Optimal Task and Motion Planning and Execution for Multiagent Systems in Dynamic Environments

Marco Faroni, Alessandro Umbrico, Manuel Beschi, *Member, IEEE*, Andrea Orlandini, Amedeo Cesta, and Nicola Pedrocchi

***Abstract*—Combining symbolic and geometric reasoning in multiagent systems is a challenging task that involves planning, scheduling, and synchronization problems. Existing works overlooked the variability of task duration and geometric feasibility intrinsic to these systems because of the interaction between agents and the environment. We propose a combined task and motion planning approach to optimize the sequencing, assignment, and execution of tasks under temporal and spatial variability. The framework relies on decoupling tasks and actions, where an action is one possible geometric realization of a symbolic task. At the task level, timeline-based planning deals with temporal constraints, duration variability, and synergic assignment of tasks. At the action level, online motion planning plans for the actual movements dealing with environmental changes. We demonstrate the approach's effectiveness in a collaborative manufacturing scenario, in which a robotic arm and a human worker shall assemble a mosaic in the shortest time possible. Compared with existing works, our approach applies to a broader range of applications and reduces the execution time of the process.**

***Index Terms*—AI-enabled robotics, autonomous agents, collaborative robotics, heterogeneous multiagent systems, planning, scheduling and coordination, task and motion planning (TAMP).**

I. Introduction

HUMAN–ROBOT collaboration (HRC) boosts the flexibility of manufacturing processes, although the inefficient coordination between humans and robots often jeopardizes productivity [1]. From a planning perspective, efficiency in HRC is tied to different intertwined problems. First, the system should find a suitable sequence of operations (task planning), assign them to the agents (task assignment), and schedule their execution (scheduling). At run time, the execution of the operations should be adapted to the human and robot's state (motion planning and replanning). All these steps should also consider the variability of the duration and good (or bad) synergy of simultaneous collaborative operations. The complexity of the overall planning problem limits the effectiveness of existing methods in real-world scenarios, and standardized shared approaches to task and motion planning (TAMP) are still to come.

This article proposes a tiered approach for multiagent systems, interleaving task planning, scheduling, assignment, and action planning. The method addresses the tasks' temporal and geometric uncertainty by decoupling the abstract representation of the task from all its possible realizations. Timeline-based planning reasons on abstract tasks, while online action planning optimizes their geometric implementations. Compared with existing methods, our approach deals with various real-world problems and reduces execution and idle times.

A. Related Works

This article deals with interleaving task planning and motion planning and how to consider human behaviors in this process. Existing methods address the first aspect by following the combined TAMP paradigm [2]. Usually, TAMP provides a task planner able to reason geometrically through calls to a motion planning algorithm. In such a hierarchical approach, a task planning algorithm finds a feasible sequence of actions, and a motion planner checks for geometric feasibility [3], [4], [5], [6], [7]. Most TAMP methods focus on the feasibility of the plan rather than its optimality (except for a few exceptions [8], [9]). Few works address temporal planning to consider task duration [10].

HRC-oriented works usually focus on subproblems, such as scheduling human and robot actions [11], [12], [13], or cooperative planning at a symbolic level [14], [15], [16]. Few works address TAMP by explicitly modeling the human agent [17], [18]. For example, [19] and [20] proposed a hierarchical agent-based task planner, where complex tasks are decomposed into simple actions. The method improves the collaborative experience by considering human preferences as social costs, but throughput-oriented objectives are not considered [21], [22]. In manufacturing-oriented methods, [23]

Manuscript received 23 May 2022; revised 13 September 2022 and 18 March 2023; accepted 25 March 2023. This work was supported in part by the ShareWork Project (H2020, European Commission) under Grant 820807. The work of Alessandro Umbrico, Andrea Orlandini, and Amedeo Cesta was supported in part by the ROXANNE Project (H2020, ROSIN) under Grant 732287. This article was recommended by Associate Editor Z. Ju. *(Corresponding author: Alessandro Umbrico.)*

Marco Faroni is with the Department of Robotics, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: mfaroni@umich.edu).

