A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes

Lars Johannsmeier and Sami Haddadin

***Abstract*—In this letter, we propose a framework for task allocation in human-robot collaborative assembly planning. Our framework distinguishes between two main layers of abstraction and allocation. In the higher layer, we use an abstract world model, incorporating a multiagent human-robot team approach in order to describe the collaborative assembly planning problem. From this, nominal co-ordinated skill sequences for every agent are produced. In order to be able to treat humans and robots as agents of the same form, we move relevant differences/peculiarities into distinct cost functions. The layer beneath handles the concrete skill execution. On atomic level, skills are composed of complex hierarchical and concurrent hybrid state machines, which in turn co-ordinate the real-time behavior of the robot. Their careful design allows to cope with unpredictable events also on decisional level without having to explicitly plan for them, instead one may rely also on manually designed skills. Such events are likely to happen in dynamic and potentially partially known environments, which is especially true in case of human presence.**

***Index Terms*—Physical Human-Robot Interaction, Assembly, Co-Worker, Optimal Planning.**

I. Introduction

THE STRICT separation of human and robot workspaces was considered a necessity until a few years ago, as humans were usually considered as disturbances to the execution of a robot task and robots in turn posed significant threats to the human. However, with the recent emergence of safer robots [1] that enable humans and robots to share the same workspace, the field of physical human-robot-interaction (pHRI) gained also significant practical relevance. In [2], [3], [4], [5] one can find overviews on the current state of the art in human-robot collaboration. On the practical side, pHRI is nowadays of particular interest in purely manual industrial assembly tasks. Despite robots have proven to be well suited for many tasks and the potential of safer robots is now well understood, numerous rather complex processes cannot be accomplished yet by robots alone due to their limited cognitive capabilities. In particular, flexible tasks, as well as unstructured or underspecified environments pose still significant challenges to full automation. Therefore, the combination of human cognitive skills with the skills of collaborative robots allows not only for working alongside humans and in partially unknown environments, but also collaborate with other agents, let them be humans or other robots, for accomplishing a common goal. Consequently, the robot needs to be able to plan complex tasks, which involve interactions, collaborations, common goals and certain group dynamics.

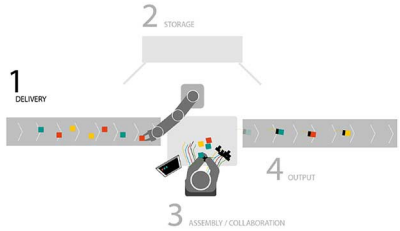

Fig. 1. Collaborative assembly applications of human-robot teams as a milestone towards pHRI in everyday scenarios.

Realizing such human-robot collaboration in assembly tasks is a rather difficult problem. Since humans and robots share common workspaces, also safety is obviously of great concern and injuries have to be avoided [6]. However, this topic is clearly out of the scope of this letter. Moreover, humans introduce potential indeterminism into the overall process.

The scope of the letter is as follows. We consider industrial assembly processes that are typically well understood and planned in advance. Therefore, we consider teams that build products, which construction plans are known beforehand. Also, we assume that during task execution all necessary tools and parts are fed to the team by suitable mechanisms such as conveyor belts, see Fig. 1. In turn, the requirements for autonomy are restricted to specific problems such as being able to execute a set of assembly and manipulation skills, or to communicate with humans and other robots within the context of assembly processes.

Up to now, existing approaches aim for performing basic and isolated assembly steps in human-robot coexistence. There, each agent performs its task separately and non-physical communications are exchanged also based on estimating the human intentions [7], [8], [9]. However, these works do not consider yet the planning of entire collaborative assemblies, where physical interactions are an essential skill and play a great role in the successful execution of a given task. Furthermore, to

Manuscript received August 30, 2015; accepted January 28, 2016. Date of publication February 29, 2016; date of current version April 11, 2016. This paper was recommended for publication by Associate Editor D. Prattichizzo and Editor Y. Yokokohji upon evaluation of the reviewers' comments. This work was supported by the European Union's Horizon 2020 research and innovation programme under Grant 688857.

The authors are with the Faculty of Electrical Engineering and Computer Science, Institute of Automatic Control, Leibniz Universität Hannover, D-30167 Hannover, Germany (e-mail: johannsmeier@irt.uni-hannover.de; haddadin@irt.uni-hannover.de).

Digital Object Identifier 10.1109/LRA.2016.2535907

the best of the authors' knowledge, the automatic generation of entire collaborative assembly plans and the according allocation to the team members, potentially even under optimality considerations, has not yet been considered. Related work with a stronger emphasis on the human and the communication with the robot can be found in [10], [11]. Further, in [12], e.g., the system *SHARY* for human aware task planning was introduced. It produces social plans of a task by implementing communication schemes to negotiate the task solution with the human partner. In [8] Dynamic Neural Fields (DNFs) were used to build a decision making system for interaction in cooperative human-robot tasks.

In summary, the contribution of this letter is as follows. We propose a framework for human-robot collaboration that comprises three different architectural levels: the *team-level assembly task planner*, the *agent-level skill planning*, and the *skill execution level* that is the final decisional component above the robot real-time control stack.

