Copyright Information

This is a post-peer-review, pre-copyedit version of the following paper

Darvish, K., Bruno, B., Simetti, E., Mastrogiovanni, F., & Casalino, G. (2018, August). Interleaved Online Task Planning, Simulation, Task Allocation and Motion Control for Flexible Human-Robot Cooperation. In 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN) (pp. 58-65). IEEE.

The final authenticated version is available online at:
https://doi.org/10.1109/ROMAN.2018.8525644

You are welcome to cite this work using the following bibliographic information:

BibTeX

```
@inproceedings{Simetti2018interleaved,
  title={Interleaved Online Task Planning, Simulation, Task
     Allocation and Motion Control for Flexible Human-Robot
     Cooperation},
  author={Darvish, Kourosh and Bruno, Barbara and Simetti, Enrico and
      Mastrogiovanni, Fulvio and Casalino, Giuseppe},
  booktitle={2018 27th IEEE International Symposium on Robot and
     Human Interactive Communication (RO-MAN)},
  pages={58--65},
  year={2018},
  organization={IEEE},
  doi={10.1109/ROMAN.2018.8525644},
  ISSN={1944-9437},
}
```

©2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Interleaved Online Task Planning, Simulation, Task Allocation and Motion Control for Flexible Human-Robot Cooperation

Kourosh Darvish[1], Barbara Bruno, Enrico Simetti, Fulvio Mastrogiovanni, Giuseppe Casalino

***Abstract*— Modern manufacturing paradigms introduce the need for robots able to naturally cooperate with humans in an unstructured and dynamic environment. In this article we extend FlexHRC, an architecture for flexible and collaborative manufacturing robots, with an online perception-simulation-planning framework that allows the robot to assess the status of the workspace, keeping track at all times of the stage at which the cooperative manufacturing process is, to identify its next action, to simulate it to check its feasibility and, as a consequence, to dynamically allocate tasks to itself or the human operator. We have tested the FlexHRC with a dual-arm manipulator cooperating with a person to assemble a table with one tabletop and four legs.**

I. Introduction

Consumer-driven markets have introduced a new paradigm to production, called *social manufacturing*, according to which consumers should be fully involved in the production process [1]. A flexible and agile production process is envisaged, and it is expected that frequent reconfigurations at the shop-floor and warehouse levels will be necessary to cope with diversified demands on tight schedules [1], [2]. Human-Robot Cooperation (HRC) is considered a very promising approach to deal with highly dynamic manufacturing processes, because it trades-off robot capabilities and the flexibility of human operators. The approach, however, also entails a number of challenges, including physical safety, psychological assessment of operators' well-being, human-centric design, natural and intuitive information exchange and communication, as well as robot autonomy and learning [2], [3], [4].

A natural and efficient interaction assumes different degrees of autonomy in robot decision making processes, and human-like communication capabilities. On the one hand, if simple tasks are to be performed by the collaborative robot, it should execute the associated actions making internal decisions without the need for human intervention. On the other hand, the robot should recognize at runtime and adapt to human operators' intentions using sensory data, while balancing the need to meet common goals and to limit the cognitive burden on the human side [5].

In this paper, we present a novel approach to human-robot cooperation in shop-floor settings aimed at increasing the online flexibility, the adaptation capabilities, and the controlled autonomy of collaborative robots, which is integrated in the FlexHRC architecture introduced in [6]. FlexHRC estimates future robot behaviors online, exploits such estimates to take decisions as far as cooperation is concerned, and allocates tasks to either humans or robots, reactively adapting to human operator decisions. The main contribution of the paper is two-fold: (i) an in-the-loop prediction of the outcome of robot actions via different, alternative simulations of robot behaviors aimed at minimizing the chance of task execution failures and at defining promising human-robot cooperation strategies, and (ii) a new dynamic robot action selection and task allocation method taking into account both humans and robots, specifically designed for HRC scenarios. Online or offline action planning and task allocation highly affect the overall cooperation efficiency, effectiveness, and flexibility. While working offline is computationally more efficient at the representation level, online use enhances flexibility and the overall success rate of the cooperation process.

In the literature, different task planning approaches have been proposed for HRC scenarios. The approaches described in [7], [8] generate different plans offline, and optimize a utility function to find the optimal sequence of actions and to perform task allocation. During online execution, humans and robots are constrained to follow the optimal plan while executing the cooperation task. Major issues arise when the cooperation process cannot be described in its entirety, or some relevant features of the environment are uncertain, or cannot be properly perceived by the robot or represented. In all these cases finding an optimal sequence of actions for allocated tasks cannot be but sub-optimal. Other approaches [9], [10], [6] allow for tuning the cooperation process as it unfolds, by enabling human operators only to decide how to progress, while a robot reactively adapts to human actions.

