q. (1992). Theory and algorithms for plan merging. *Artificial Intelligence*, *57*, 143–182.

Gaschler, A., Petrick, R. P. A., Giuliani, M., Rickert, M., & Knoll, A. (2013). KVP: A knowledge of volumes approach to robot task planning. In *Proceedings of IROS* (pp. 202–208).

Gaston, M. E., & desJardins, M. (2008). The effect of network structure on dynamic team formation in multi-agent systems. *Computational Intelligence*, *24*(2), 122–157.

Gebser, M., Kaufmann, B., Neumann, A., & Schaub, T. (2007). clasp: A conflict-driven answer set solver. In *Proceedings of LPNMR*.

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

Georgeff, M. P. (1988). Communication and interaction in multi-agent planning. In *Proceedings of DAI* (pp. 200–204).

Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., & Turner, H. (2004). Nonmonotonic causal theories. *Artificial Intelligence*, *153*, 49–104.

Gombolay, M. C., Wilcox, R. J., & Shah, J. A. (2013). Fast scheduling of multi-robot teams with temporospatial constraints. In *Proceedings of RSS*.

Gravot, F., Cambon, S., & Alami, R. (2005). Robotics research the eleventh international symposium. In *Springer tracts in advanced robotics, vol. 15, chap. aSyMov: A Planner That Deals with Intricate Symbolic and Geometric Problems* (pp. 100–110). Springer.

Hauser, K., & Latombe, J. C. (2009). Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries. In *Workshop on bridging the gap between task and motion planning at ICAPS*.

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

Havur, G., Ozbilgin, G., Erdem, E., & Patoglu, V. (2014). Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach. In *Proceedings of ICRA* (pp. 445–452).

Hertle, A., Dornhege, C., Keller, T., & Nebel, B. (2012). Planning with semantic attachments: An object-oriented view. In *Proceedings of ECAI* (pp. 402–407).

Hooker, J. (2005). A hybrid method for the planning and scheduling. *Constraints*, *10*(4), 385–401.

Kaelbling, L. P., & Lozano-Pérez, T. (2013). Integrated task and motion planning in belief space. *International Journal of Robotics Research*, *32*(9–10), 1194–1227.

Kavraki, L., Svestka, P., Latombe, J., & Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*(4), 566–580.

Kuffner Jr., J. J., & Lavalle, S. M. (2000b). RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of ICRA* (pp. 995–1001).

Kuffner Jr., J., & LaValle, S. (2000a). RRT-connect: An efficient approach to single-query path planning. In *Proceedings of ICRA* (pp. 995–1001).

Lagriffoul, F., Dimitrov, D., Bidot, J., Saffiotti, A., & Karlsson, L. (2014). Efficiently combining task and motion planning using geometric constraints. *International Journal of Robotics Research*, *33*(14), 1726–1747.

Latombe, J. C. (1991). *Robot motion planning*. Dordrecht: Kluwer Academic.

LaValle, S., & Hutchinson, S. (1998). Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, *14*(6), 912–925.

Lin, S. H. (2011). Coordinating time-constrained multi-agent resource sharing with fault detection. In *Proceedings of IEEM* (pp. 1000–1004).

Luo, L., Chakraborty, N., & Sycara, K. (2013). Distributed algorithm design for multi-robot task assignment with deadlines for tasks. In *Proceedings of ICRA*.

Maliah, S., Shani, G., & Stern, R. (2014). Privacy preserving landmark detection. In *Proceedings of ECAI* (pp. 597–602).

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

McCarthy, J., & Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer & D. Michie (Eds.), *Machine intelligence* (Vol. 4, pp. 463–502). Edinburgh: Edinburgh University Press.

Nair, R., Tambe, M., & Marsella, S. (2002). Team formation for reformation in multiagent domains like robocuprescue. In *Proceedings of RoboCup* (pp. 150–161).

Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., & Barry, M. (2001). An A-Prolog decision support system for the space shuttle. In *Proceedings of PADL* (pp. 169–183).