The planner at team-level performs the task allocation for the agents based on an abstract world model and with the help of suitable cost metrics. In this context, note the first theoretical analysis in multi-agent task allocation [13]. To model our types of assembly processes suitably we employ AND/OR graphs [14] as they implicitly model parallelism. The team-level planner produces task sequences for every agent via A* graph-search, from which it derives task descriptions that are then passed down to the agent's skill execution level. The agents in turn implement modular and parameterizable skills via hierarchical and concurrent state machines [15] in order to map abstract task descriptions to the subsequent real-time-level. Here, the concrete motion and interaction controls are performed in combination with protective reflexes.

The proposed separation in terms of entities (agents, team-level planner) and abstraction leads to an efficient planning framework that can produce optimal nominal task plans to build a desired assembly, while handling failures due to dynamic and uncertain environments in a safe and reliable way on the lower levels of control.

The remainder of this letter is organized as follows. Section II gives a brief overview on modeling assembly planning and assembly processes based on AND/OR graphs. In Section III the structure of our collaborative assembly and task allocation framework, both at team- and agent-level, is provided. In Section IV two basic experiments underline the feasibility of our proposed approach. Finally, our letter concludes in Sec. V.

II. Assembly Planning

To be able to create (large) assembly plans and design control processes that implement those plans, it is necessary to select a proper representation. In this section we briefly introduce such a representation and make reasonable assumptions about the structure of the assembly, simplifying the subsequent theoretical analysis. Note that in this work we do not treat the generation process of assembly plans. Instead, we assume that we already have such plans at our disposal. A popular algorithm to generate assembly plans is e.g. described in [16].

In the following, a mechanical assembly is denoted by M. An intuitive representation for a single assembly is the graph of connections G. This is obtained by simplifying the graph of contact, which was already used in several works [16], [17], [18]. The resulting graph of connection contains a node for every part $p \in P$, where $P \subset M$, and an edge if and only if there is at least one contact between the two respective parts.

The parts p are assumed to be solid rigid objects, i.e. their shape remains the same throughout the complete assembly process. A subassembly $P_s \subset P$ is defined as a subset of P that has similar properties as the complete assembly. If a subassembly consists of more than one part, all the parts are connected, meaning the graph of connections induced by that subassembly is connected. For the sake of simplicity we assume that every part p is unique, i.e. they are not interchangeable when they have equal properties (e.g. two screws of the same type). Furthermore, we assume that the geometry and characteristics of the connection of every pair of parts are unique. Thus, every subassembly implicitly defines the complete geometry of the involved parts as well as their connections. A subassembly with only one part is considered *atomic*.

In order to build an assembly an *assembly plan* is needed, which describes the possibilities of how to assemble a workpiece. In particular, it assumes that all parts are in their designated places and the necessary connections were made already, so that one begins with the correct subassemblies. The process of executing the assembly plan is called *assembly process*. We call the process of assigning the available workers $w \in W$ (for which we also use the more general term *agent*) to the assembly actions *task allocation*. A sequence of assembly actions that lead from the initial configuration to the finished product is called an *assembly sequence* α which, in general, is a partially ordered set of assembly actions a. An *assembly action* a denotes one step in the assembly process.

Moreover, we introduce three predicates to further concretize the scenario we are investigating[1]. First, we assume that a subassembly P_s is stable, $st(P_s)$, i.e. its parts remain in contact without applying any external forces and the relative geometry of all parts does not change. Then, we classify the actions a that are used in the assembly process as geometrically feasible and mechanically feasible:

- $gf(a)$ denotes that there exists a collision free path to join the two involved subassemblies.
- $mf(a)$ holds if it is possible to permanently establish all necessary connections.
  a) *Sequentiality*: A plan is sequential if there exists a sequence of assembly actions, involving only one subassembly that can represent the plan. This means that it is never necessary, though possible, to move two subassemblies at a time. A plan that holds this property could thus be executed by a robot with a single manipulator. Whether an assembly can be described by a sequential plan depends on its geometry. Also, it requires that every subassembly is stable.
  b) *Monotonicity*: A plan is called monotone if it contains no action that would break an already established connection or modifies the relative geometry

[1]Please note that they hardly pose any constraints when considering real-world assemblies.

in an already existing subassembly. Monotonicity is often built into real world assemblies.

c) *Coherency*: A plan is coherent for a graph G if every subassembly occurring in the plan corresponds to a connected subgraph of G. More formally, there exists a connected graph G such that

- there exists an isomorphism from the set of parts to the set of nodes in G and
- for every subassembly occurring in the plan, the subgraph of G that is induced by that subassembly is connected.

A. Assembly Plan Representation

In a collaborative scenario where actions are distributed among several agents $w \in W$, it may be desirable and more efficient that some of the actions are executed in parallel. In particular, when humans and robots work together, scenarios become possible where joining subassemblies may require rather complex and delicate procedures that are extremely hard to be automated, while assembly of subassemblies may be very easy to automate. Consequently, one possibility is that the human handles the complex task of joining the subassemblies, while the robotic co-workers prepare the mentioned subassemblies and "feed" them to the human when needed.

As mentioned above we chose AND/OR graphs as the representation of assembly plans because of their ability to explicitly facilitate parallel execution of assembly actions, as well as the time independence of parallelly executable actions. Using AND/OR graphs for the representation of assembly sequences was first proposed in [14]. More recent applications can be found in [19]. The AND/OR graph of a particular assembly can be constructed by disassembling the complete assembly until only atomic parts p are left. Having this idea in mind, the graph of feasible assembly actions is built.