In a number of HRC scenarios, task allocation has been addressed as an optimization problem with constrained resources [11], [12]. These approaches optimize a *utility* metrics which estimates the performance as a function of the expected quality of task execution and the resources' cost [13]. Considered metrics include execution time, idle periods, task switching time, resources and human- or robot-related costs, reachability, as well as human factors such as the cognitive task load, and the psychological burden [11], [12], [14], [7]. The approaches in [7], [12], [6], [8] assume offline allocation, whereas others perform it online [11], [14]. It is noteworthy that offline task allocation is limited as far as the reliability is concerned, due to uncertainties and changes in the workspace and unpredictable human decisions.

The paper is organized as follows. Section II introduces the FlexHRC architecture. Section III describes the proposed solution for online task planning, simulation and task allo-

All the authors are with the Department of Informatics, Bioengineering, Robotics, and Systems Engineering, University of Genoa, Via Opera Pia 13, 16145, Genoa, Italy.

[1]Corresponding author's email: kourosh.darvish@edu.unige.it

cation. Section IV, describes the experimental scenario and discusses the experimental results. Conclusions follow.

II. System's Architecture

A. Rationale

Figure 1 shows the three levels of the FlexHRC architecture, namely *perception*, *action*, and *representation*. The perception level (in blue) is composed of three modules, namely *Object & Scene Perception*, *Human Action Recognition* (HAR), and *Knowledge Base*, which provide and store information about the state of the workspace and about human actions. The action level (in red) is responsible for managing the robot and consists of three modules, i.e., *Robot Execution Manager*, *Controller* and *Simulator*, the latter two being related to actions defined by the former and to be performed by the real or the simulated robot, respectively. The *Task Representation* and *Planner* form the representation level (in green), where the cooperation process is defined and handled.

B. Perception Level

1) Human Action Recognition: The module gets inertial data from wearable devices worn by human operators and models human actions in terms of gestures using *Gaussian Mixture Modeling* and *Gaussian Mixture Regression* over the inertial data. Online, it computes the Malahanobis distance between the continuous data stream and the available gesture models to classify operator motions, and therefore actions, providing the *possibility* of each modelled gesture to match the data stream [15]. Finally, as discussed in [6], the module checks the *possibility* patterns of all gestures to recognize the execution of an action when performed by an operator.

2) Object & Scene Perception: The module receives depth information related to the robot's workspace from an RGB-D sensor mounted on top of the robot, it applies a Euclidean metric for clustering the point cloud, and it uses the Random Sample Consensus (RANSAC) method to classify and tag objects considered as *primitive* shapes with a semantic label. For each classified shape, a number of relevant geometrical features to determine grasping poses are computed [16].

3) Knowledge Base: The module is a data structure that stores information related to the status of the workspace (i.e., the objects therein) and the robot.

C. Action Level

1) Robot Execution Manager: This module is responsible for turning high-level action commands, received from the *Planner*, into commands that the robot *Controller* can execute. Simple actions, such as *approaching an object*, can be performed by the *Controller* module directly, whereas more complex actions, such as *screwing*, are mapped to a number of simpler actions defined by the controller interface. Moreover, the *Robot Execution Manager* performs a symbolic-to-numerical mapping to the controller-level representation, and checks the success/failure of an executed action.

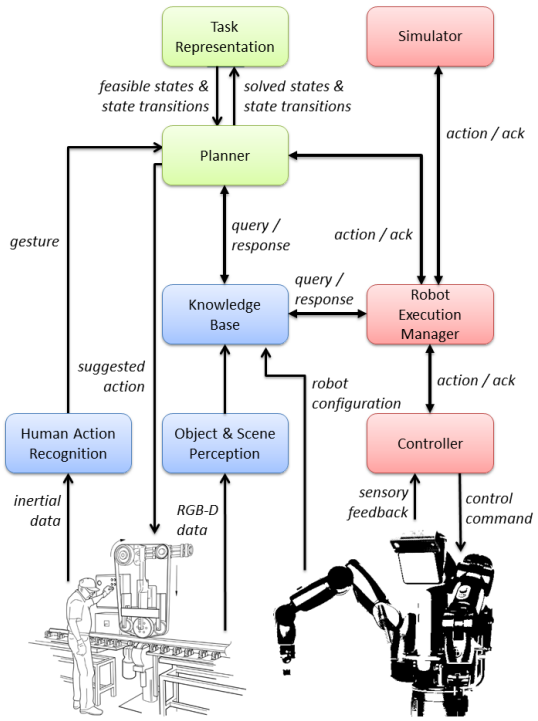

Fig. 1: The FlexHRC architecture: modules and data flow.

2) Controller: Given a certain target pose for the robot, a Task Priority framework controls robot motions at the kinematic level. The *Controller* module issues a sequence of prioritized optimization problems to find the most appropriate control signals. The scalar value $x_o(c)$ defines the control objective as a function of the configuration value vector c. The framework can take both *equality* and *inequality* control objectives into account. A control task is defined so as to accomplish each control objective. The reference rate $\dot{\hat{x}}_o$ for a control task is:

$$\dot{\hat{x}}_o \triangleq \gamma(x_o^* - x_o),\ \gamma > 0. \tag{1}$$

where the gain γ defines the desired convergence rate. The Jacobian relationship $J_o(c)$ maps the system velocity $\dot{y}$ vector to the task velocity scalar $\dot{x}_o$ by:

$$\dot{x}_o = J_o(c)\dot{y}. \tag{2}$$

For each control objective, we define an activation function $\alpha(x_o) = \alpha_o(x_o)$, which is a continuous sigmoid function such that $\alpha_o(x_o) \in [0, 1]$. The solution found by the Task Priority framework with p priority levels is a recursion of p minimization problems [17]. The interested reader can find a detailed description of the *Controller* module in [6].