Alessandro Umbrico, Andrea Orlandini, and Amedeo Cesta are with the Institute of Cognitive Sciences and Technologies, National Research Council of Italy, 00185 Rome, Italy (e-mail: alessandro.umbrico@istc.cnr.it andrea.orlandini@istc.cnr.it amedeo.cesta@istc.cnr.it).

Manuel Beschi is with the Dipartimento di Ingegneria Meccanica e Industriale, University of Brescia, 25121 Brescia, Italy, and also with the Institute of Intelligent Industrial Technologies and Systems, National Research Council of Italy, 20133 Milan, Italy (e-mail: manuel.beschi@unibs.it).

Nicola Pedrocchi is with the Institute of Intelligent Industrial Technologies and Systems, National Research Council of Italy, 20133 Milan, Italy (e-mail: nicola.pedrocchi@stiima.cnr.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2023.3263380.

Digital Object Identifier 10.1109/TCYB.2023.3263380

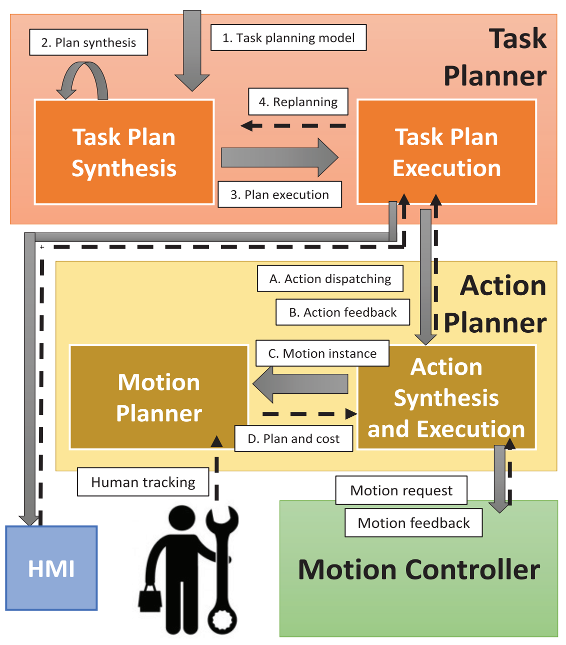

Fig. 1. Proposed hierarchical framework.

optimizes the ergonomics of the human worker by using an online workflow scheduler. Darvish et al. [24] and [25] proposed a TAMP framework for planning and executing tasks using first-order logic graphs. A contingent-based approach was proposed in [26] and [27] to deal with uncertainty on the outcome of actions.

The approaches above focus on finding feasible plans and do not consider: 1) process throughput; 2) temporal constraints and uncertainty; and 3) human–robot synergy. Regarding 1) and 2), timeline-based task planning [28] has proved to be a powerful approach in many real-world applications [29], [30]. The value of this approach consists of integrating planning and scheduling in a unified reasoning framework, making decisions about what actions to perform and when. This approach can also model the *features' controllability*, that is, the planner knows whether it can control the task's beginning or the end [31]. Timelines were applied to HRC in [32], but the integration with motion planning was inefficient since it reasoned at a low level of abstraction (point-to-point movements). Consequently, motion plans were precomputed, hindering the flexibility of the approach in dynamic environments.

B. Contribution

As shown in Fig. 1, we propose a hierarchical planner where the higher layer reasons over symbolic tasks, optimizing their order, scheduling, and assignment under temporal constraints and duration variability. At the same time, the lower level turns the symbols into robot motions, selecting the optimal task execution among all the possible alternatives.

Our formulation relies on the definition of ACTIONS as a set of instances of a TASK (Section II). Such formulation explicitly maps a symbolic TASK to all its possible geometric realizations. At the task planning level, this formulation considers the duration uncertainty of the tasks, temporal constraints, and the possible synergy of simultaneous tasks performed by the different agents. At the action level, instead, the robot plans and executes its motions based on the current context.