Definition 1: (AND/OR graph of feasible assembly sequences): Let $M = (P, st, gf, mf)$ be a mechanical assembly. The *AND/OR graph* Y_M *of feasible assembly sequences* of assembly M is defined as the hypergraph (V, E), where

$$V = \{P_s | P_s \subseteq P \wedge st(P_s)\},$$
$$E = \{\{P_{s,k}, P_{s,i}, P_{s,j}\} | P_{s,k}, P_{s,i}, P_{s,j} \in V\},$$

and

$$P_{s,k} = (P_{s,i} \cup P_{s,j}) \wedge gf(\{P_{s,i}, P_{s,j}\}) \wedge mf(\{P_{s,i}, P_{s,j}\}).$$

Note, that though each edge in the hypergraph is an unordered set, one of the three subassemblies, namely $P_{s,k}$ is distinguished because it is the union of the other two sets $P_{s,i}$ and $P_{s,j}$.

Figure 2 shows an excerpt from an AND/OR graph of a four part assembly consisting of the parts $P = \{A, B, C, D\}$.

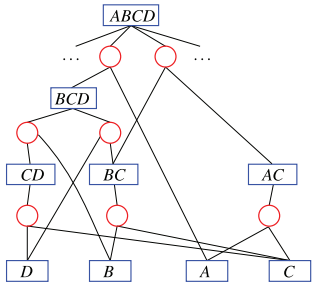

Fig. 2. Partial AND/OR graph of an exemplary assembly. The blue colored rectangles depict OR nodes, the red colored circles AND nodes.

III. Task Allocation and Planning

In this section we discuss our task allocation framework that handles multi-agent planning, single-agent planning and real-time execution in assembly scenarios, see Fig. 3 for an overview. The used notation $w \rightarrow a$ denotes a specific allocation of agent w to action a. *Atomic actions* are the most basic implementations at agent-level and communicate directly with the real-time-level. The framework comprises three main levels, which are strictly separated. However, they communicate bilaterally with each other and share common information. Although we consider the approach as rather general in principle, we focus on its application to industrial assembly processes for sake of clarity.

The **first level**, called *team-level*, plans the assembly process from the view of a foreman, i.e. it has information about the possible construction plans, assembly parts, the available agents and other resources. It knows only actions that can manipulate this world model in an abstract way, i.e. it is domain-specific. The planning process on team-level solves the problem of task allocation in the assembly process for a team of human and robotic workers. It should be noted that at this level of abstraction we make no explicit distinction between human and robotic workers, since the team-level planner relays only abstract task descriptions without the need to take into account specific implementations of the necessary skills. This is due to the fact that the cost function (see Sec. III-A3) makes it possible to distinguish between agents by encoding their respective capabilities. At this point it is important to state how the planner interacts with the agents. During the search process the planner sends the request to perform a specific action to an agent. The agent answers with the cost and possibly a further request for an interaction, which in turn is integrated into the planning process. If an agent is not able to perform a specific action it returns infinite costs. The planner then uses the received costs from all available agents to determine the optimal task allocation. Please note further, that the planning process at team-level is offline, see Fig. 3, hence, fast replanning and refinement methods are out of the planner's scope, despite it might be possible for a manageable assembly complexity.

The **second level**, called *agent-level*, maps the team-level planning process to a degree of abstraction just above the actual body (actuators and sensors) of the robot. In our case we use the already mentioned hierarchical and concurrent state machines. In general, the agent-level may of course contain also other sophisticated planning systems, which in turn use robot specific actions to accomplish a given task. However, for executing industrially relevant skills that need a high level of expert knowledge, practical experience shows that their automatic planning instead of semi-handcrafting in collaboration with process experts is the significantly less capable approach

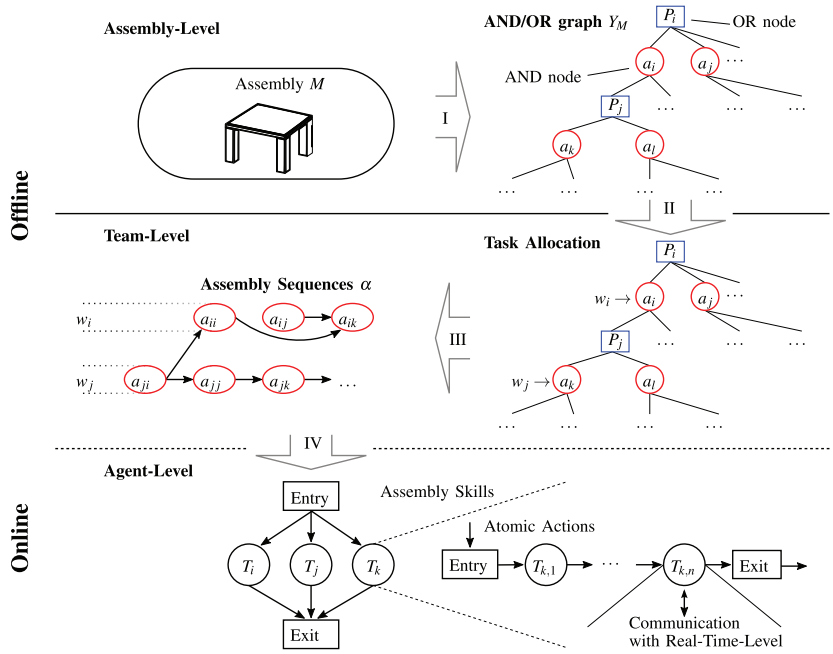

Fig. 3. Collaborative assembly planning framework. The top layer depicts the input to the team-level planner, which is an AND/OR graph Y_M resulting from an assembly M (Step I) . The planner then solves the problem of optimal task allocation (Step II) for multiple agents considering a given cost function and constructs a set of assembly sequences (Step III). The actions from the sequences are then passed to the respective agents (Step IV), which possess corresponding skills T. These skills in turn consist of more basic structures, i.e. atomic actions, which eventually map to the real-time-level.