3) Robot Simulator: The *Robot Simulator* module predicts the robot behavior in its workspace by simulating the robot's closed loop kinematic model, thus allowing for predicting the outcome (in terms of success/failure) of any given action before its actual execution. A simulation is marked as successful if the robot reaches the given goal within a predefined time, and it is marked as failed otherwise.

Although we simulate the robot's kinematic motion, during online execution there may be disturbances affecting robot motion, that may even lead the robot to instability. For the work described in this paper, we assume that the *Controller* module can compensate such disturbances. The output of the simulator includes a label indicating success or failure of an action, estimated execution time, suggested robot's trajectory, and the effort to perform the action.

The robot simulator solves a system of ordinary differential equations in the form:

$$\dot{X}(t) = F(X(t), U(X)). \tag{3}$$

from time t_0 and initial conditions X_0, to time t_{end}, where $X(t)$ is the vector of system states at each moment, and $U(X)$ is the control output vector, which is a function of states at each moment. To solve (3) we use the Runge-Kutta method, whereas to compute the control outputs we use the Task Priority based controller described above.

D. Representation Level

1) Task Representation: In order to semantically formalize the cooperation process between human operators and robots we employ AND/OR graphs. An AND/OR graph $G(N, H)$ consists of a set of nodes N and a set of hyper-arcs H. A node $n \in N$ represents a *state* of the cooperation, whereas a hyper-arc $h \in H$ represents a specific *transition* among states. In particular, a hyper-arc h connects a set of *child* nodes $N_C \subseteq N$ to a *parent* node $n_P \in N$. The relation between child nodes in a hyper-arc is the logic AND, while the relation between different hyper-arcs inducing on the same parent node is the logic OR. We assign to each node $n_i \in N$ a cost $c(n_i)$, and to each hyper-arc h_i a cost $c(h_i)$, according to criteria and metrics related to the human-robot cooperation. A *path* P_i in an AND/OR graph is a set of nodes and hyper-arcs, such that $P_i = \{n_i, \ldots, n_j, h_l, \ldots, h_m\}$, connecting the set of leaf nodes $N_L \subseteq N$ to the root node $n_r \in N$. Leaf nodes are nodes without child nodes. Each path P_i is associated with a cost $c(P_i)$, such that:

$$(P_i) = \sum_{k_1=i}^{j} c(n_{k_1}) + \sum_{k_2=l}^{m} c(h_{k_2}). \tag{4}$$

A hyper-arc $h_i \in H$ corresponds to a sequence of k ordered actions $A(h_i) = (a_j, \ldots, a_k)$ to be executed by the human operator or the robot to reach a given node. Most nodes represent stages of the cooperative process (e.g., sub-assemblies), but some are robot-specific, for example defining maintenance/setup operations or workspace checks. An action a_i is initially *undone*. When the human or the robot executes action a_i, the action status changes to *done*. Initially, all the actions associated with hyper-arcs in a graph are *undone*. A hyper-arc is *solved* if all the actions in a hyper-arc are marked as *done* in a certain order. Similarly, a node n_i is marked as *solved* if all the actions associated with it are *done* or they constitute an empty set. A node n_i is *feasible* if there exists a solved hyper-arc $h_j \in H$ for which $parent(h_j) = n_i$ holds; otherwise, if there is no hyper-arc inducing on node n_i, it is unfeasible. All *leaf* nodes in an AND/OR graph are feasible when the cooperation process starts. A hyper-arc $h_i \in H$ is feasible if all its child nodes are solved. Once a hyper-arc is solved, all of its child nodes and all the hyper-arcs connected to them become unfeasible. At all times, the set N_f of feasible nodes and the set H_f of feasible hyper-arcs are defined. An AND/OR graph is *solved* if the *root* node is *solved*.

The AND/OR graph traversal algorithm includes two phases, performed respectively offline and online. The offline phase loads the description of the AND/OR graph defining a cooperation process and initializes all nodes, hyper-arcs and paths. In the online phase, when the AND/OR graph is queried with the last set N_s of solved nodes and set H_s of solved hyper-arcs, the algorithm updates the status of all nodes and hyper-arcs, as well as the cost of the paths, and provides the sets N_f and H_f of currently feasible nodes and hyper-arcs. The offline and online phases have been described in [6].

2) Planner: The *Planner* module maps the states and state transitions to the corresponding sets of ordered actions. Using the workspace information accumulated in the *Knowledge Base* and the predictions on the robot's future behaviour done by the Simulator, the module grounds the parameters of the chosen action and assigns it to the robot or human. The planning process, which constitutes the main contributions of this article, is described in Section III.