Plaku, E., & Hager, G. D. (2010). Sampling-based motion and symbolic action planning with geometric and differential constraints. In *Proceedings of ICRA* (pp. 5002–5008).

Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., et al. (2012). Team-building with answer set programming in the Gioia-Tauro seaport. *Theory and Practice of Logic Programming*, *12*(3), 361–381.

Saribatur, Z. G., Erdem, E., & Patoglu, V. (2014). Cognitive factories with multiple teams of heterogeneous robots: Hybrid reasoning for optimal feasible global plans. In *Proceedings of IROS* (pp. 2923–2930).

Shoham, Y., & Tennenholtz, M. (1995). On social laws for artificial agent societies:Off-line design. *Artificial Intelligence*, *73*, 231–252.

Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., & Abbeel, P. (2014). Combined task and motion planning through an extensible planner-independent interface layer. In *Proceedings of ICRA* (pp. 639–646).

Stuart, C. (1985). An implementation of a multi-agent plan synchronizer. In *Proceedings of IJCAI* (pp. 1031–1033).

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

Sycara, K. P., Roth, S. P., Sadeh, N. M., & Fox, M. S. (1991). Resource allocation in distributed factory scheduling. *IEEE Expert*, *6*(1), 29–40.

Tan, W., & Khoshnevis, B. (2000). Integration of process planning and scheduling—a review. *Journal of Intelligent Manufacturing*, *11*(1), 51–63.

ter Mors, A., Valk, J., & Witteveen, C. (2004). Coordinating autonomous planners. In *Proceedings of IC-AI* (pp. 795–801).

Tiihonen, J., Soininen, T., & Sulonen, R. (2003). A practical tool for mass-customising configurable products. In *Proceedings of the international conference on engineering design* (pp. 1290–1299).

Torreño, A., Onaindia, E., & Sapena, O. (2012). An approach to multi-agent planning with incomplete information. In *Proceedings of ECAI* (pp. 762–767).

Torreño, A., Sapena, O., & Onaindia, E. (2015). Global heuristics for distributed cooperative multi-agent planning. In *Proceedings of ICAPS* (pp. 225–233).

Trejo, R., Galloway, J., Sachar, C., Kreinovich, V., Baral, C., & 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.

Turpin, M., Michael, N., & Kumar, V. (2013). Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In *Proceedings of ICRA* (pp. 842–848).

Velagapudi, P., Sycara, K. P., & Scerri, P. (2010). Decentralized prioritized planning in large multirobot teams. In *Proceedings of IROS* (pp. 4603–4609).

Weyhrauch, R. W. (1978). *Prolegomena to a theory of formal reasoning*. Tech. rep.: Stanford University.

Wolfe, J., Marthi, B., & Russell, S. (2010). Combined task and motion planning for mobile manipulation. In *Proceedings of ICAPS* (pp. 254–258).

Yang, Q., Nau, D. S., & 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., Rühr, T., Friedrich, M., & Herle, S. (2009). The cognitive factory. In *Changeable and reconf. manufacturing systems* (pp. 355–371).

Zeynep G. Saribatur is a graduate student at TU Wien. She received her B.S. in computer engineering at Bogazici University (2012) and her M.S. in computer science and engineering at Sabanci University (2014). Her research interests include artificial intelligence, knowledge representation and reasoning.

Volkan Patoglu is an associate professor in mechatronics engineering at Sabanci University. He received his Ph.D. in mechanical Engineering at the University of Michigan, Ann Arbor (2005), and carried out postdoctoral research at the Rice University from 2005 to 2006. His research is in the area of physical human-machine interaction, in particular, design and control of force feedback robotic systems with applications to rehabilitation and skill training. His research extends to cognitive robotics.

Esra Erdem is an associate professor in computer science and engineering at Sabanci University. She received her Ph.D. in computer sciences at the University of Texas at Austin (2002), and carried out postdoctoral research at the University of Toronto and Vienna University of Technology from 2002 to 2006. Her research is in the area of artificial intelligence, in particular, the mathematical foundations of knowledge representation and reasoning, and their applications to cognitive robotics and computational biology.