as of today. Note that since the agent-level is implemented on the robot itself, it is possible to use different robot types. For this, the systems have to provide the necessary formal semantics to the team-level, which obviously requires the systems to be able to execute the same actions in principle. Another important task of the agent-level is the consistent handling of events such as collisions or human induced interruptions of the assembly process. Although the reaction to the actual event itself is assumed to be handled by the real-time layer, the agent-level either has to deal with the consequences in a way that is consistent with the overall plan on team-level, or report a failure in plan execution and request replanning from the team-level planner.

The **third level**, the real-time level, is directly responsible for executing trajectory planning, controllers, etc. At this level, also reflexes of the robot to unforeseen events are implemented as simple reactions to unexpected disturbances. Further details about this level can e.g. be found in [20].

In the following, we discuss the respective levels in more detail.

A. Team-Level Planning and Task Allocation

Let us now elaborate the team-level planner in the context of assembly planning. First, the planner takes an assembly plan in the form of an AND/OR graph as input. Furthermore, a reference to a database is needed, which contains the necessary information about part locations, agents and other resources. To optimally solve the allocation problem with respect to a particular cost metric, we apply the well-known A* graph search algorithm [21]. In the following, we provide the formal problem statement, relevant cost metrics and heuristics, as well as further considerations concerning multi-agent allocation.

1) Problem Statement: The overall goal is to find an allocation of agents to assembly actions that is optimal with respect to a specified cost function. In the following, $S = \{s_1, \ldots, s_n\}$ denotes the set of states, each of which corresponds to a set of OR nodes $v_o(s) \subseteq V_o$, where V_o denotes the set of all OR nodes in the AND/OR graph Y_M. Similarly, the assembly actions $a(s)$ that are applicable in a state s correspond to a set of AND nodes $v_a(s) \subseteq V_a$, where V_a denotes the set of all AND nodes in Y_M. Since Y_M and the number of available agents, and therefore, the number of possible allocations, are finite, the search space is finite as well.

The general idea is now to propagate through Y_M from the root node via disassembly actions until some state contains only *atomic* subassemblies. The following formal definition of the problem complies to the simplifications for the assembly process we introduced in Sec. II.

- The initial state s_0 corresponds to the root node $v_{o,0}$ of Y_M, which corresponds to the complete assembly M.
- In a state s the number of applicable assembly actions is at most as large as the number of OR nodes $n_{s,o}$ in that state due to strict sequentiality. Also, a single OR node can provide at most one AND node to a valid set of actions, since a given assembly can not be disassembled into two different configurations at the same time. Therefore, in s there are $n_{s,a} = \prod_i n_{a,i}$ different valid sets of actions, where $n_{a,i}$ denotes the number of children of the i-th OR node. Also note that an agent w does not have to be assigned to an action. In addition, every action from $a(s)$ can be assigned to every agent $w \in W$. In a state s, there is a maximum number of assignments

$$N_A = \sum_{i=1}^{l} n_{s,a} \binom{n_w}{i} n_{s,o}!,$$

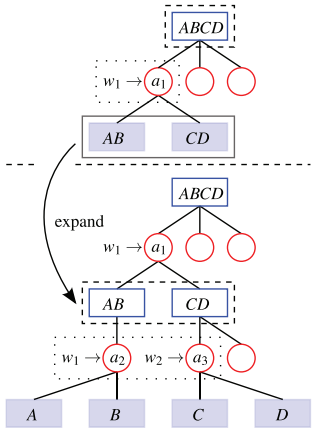

Fig. 4. Example propagation via search through search space. In the top graph agent w_1 has been assigned to the indicated AND node in state s and thus, the corresponding assembly action. This AND node has two children that are expanded and form the new state s'. In the bottom graph two parallel actions are assigned, which again yield two children, respectively. The dashed boxes depict the currently active state, the dotted ones the chosen allocation.

where $l = \min(n_w, n_{s,o})$ is either the number of available workers n_w or the number of OR nodes with children n_o, i.e. whichever is lower.

- The goal state s_g contains only OR nodes that correspond to *atomic* subassemblies.
- The cost of an action can be chosen from different possibilities and will be further elaborated in Sec. III-A3.

To illustrate the propagation of the search algorithm through the search space (i.e. the AND/OR graph) Fig. 4 depicts an example.

2) Multi-Agent Considerations: Up to now, we made no particular assumptions regarding the number of workers in the problem statement. Although the AND/OR graph implicitly models parallelism, it is at no point required to have more than one worker available if the simplifications from Sec. II are met. Yet, to potentially increase efficiency/speed of the assembly process, we use a team of agents. For this, we have to consider some factors, which are introduced into the planning process on team-level.

a) Actions and Interactions: Within a multi-agent scenario in the domain of collaborative assembly planning, actions are not just used to build the assembly but also to enable interactions between agents. Therefore, we distinguish between actions that directly modify the assembly and actions that characterize an interaction between two agents. Also note, that at team level actions are considered to be *atomic*, i.e. they directly change the model of the world. At agent level the same actions are called skills which in turn map to the real-time-level. We formally define the actions at team level as follows.