III. Online Planning, Simulation and Task Allocation

The Planner receives a set of feasible states and state transitions $S_f = \{N_f, H_f\}$ with their associated costs from the AND/OR graph, and among them it selects the one with the minimum path cost s_i to follow. To solve s_i the robot or the human should execute the associated ordered sequence of actions $A_i = (a_1, ..., a_n)$ [6]. In the current version of the *Planner* module, an action a_i is defined by the 5-ple:

$$\langle Params, Precond, Effects, TempCond, Agents \rangle,$$

where $Params$ is a set of parameters (e.g., required precision, or possible grasping poses), $Precond$ includes all action preconditions (i.e., state features for a_i to be executed), $Effects$ is the set of action effects (i.e., state features modified by a_i), $TempCond$ is a set of *temporary* conditions, and $Agents$ is the set of agents ag_i responsible to perform the action, i.e., human operators and/or robots. For an action a_i, if the number of agents is more than one, then a_i is labelled as a *joint action*. In a joint action, all the responsible agents must have a consensus on a set of constraints, which we refer to as a *convention*, to perform it.

The current version of FlexHRC allows the cooperative robot to examine the actions' execution proactively using the robot simulator (first contribution). Moreover, if the actions in A_i are not assigned to a set of collaborators or the parameters are not grounded, the *Planner* creates a data structure online to examine all possible combinations. Later, it examines all the combinations by simulating them

and finds the utility value J, a performance measure of A_i actions execution, for all of combinations online. The *Planner* selects the set of collaborators and grounds the parameters of A_i actions such that utility value is optimized (second contribution). If all the combinations of the A_i associated with the state s_i fail to execute in the simulation, the *Planner* alters the feasibility of s_i and performs the planning steps again proactively (re-planning).

The *Planner* module workflow is organized in two phases, the first offline and the second online. The offline phase loads all action definitions, their parameters, the semantic knowledge associated with the capability of agents to perform different actions (i.e., either alone or jointly), and, from the description of the cooperative process given in the form of an AND/OR graph, the action sequences $A(h_i)$ associated with each hyper-arc.

During the online phase, as the human-robot cooperation process unfolds, the sequences of actions corresponding to currently feasible states and state transitions are evaluated. More precisely, thanks to the predictions of robot behaviors given by the simulations, the *Planner* module maximizes the performance of the human-robot cooperation process in terms of actual costs $c(P_i)$ associated with path P_i in the AND/OR graph as shown in (4). The module receives the set of feasible states (i.e., graph nodes) and state transitions (i.e., hyper-arcs) from the *Task Representation* module, it finds the sequence of actions for human operators and robots that minimizes the cost (4) of the cooperation (i.e., a specific feasible hyper-arc in the graph), it assigns actions to them and, upon receiving perceptual information, it updates the feasibility status of states and states transitions.

In classical action planning approaches, preconditions and effects model how actions modify symbolic states, once executed [18]; in FlexHRC, instead, the sequence of actions $A(h_i)$ associated with an hyper-arc h_i and the parameters of each actions are defined beforehand. In fact, in a realistic, goal-oriented HRC scenario it is necessary to increase a robot's capability in operating both proactively and *without ambiguity* from the perspective of human operators. To this aim, the *Planner* module is expected to decide whether a human operator or a robot should perform an action, whereas the cooperative robot behavior should maximize the overall objective function (by minimizing the cost of a cooperation path P_i). The allocation of actions to either human operators or robots, as well as the definition of the most appropriate instances of action parameters, are done in two possible ways. In the simplest case, the cooperation designer allocates possible agents to actions, or defines a subset of action parameters offline. The *Planner* thus only has to fetch and use such information. In the more complex case, multiple agents or parameter instances are associated with an action and the *Planner* is responsible for identifying the most effective combination along the currently optimal cooperation path (i.e., the path minimizing the current overall cooperation cost). To this aim, the *Planner* fetches parameter values from the *Knowledge Base* module and simulation results from the *Simulator* module.

The interplay between the *Planner* and the *Simulator* modules is of particular relevance for the human-robot cooperation process in the FlexHRC architecture, especially for dynamic task allocation, parameter instantiation and proactive recovery from failures.

Concerning the first two goals, consider the action *Put down screwdriver on table*, which does not specify which screwdriver and table are involved; in principle, there may exist more than one screwdriver and table in the robot workspace, which may ground such a command. On the one hand, the robot should consider all the possible instances of *screwdriver* and *table* to find the optimal and complete plan to address such a command. On the other hand, there might be more than one agent *capable* of performing the associated *put down* action, and therefore they should coordinate to determine which one executes it. In fact, if the number of instances of screwdrivers and tables are m and n, respectively, and k agents (human operators or robots) can perform the action, there are $m \times n \times k$ possible realization of the *put down* action.