Definition 2: (Assembly action): Let $a(type, \mathcal{W})$ be a general action in the context of assembly processes. Let *type* denote a descriptor that specifies the action type that is requested and $\mathcal{W}$ can either be a single assignment $\mathcal{W} = w_i$, or a pair assignment $\mathcal{W} = (w_i, w_j)$ depending on the *type*. Note that we consider interactions, i.e., actions with a pair assignment with at most two agents.

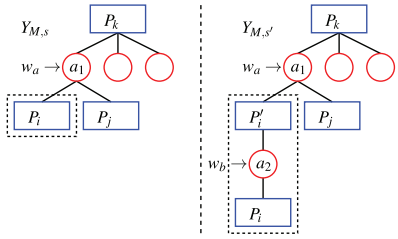

Fig. 5. To model an interaction between two agents in the AND/OR graph we insert a new AND/OR node pair. In this example, the subassembly P_k is created by agent w_a via joining P_i and P_j, which corresponds to the action $a_1 := a(assemble, w_a)$. Yet, some kind of interaction is necessary to complete the assembly step. E.g., the performing agent cannot reach P_i so another agent w_b needs to hand this part over. Therefore, P_i' and the AND node that corresponds to the interaction $a_2 := a(hand\_over, (w_a, w_b))$ are inserted on the right side.

If e.g. $type = hand\_over$ and $\mathcal{W} = (w_1, w_2)$, a needed part is not reachable by w_1 and must be provided by w_2 via a hand-over action.

Note that there is no need to change the problem statement to include interactions, it is only necessary to integrate them into the expanded AND/OR graph $Y_{M,s}$ that is defined as follows.

Definition 3: (Expanded AND/OR graph): Let $Y_{M,s'}$ be called the local AND/OR graph of the state s', then $Y_{M,s'}$ is created as a copy of $Y_{M,s}$ of state s, which is the predecessor of s'. The initial Y_M is the AND/OR graph of s_0.

Since every interaction is specific to the assigned agent pair and arises from a particular context (e.g one agent cannot reach a part, yet another can, or one agent needs the assistance of another agent to join two subassemblies), it makes sense to change the AND/OR graph only locally in the search problem, i.e. for the respective state s. Over a sequence of states s the graph is expanding with the occurrence of interactions.

If an interaction is necessary, we insert a new OR node with the corresponding subassembly P_s' and a new AND node v_a, which represents the interaction, see Fig. 5. The new AND node (i.e. the action) can then be assigned to an agent as described above. The new (intermediate) subassembly P_s' has the same parts as P_s i.e. $p(P_s') = p(P_s)$. However, the state of $p(P_s')$ is different from that of $p(P_s)$ depending on the *type* of interaction.

b) Synchronization: Another important aspect in a multi-agent scenario is synchronization. In an ideal situation, the nominal plan would be executed in the real world without any modifications. In practice, this is highly unlikely, in particular due to dynamically changing environments and uncertainty. Thus, it is not possible to guarantee e.g. a successful hand-over action without communication. Since the action sequences arise from the AND/OR graph, every need for synchronization can be dealt with by interactions.

This means that agents do not need to synchronize their actions, as long as their action sequences α are not connected via an interaction. Since the team-level planner knows (via confirmation from the agents) which actions have been performed, it can let agents wait until the requirements for their next scheduled task are fulfilled. This way only tasks that are applicable are performed. We refer to Fig. 6 for the following example, which is also the graph for the experiments in Sec. IV: Consider

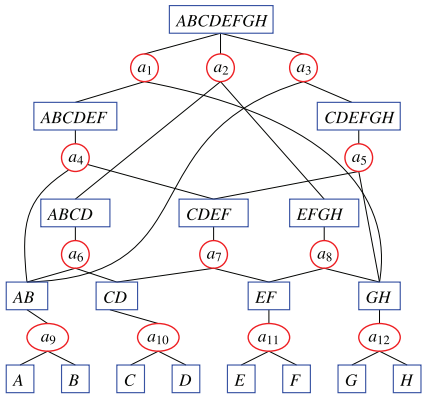

Fig. 6. AND/OR graph of the experiments assembly plan.

an agent w_1 that performs action a_{11} and is then scheduled to perform action a_7. Also consider agent w_2 that performs action a_{10}. w_1 can not perform a_7 as long w_2 does not confirm the completion of a_{10}. Hence, the structure of the AND/OR graph represents the agents actions.

3) Optimization Metric and Heuristic: Although the most obvious way to evaluate the cost of an allocation is to measure the overall execution time, there may be other metrics that are more important in specific situations. Furthermore, it is unlikely that time constraints assumed by the team-level planner are adhered to by the human co-workers, since the daily work routine of a factory worker is undetermined over short periods of time. Considering this and the fact that we do not explicitly distinguish between humans and robots at team-level, we need other cost metrics that are more applicable in human-robot collaboration and encode all necessary information about the specific workers. For example, the assembly could involve the handling of dangerous and/or heavy material and thus the human workload and risk should be minimized. Let us therefore introduce some example cost functions that are suitable candidates to our given problem:

- Execution Time: We may distinguish between overall execution time and local execution time, i.e. the time needed for the execution of a single action.
- Resource costs: Resources such as energy consumption, peak power,. . .could be taken into account.
- Risk factors: Danger to human, workload amplitude and frequency or ergonomic factors may be of relevance.
- Assumptions about the human worker: A worker profile could be generated that maps to a cost function, using properties such as attention level, general experience level, and reliability.