An example of the analysis done by the *Simulator* in the case of action *Put down screwdriver on table* is shown in Figure 2, where it is assumed that in the workspace there are two tables $\{Tab1, Tab2\}$, two screwdrivers $\{SD1, SD2\}$, and two robot agents $\{R1, R2\}$ that can perform all the steps required by the action. In the case of Figure 2, there are $2 \times 2 \times 2 = 8$ possible combinations to ground the parameters and allocate the tasks to the agents, represented as 8 branches of the so-called *simulation tree*. We select the branches to simulate using Breadth-first search algorithm. When all the branches are simulated, we compare them according to a predefined utility function. In this work, it is simply execution time, i.e.:

$$J_j = \sum_{k=0}^{k=K} t_{j,k}. \tag{5}$$

where J_j is the utility value of branch j, K is the number of steps required by the action to simulate, and $t_{j,k}$ is the time it takes to perform step k of branch j. On the basis of (5), the *Simulator* module determines the most appropriate and realistic combination of screwdriver, table and robot minimizing the overall cooperation cost, that the *Planner* compares with the a-priori defined best combination involving a human operator performing the action, to finally allocate it to one agent or the other.

If an action execution fails in the simulation tree, the *Planner* deletes the branch associated to that action. For example, if $Approach(SD1, R1)$ fails in the simulation, the *Planner* deletes branches (1) and (2). If $Approach(Tab1, R1)$ fails to perform, it deletes only branch (1). If all the branches are deleted, the simulations imply the robots cannot execute the state transition. Therefore the *Planner* sets the feasibility value of the state transition associated to the optimal path P_i to $false$. Later the *Planner* get updated, and generates new simulation tree for the feasible state transition associated with a new optimal path P_o.

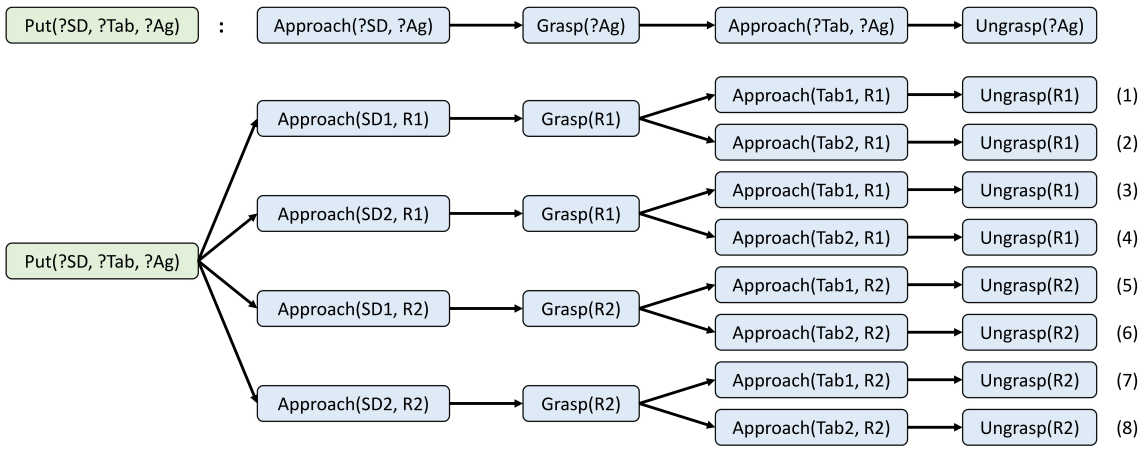

Fig. 2: The simulation tree used by the *Simulator* module for online task allocation and parameter grounding. SD stands for a screwdriver, Ag stands for an agent, and Tab stands for a table.

Failures occur whenever a robot does not succeed in executing a given command because of unexpected conditions of the workspace, uncertainty, or the impossibility to meet specific kinematics constraints or safety requirements. In this case, the *Simulator* module can proactively determine the feasibility of actions before they are actually executed and it can suggest the *Planner* to allocate actions to human operators if their outcome is uncertain, or inefficient, given the robot capabilities and the current status of the workspace.

IV. Experimental Evaluation

A. Setup

The equipment adopted for the evaluation of the proposed system includes a dual-arm seven DoF Baxter robot for cooperative manipulation, an LG G watch R (W110) smartwatch to acquire the acceleration data from the right wrist of the person, an LG G3 smartphone as a communication bridge between the smartwatch (Bluetooth) and the workstation (WiFi), and a Microsoft Kinect to acquire RGB-D data of the workspace. The whole FlexHRC architecture shown in Figure 1 is based on ROS Indigo and it runs on a workstation with Ubuntu 14.04 LTS 64-bit, 16 GB RAM, and Intel Core i7-4790, 3.60GHz CPU.

To test the FlexHRC architecture, we consider the cooperative assembly of a table consisting of one tabletop and four legs. The corresponding AND/OR graph includes 256 paths from the leaves (the 4 legs and tabletop, randomly placed in the robot workspace) to the root (the assembled table). The cooperation involves three agents: *human*, *robot left arm*, and *robot right arm*. The human can perform the actions *pick up*, *screw*, and *put down*. The robot can perform the actions *approach*, *transport*, *grasp*, *ungrasp*, *screw* and *unscrew*. The action *transport* can be performed singly or jointly by the two robot arms. The human obtains information about the cooperation, such as the robot and the human symbolic actions, in natural language from the robot display. An action is defined as *failed*, both in reality and in simulation, if it is not successful within 30 seconds.

TABLE I: Required planning, simulation, task representation, human, and robot time for the table assembly task.

$\mathcal{P}$	$time\ [s]$	$time\ [\%]$
Planning	0.01	0.00
Simulation	6.28	1.77
Task Representation	0.57	0.16
Human Action	97.08	27.35
Robot Action	251.00	70.72

B. Results

Figure 3 illustrates the human-robot cooperation for the table assembly task. For this task, the offline phase of the planning takes 0.17 seconds, while the online cooperation takes 354.95 seconds for successful table assemble. Table I reports the time allocation for the FlexHRC modules. As the Table shows, not only the *Planner* and *Task Representation* modules require very little time to identify at all stages feasible and optimal actions (taking less than 1% of the overall execution time), but also the *Simulator* is very efficient: in the test, it simulates a total of 3720 seconds of robot actions in less than 7 seconds.

Figure 3 (1) shows the initial configuration of the workspace: the robot initializes the cooperation state, receives the first feasible state transitions from the *Task Representation* module and simulates them to allow the *Planner* to optimally allocate them to the agents. Figure 4 shows the configuration values in simulation (dashed lines) and reality (solid lines) for the *robot left arm* for the action *Approach(Tabletop)*, corresponding to scenes (1-3) of Figure 3. Figure 4 demonstrates that the simulation can reliably predict the robot behavior in case of small disturbances. Scenes (3-8) of Figure 3 show the placing of the tabletop in its final position. The tabletop weighs 1.4 Kg, and, to test a condition of *high disturbance*, this fact is purposefully not considered in the simulation. Figure 5 shows the simulation and reality results for the *left arm* tool frame. As expected, the controller is able to compensate the high disturbances on

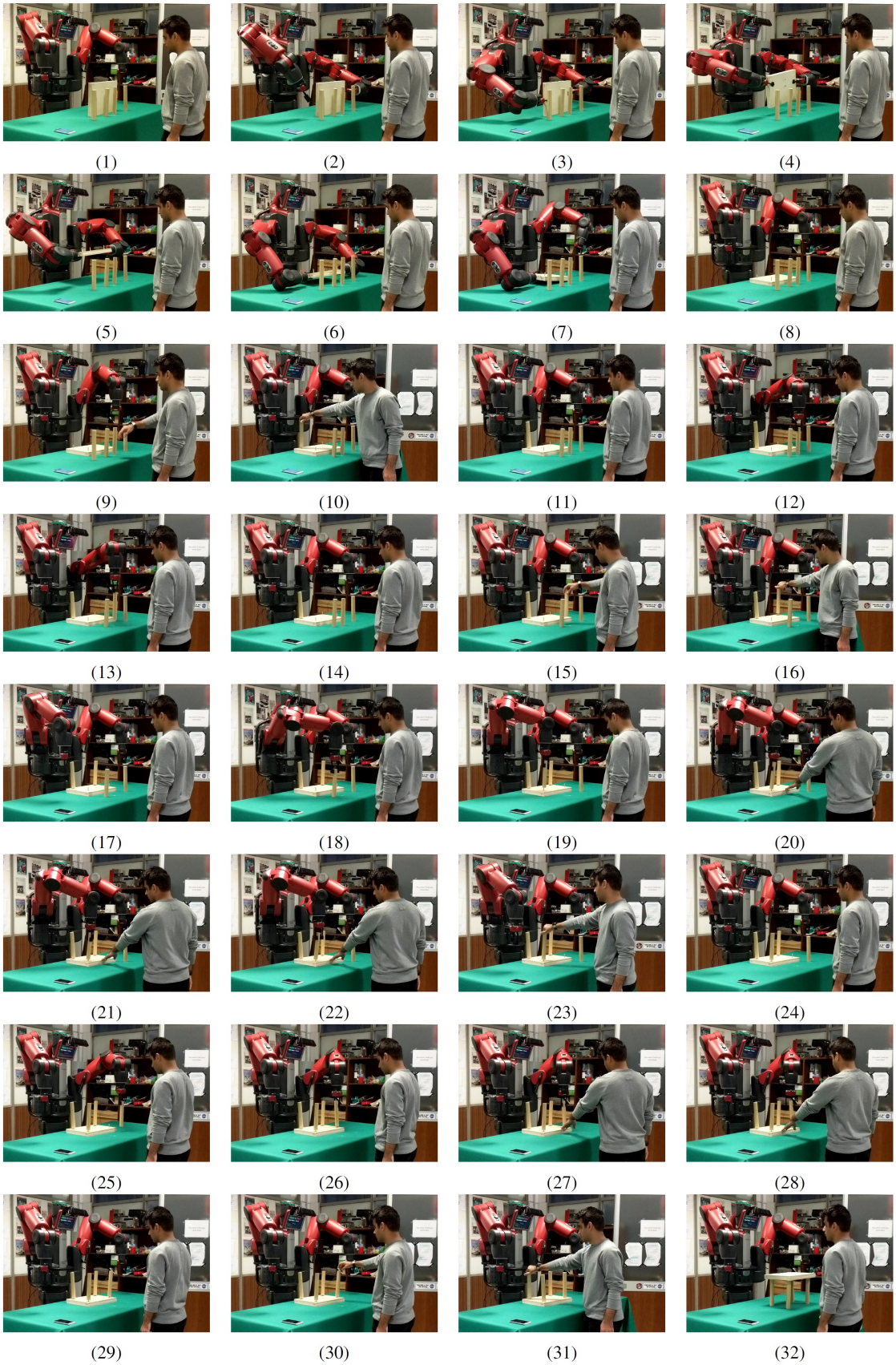

Fig. 3: The sequence of actions associated to the table assembly task.