A cost function for a robot could e.g. look like $c_r(w,a) = k_1 f_t(w,a) + k_2 f_p(w,a)$, where w and a denote the worker-action pair of the assignment, f_t the amount of time w would need to perform a and f_p the amount of power consumption of w when performing a. k_1 and k_2 are weighting factors. A human cost function could be $c_h(w,a) = k_1 f_a(w,a) + k_2 f_w(w,a)$, where f_a is a measurement for the anticipated attention level of the human when performing action a and f_w is a workload measurement.

Note that joint costs, i.e. costs that arise from interactions, can be treated explicitly. This is useful in order to integrate interdependencies between two specific agents into the cost function.

Now, in order to make use of the advantages of the A* algorithm, a suitable heuristic needs to be defined. Our approach for the stated search problem is as follows. The minimum amount of assembly actions $n_{a,min}$ that need to be applied under the assumptions from Sec. II to get from the current state s to the goal state s_g are found to be

$$n_{a,min} = \log_2(\max_i(n_p(v_{o,i}))).$$

The function n_p yields the number of parts of the subassembly P_s that corresponds to the OR node $v_{o,i}$. An admissible and consistent heuristic can then be derived by multiplying $n_{a,min}$ with the minimum cost an agent can achieve for any action that has not yet been applied.

4) Problem Reduction: Due to the large number of possible agent assignments to actions if many agents are used, or a complex assembly with many possibilities is to be built, it is often more efficient to reduce the problem complexity first.

The method we propose to achieve this, is to reduce the search space by simplifying the AND/OR graph. For this, we introduce *reducible subassemblies*

Definition 4: (Reducible Subassembly): Let $v_{o,R}$ be the root node of the partial AND/OR graph $Y_R \subset Y_M$ that is induced by the *reducible subassembly* $P_{S,R}$ and $V_{a,p}$ the set of parent nodes of $v_{o,R}$. If all edges $E = \{\{(v_{o,R}, v_o\}|v_o \in V_{a,p}\}$ were removed, the result would be two distinct AND/OR graphs, i.e. $Y_R \cap Y'_M = \emptyset$, where $Y'_M = Y_M \setminus Y_R$.

Obviously, this scheme is a *divide-and-conquer* approach. The graph Y'_M is now smaller because it only contains the root node $v_{o,r}$ of the *reducible subassembly* $P_{S,R}$. Hence, the allocation problem is easier to solve. This is the case because several small search spaces are now present instead of a single larger one.

B. Agent-Level Task Allocation

The agent-level implements the autonomous behavior of a single robot to ensure the safe and successful execution of the actions that are assigned to the robot by the planner at team level. The assembly skills at agent level directly correspond to the actions received from the team-level planner. Arrow IV in Fig. 3 depicts this connection. For a robot to be able to execute such assembly skills, they have to be implemented at agent level for which we use hierarchical and concurrent state machines [15]. We call the skills defined by the highest layer of this state machine compound skills, since they are composed of more basic ones, which are also called *atomic action* (see bottom layer of Fig. 3). This approach originates from [20]. The most important basic skills we implemented are so far:

- *pick-up part:* This action assumes that the robot is already at the right location to perform the pick-up. It then adjusts its end-effector according to its available information about the specified part and grasps it.
- *assemble part:* This action assumes that the robot is at the right location to assemble the part. Furthermore, it implements the process of assembling the part.

TABLE I
COST METRICS C_1 (LEFT) AND C_2 (RIGHT)

C_1	r_1	r_2	h
a_1	∞	∞	20
a_2	∞	∞	5
a_3	∞	∞	15
a_4	10	10	20
a_5	5	5	5
a_6	20	10	3
a_7	10	5	15
a_8	∞	5	10
a_9	20	20	5
a_{10}	10	10	5
a_{11}	10	10	10
a_{12}	10	10	10

C_2	r_1	r_2	h
a_1	∞	∞	50
a_2	∞	∞	50
a_3	∞	∞	50
a_4	10	10	200
a_5	5	5	50
a_6	20	10	30
a_7	10	5	100
a_8	∞	5	100
a_9	20	20	50
a_{10}	10	10	50
a_{11}	10	10	100
a_{12}	10	10	100

- *hand-over:* To realize the hand-over of an object from one to another, several modalities are possible. In our case, we use so called haptic gestures [22], which correspond to an interpretation of contact force patterns and the according execution of behaviors. Other approaches to hand-over tasks can for example be found in [23].
- *collision handling:* This action corresponds to a collision reflex reaction based on the interpretation of sensed contact forces along the robot structure. In case of collisions the robot switches e.g. to manual guidance mode. The human may then safely store the robot at pre-defined locations until the human signals e.g. via haptic gestures that the robot may continue. Suitable algorithms for the detection of collisions can be found in [24]. We assume that the human brings the robot to the right state to reengage to the task.

Next, the experimental performance of our approach is validated.

IV. EXPERIMENTS

We conducted two kinds of experiments. First, we investigated how our planner works on team-level in a realistic setting, however, isolated from real-world effects. In the second experiment we tested the whole framework in a real human-robot collaborative assembly scenario.

To show the output of the team level planner we conducted a computational experiment in which a small assembly, consisting of eight parts is to be assembled by a team of two robots and one human with two different cost measures. The agents are denoted by $W = \{r_1, r_2, h\}$. The corresponding AND/OR graph is shown in Fig. 6. None of the atomic parts are in humans reach and the time needed for hand-over actions is estimated to be 8 seconds. Except hand-over actions no interactions are involved.

As cost function we use $c_m = \max_i c(\langle w, a\rangle_i).\langle w, a\rangle$ denotes the assignment of an agent w to an action a in the state s. The specific cost c for an assignment $\langle w, a\rangle$ is listed in Table I. The left table shows the amount of time the agents need for a specific action, i.e. the cost metric C_1. In the second cost metric C_2, human workload is considered as well. Metrics for measuring the human workload can for example be found in [25]. The resulting action sequences are depicted in Fig. 7.

As can be seen from Fig. 7a, the planner produces parallelized execution schemes where possible, which leads to a short overall execution time. A disadvantage of such a parallelized assembly process is the dependency of the agents on each other. If one agent is disturbed in its task, other agents may have to wait. In Fig. 7b the human workload has been considered as well, resulting in an execution scheme where the robots work in parallel and the human is only assigned to a task the robots are not capable of performing.

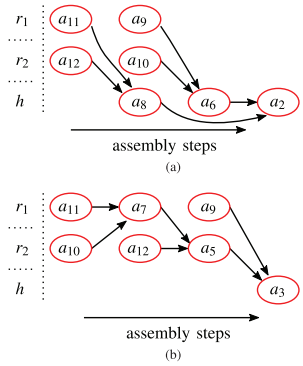

Fig. 7. Agents' assembly sequences for C_1 (a), and C_2 (b). The solid arrows depict a precedence relation, i.e. the source of the arrows provides a needed subassembly to the sink.

Note, that in order to formally incorporate the aspect of a (human) worker being distracted and therefore potentially disturbing the entire assembly process, one could use a cost function that encodes e.g. some form of worker profile (as mentioned in Sec. III-A3). The probability of a worker being distracted from his task could depend, among others, on his daily routine and experience.

The simulation experiments show that the planner can be adapted to the requirements of a specific scenario by providing a cost function that reflects the needs of the situation. E.g. consider that the assembly from Fig. 6 is being built in large quantities by human-robot teams. When the demand is normal, a cost metric similar to C_2 could be used. The robots would do most of the work while the human co-workers could be available to other tasks as well. If the demand rises, a cost metric such as C_1 could be used, resulting in a higher production output at higher human workload.

The simulation experiments were performed on a computer with a Windows 7 operating system, an Intel i7-3770 processor with 3.4 GHz and 4 GB RAM. The A* algorithm took about 40 ms, respectively 2 ms to calculate the action sequences and expanded 74, respectively 6 nodes. A Fibonacci heap was used as data structure for the open list and a hash table for the closed set. In order to indicate the scaleability of our approach we conducted an additional experiment with a strongly connected assembly [14] with 10 parts, two workers and random costs for the worker-actions pairs. The amount of planning time averages at 15 s. As another example we conducted the same experiment on a binary assembly with 32 parts. All subassemblies of a binary assembly can be divided into two subassemblies with an equal number of parts. Here, the average execution time is 3.5 seconds.

The second experiment (its setting is not related to the previous simulations) was conducted with a real robot in a collaborative assembly scenario, see Fig. 8. Note, that for

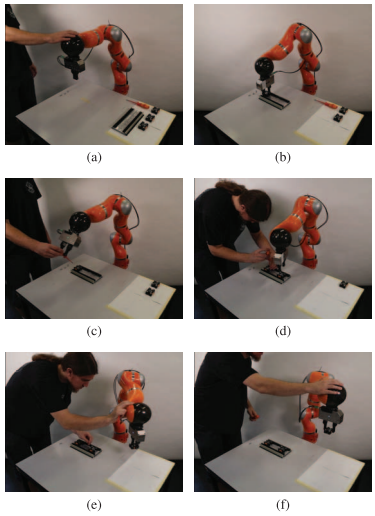

Fig. 8. The robot is started by the human (a), the robot uses its assembly skills (b), a tool is handed over to the human (c), the human co-worker and the robot work in parallel (d), the human stops the robot in order to finish an assembly step the robot would otherwise disturb (e), the human signals the robot to continue its nominal behavior (f).

sake of simplicity we did not time the actions of the robot and the human (apart from the implicit, discrete timing determined by the team-level) in this experiment. Nonetheless, it is straight forward to introduce an additional timing e.g. via simple human confirmation. Such an input would be incorporated on team-level. Other synchronization schemes that could be employed on agent- or real-time-level are for example time scaling, i.e. the robot would drive slower or even stop in the vicinity of the human [1]. For such capabilities visual perception and/or real-time robot-to-robot communication would have to be available.

V. Conclusion

In this letter we proposed a framework for collaborative assembly planning that is suited to solve real-world assembly problems by combining the capabilities of humans and robots in an optimal way. Our layered architecture enables the system not only to generate nominal plans, but also react to and cope with unforeseen and possibly faulty events, a crucial capability to be prepared for the real-world. Our layered planning scheme differentiates between the human-robot team as a whole, the single agent, i.e. every robot and human, and the real-time command structure of that agent. This makes the system able to deal with problems at the right level of abstraction in order to find the most efficient solution, respectively. Also experimentally the system performs very well and shows promising results. Future work will in particular concern the application of our system to other scenarios and also apply it in real-world factory settings.