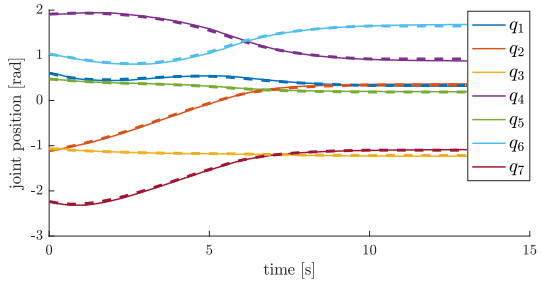

Fig. 4: The real (solid line) and simulated (dashed line) joints positions of the *robot left arm* with low disturbance.

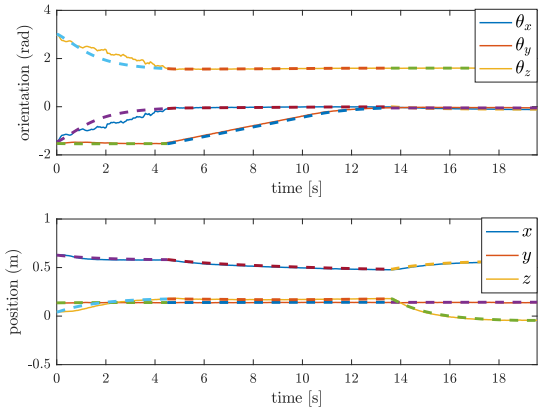

Fig. 5: The real (solid line) and simulated (dashed line) tool frame orientation and position of the *robot left arm* with high disturbance.

the robot while performing the joint *transportation* action.

Once the tabletop is in its final position, the state of the cooperation is updated and the robot is given the new feasible states and state transitions, which all lead to the placement of a leg. Since this scenario assumes two robot agents and four legs in the workspace, the *Simulator* generates and tests eight simulation branches. On the basis of the simulation results, the *Planner* assigns the action to the *robot left arm*, but, as the robot begins moving, the human decides to perform an action and connect one of the legs to the tabletop (scenes (9-10) of Figure 3). This situation tests the planning flexibility. Whenever the human interrupts a robot action, the robot goes to its initial pose. At scene (11) one leg and the tabletop are connected and the robot, upon analysing the workspace, again updates its *Task Representation*.

The *Simulator* analyses the new feasible actions, generating six simulation branches (two agents and three remaining legs). The simulations fail: none of the robot arms is able to place any of the legs on the tabletop, given their current placement within the workspace (see Figure 6). This situation tests the ability of the FlexHRC architecture to recover from failures. As a result of the failure notification, the optimal state transition (hyper-arc in the AND/OR graph) becomes *unfeasible*, the path cost is updated and a new optimal path is identified by the *Task Representation* module, which, in

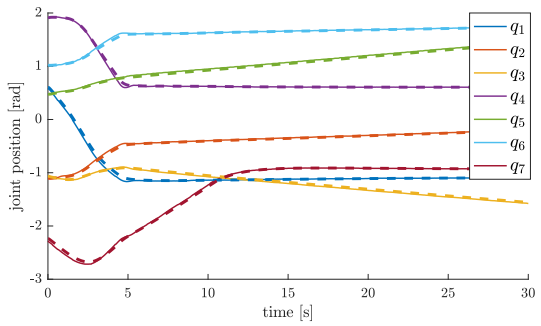

Fig. 6: The real robot (solid line) and simulated robot (dashed line) joints position of the *left arm* for a failed action: some of the joint positions do not converge to constant values.

particular, requires the robot to move a leg in front of the human to let him screw it to the tabletop. The new set of feasible states and state transitions is given to the *Planner* and, consequently, to the *Simulator*. Since the simulation proves successful, the action is actually executed (scenes (12-13) of Figure 3). Having placed a leg in front of the human operator, in scene (14) the robot asks the person to connect a leg to the tabletop to proceed along the cooperative process (scenes (15-16) of Figure 3).