References

[1] S. Haddadin, "Towards safe robots: Approaching asimov's 1st law," Ph.D. dissertation, Institut für Mensch-Maschine-Interaktion, Rheinisch-Westfälisch Technische Hochschule Aachen (RWTH Aachen), Aachen, Germany, 2011.

[2] A. Bauer, D. Wollherr, and M. Buss, "Human–robot collaboration: A survey," *Int. J. Humanoid Robot.*, vol. 5, no. 1, pp. 47–66, 2008.

[3] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mech. Mach. Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[4] T. Tadele, T. de Vries, and S. Stramigioli, "The safety of domestic robotics: A survey of various safety-related publications," *IEEE Robot. Autom. Mag.*, vol. 21, no. 3, pp. 134–142, Sep. 2014.

[5] B. Chandrasekaran and J. M. Conrad, "Human-robot collaboration: A survey," in *Proc. IEEE SoutheastCon*, 2015, pp. 1–8.

[6] S. Haddadin, "Physical safety in robotics," in *Formal Modeling and Verification of Cyber-Physical Systems*. New York, NY, USA: Springer, 2015, pp. 249–271.

[7] B. Gleeson, K. MacLean, A. Haddadi, E. Croft, and J. Alcazar, "Gestures for industry: Intuitive human–robot communication from human observation," in *Proc. 8th ACM/IEEE Int. Conf. Human–Robot Interact.*, 2013, pp. 349–356.

[8] E. Bicho, W. Erlhagen, L. Louro, and E. C. e Silva, "Neuro-cognitive mechanisms of decision making in joint action: A human–robot interaction study," *Human Mov. Sci.*, vol. 30, no. 5, pp. 846–868, 2011.

[9] C. Lenz *et al.*, "Joint-action for humans and industrial robots for assembly tasks," in *Proc. 17th IEEE Int. Symp. Robot Human Interact. Commun. (RO-MAN'08)*, 2008, pp. 130–135.

[10] M. E. Foster and C. Matheson, "Following assembly plans in cooperative, task-based human-robot dialogue," in *Proc. Semant. Pragmatics Dialogue (LONDIAL)*, 2008, p. 53.

[11] C. Lenz, M. Rickert, G. Panin, and A. Knoll, "Constraint task-based control in industrial settings," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS'09)*, 2009, pp. 3058–3063.

[12] A. Clodic, H. Cao, S. Alili, V. Montreuil, R. Alami, and R. Chatila, "Shary: A supervision system adapted to human–robot interaction," in *Experimental Robotics*, O. Khatib, V. Kumar and G. Pappas, Eds. New York, NY, USA: Springer, 2009, vol. 54, pp. 229–238.

[13] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, pp. 939–954, 2004.

[14] L. S. Homem de Mello and A. C. Sanderson, "AND/OR graph representation of assembly plans," *IEEE Trans. Robot. Autom.*, vol. 6, no. 2, pp. 188–199, Apr. 1990.

[15] M. Risler and O. von Stryk, "Formal behavior specification of multi-robot systems using hierarchical state machines in XABSL," in *Proc. AAMAS/Workshop Formal Models Methods Multi-Robot Systems*, Estoril, Portugal, 2008.

[16] L. S. H. De Mello and A. C. Sanderson, "A correct and complete algorithm for the generation of mechanical assembly sequences," *IEEE Trans. Robot. Autom.*, vol. 7, no. 2, pp. 228–240, Apr. 1991.

[17] T. L. De Fazio and D. E. Whitney, "Simplified generation of all mechanical assembly sequences," *IEEE J. Robot. Autom.*, vol. 3, no. 6, pp. 640–658, Dec. 1987.

[18] Y. Huang and C. G. Lee, "Precedence knowledge in feature mating operation assembly planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1989, pp. 216–221.

[19] A. Koc, I. Sabuncuoglu, and E. Erel, "Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph," *IIE Trans.*, vol. 41, no. 10, pp. 866–881, 2009.

[20] S. Parusel, S. Haddadin, and A. Albu-Schäffer, "Modular state-based behavior control for safe human–robot interaction: A lightweight control architecture for a lightweight robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011, pp. 4298–4305.

[21] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed., NJ, USA: Prentice Hall, 2003.

[22] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmüller, A. Albu-Schäffer, and G. Hirzinger, "Towards the robotic co-worker," in *Proc. Int. Symp. Robot. Res.*, Lucerne, Switzerland, 2009, pp. 261–282.

[23] M. Huber, M. Rickert, A. Knoll, T. Brandt, and S. Glasauer, "Human–robot interaction in handing-over tasks," in *Proc. 17th IEEE Int. Symp. Robot Human Interactive Commun. (RO-MAN'08)*, 2008, pp. 107–112.

[24] S. Haddadin, A. Albu-Schäffer, A. D. Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human–robot interaction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS'08)*, 2008, pp. 3356–3363.

[25] G. Michalos, S. Makris, L. Rentzos, and G. Chryssolouris, "Dynamic job rotation for workload balancing in human based assembly systems," *CIRP J. Manuf. Sci. Technol.*, vol. 2, no. 3, pp. 153–160, 2010.