In scene (17) the robot again uses its perception modules to update its representation of the cooperation and identify the new feasible states and state transitions. Scenes (17-19) show the *approaching* and *transporting* of a leg by the *robot right arm*. Scenes (20-21) show the *screwing* of the leg to the tabletop by the robot, while scene (22) shows the subsequent *ungrasping* and *unscrewing* actions. Scene (23) shows the human operator again interrupting the robot to complete the screwing of the leg. Once three legs are connected to the tabletop (scene (24) of Figure 3), the *Planner* assigns the placement of the last leg to the *robot left arm* (scenes (24-29) of Figure 3). The human operator assists the robot by holding the tabletop and later controls the stability of the connection between the legs and the tabletop, before finally turning the assembled table in the correct orientation (scenes (30-32) of Figure 3), which signals the successful end of the cooperative process.

V. Conclusions

In this article we propose and examine a framework for dynamic task planning, simulation and allocation for cooperative robots. The framework, which extends the previously proposed FlexHRC architecture, allows both human operators and robots to deviate from the optimal plan, ensuring that all other agents adapt their actions accordingly.

In particular, the paper presents an interleaved approach to perception, task representation, planning and simulation, which allows a robot to define, at all times, the current situation of the cooperative task and its workspace, plan its optimal action, simulate its execution and, upon the outcome of the simulation, execute it in accordance with the results of the simulation or re-plan.

These features are tested in a real cooperative scenario, in which a dual-arm manipulator cooperates with a human operator to assemble a small table composed of a tabletop and four legs. The experiment proves that the proposed architecture is able to dynamically simulate actions, allocate tasks and recover from unexpected events by re-planning.

REFERENCES

[1] B. Esmaeilian, S. Behdad, and B. Wang, "The evolution and future of manufacturing: A review," *Journal of Manufacturing Systems*, vol. 39, pp. 79–100, 2016.

[2] S. Kock, T. Vittor, B. Matthias, H. Jerregard, M. Kllman, I. Lundberg, R. Mellander, and M. Hedelind, "Robot concept for scalable, flexible assembly automation: A technology study on a harmless dual-armed robot," in *Proceedings of the 2011 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, Tampere, Finland, May 2011.

[3] P. A. Lasota, T. Fong, and J. A. Shah, "A survey of methods for safe human-robot interaction," *Foundations and Trends® in Robotics*, vol. 5, no. 4, pp. 261–349, 2017.

[4] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and trends in human-computer interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[5] T. Meneweger, D. Wurhofer, V. Fuchsberger, and M. Tscheligi, "Working together with industrial robots: Experiencing robots in a production environment," in *Proceedings of the 2015 IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Kobe, Japan, August 2015.

[6] K. Darvish, F. Wanderlingh, B. Bruno, E. Simetti, F. Mastrogiovanni, and G. Casalino, "Flexible human-robot cooperation models for assisted shop-floor tasks," *Mechatronics*, vol. 51, pp. 97–114, 2018.

[7] L. Johannsmeier and S. Haddadin, "A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 41–48, January 2017.

[8] S. Lemaignan, M. Warnier, E. A. Sisbot, A. Clodic, and R. Alami, "Artificial cognition for social humanrobot interaction: An implementation," *Artificial Intelligence*, vol. 247, pp. 45–69, 2017.

[9] K. P. Hawkins, S. Bansal, N. N. Vo, and A. F. Bobick, "Anticipating human actions for collaboration in the presence of task and sensor uncertainty," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[10] S. J. Levine and B. C. Williams, "Concurrent plan recognition and execution for human-robot teams." in *Proceedings of 2014 International Conference on Automated Planning and Scheduling (ICAPS)*, Portsmouth, USA, June 2014.

[11] F. Chen, K. Sekiyama, F. Cannella, and T. Fukuda, "Optimal subtask allocation for human and robot collaboration within hybrid assembly system," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 4, pp. 1065–1075, October 2014.

[12] P. Tsarouchi, G. Michalos, S. Makris, T. Athanasatos, K. Dimoulas, and G. Chryssolouris, "On a human-robot workplace design and task allocation system," *International Journal of Computer Integrated Manufacturing*, pp. 1–8, 2017.

[13] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[14] T. R. A. Giele, T. Mioch, M. A. Neerincx, and J.-J. C. Meyer, "Dynamic task allocation for human-robot teams," in *Proceedings of the 2015 International Conference on Agents and Artificial Intelligence (ICAART)*, vol. 1, Lisbon, Portugal, January 2015.

[15] B. Bruno, F. Mastrogiovanni, A. Saffiotti, and A. Sgorbissa, "Using fuzzy logic to enhance classification of human motion primitives," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, A. Laurent, O. Strauss, B. Bouchon-Meunier, and R. R. Yager, Eds. Montpellier, France: Springer International Publishing, 2014, pp. 596–605.

[16] L. Buoncompagni and F. Mastrogiovanni, "A software architecture for object perception and semantic representation," in *XIV AI*IA Symposium on Artificial Intelligence, AIRO Workshop*, 2015, pp. 116–124.

[17] E. Simetti and G. Casalino, "A novel practical technique to integrate inequality control objectives and task transitions in priority based control," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 877–902, apr 2016.

[18] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, "Rosplan: Planning in the robot operating system." in *Proceedings of the 2015 International Conference on Automated Planning and Scheduling (ICAPS)*, Jerusalem, Israel, 2015, pp. 333–341.