Then, we propose algorithms to solve the proposed task and action planning problems (Section III). On the one hand, we convert the optimal task planning problem into a multiobjective problem—which constitutes a novelty in timeline-based planning—and use the notion of Pareto optimality to optimize the coupling of simultaneous tasks. On the other hand, we make the optimal action planning problem tractable online by converting it into a multigoal motion planning problem that can be solved with efficient off-the-shelf algorithms.

Compared to previous works, our formulation is robust to temporal and spatial uncertainty typical of hybrid multiagent systems (e.g., HRC). We assess the broad applicability of the approach and its superiority to existing works qualitatively and experimentally (Sections IV and V), showing that our approach applies to a broader range of applications and reduces execution and idle times of the process. A video of the experiments is attached to this article.

II. Task and Action Planning Formalization

A. Definitions and Approach at Glance

Our approach builds on the following definitions:

Definition 1 (Worker): It is an agent that performs a job assigned by the system. It can be a human worker or a robot.

Definition 2 (Task): It is a step of the production process performed by a worker. TASKS do not provide geometrical information on their execution, as they model an operation at a higher level of abstraction. Each TASK might be performed in several ways, making its duration a random variable rather than a scalar variable.

Definition 3 (Robot Configuration): It is a vector collecting the joint positions and the auxiliaries' state.[1]

Definition 4 (Robot Movement): It is a discrete change of the ROBOTCONFIGURATION. Each ROBOTMOVEMENT might be realized by an infinite number of possible trajectories that satisfy the physical constraints of the robot.

Definition 5 (Action): An ACTION is a sequence of ROBOTMOVEMENTS. It is a feasible implementation of a TASK. Each ROBOTMOVEMENT is defined by the ACTION parameters.[2]

The proposed hierarchical approach is shown in Fig. 1. The task planner uses a planning model of the process to search for a feasible, possibly optimal sequence of TASKS. The planning

[1]Example: consider a 2-axes mechanism with a gripper. The ROBOT CONFIGURATION is a vector of length equal to three, where the first two components are the joint positions, and the third one is the gripper state.

[2]Example: TASK "screw bolt A" corresponds to different ACTIONS, each composed by the same sequence of ROBOTMOVEMENTS: "open the gripper," "move to grasp point X" (where X is the position of a particular screwdriver), "close the gripper," "move to Y" (where Y is the screwing position for bolt A), "screw bolt A." Each ACTION differs in the value of X, Y, A. For example, multiple suitable screwdrivers might be available in different locations.

model specifies the TASKS to be performed with temporal and allocation constraints. The Task Plan Execution module dispatches the target TASK to the workers. The Action Planner converts each TASK into a set of feasible ACTIONS. Then, it chooses the best ACTION among the feasible ones and sends the motion plan to the robot controller. If the Action Planner fails at planning or executing (e.g., no trajectory could be computed, or task execution fails at runtime), the Task Planner replans according to the Sense-Plan-Act paradigm [33].

B. Model Formalization

The following elements describe a collaborative process:

$\mathcal{W}$	$\{H, R\}$ is the set of workers, *that is*, a human and a robot;
$\mathcal{P}$	$\{p_i\}$ is the set of production targets;
$\mathcal{T}$	$\{t_j^{p_i}\}$ is the set of TASKS necessary to carry out a production target p_i;
$\mathcal{A}$	$\{a_j\}$ is the set of ACTIONS;
D	$\mathcal{T} \to \mathbb{R}^2$ is the duration function that associates a TASK with an interval $[d_{\min}, d_{\max}]$;
T	$\mathcal{T} \to \mathcal{T}$ is the transition function that defines valid transitions among TASKS;
γ	$\mathcal{T} \to \{c, pc, uc\}$ is the controllability tag. The tag is "controllable," "partially controllable" or "uncontrollable" if the system can decide the execution start and end, only the start or neither of the TASK a worker performs.
F	$\mathcal{T} \to \{\{H\}, \{R\}, \{H, R\}\}$ is a function that defines for each TASK which worker can execute it;
S	$\mathcal{A} \to \mathcal{T}$ is a function that maps each ACTION to its corresponding TASK.
SV	(V, T, D, γ) is a "state variable" that describes the behaviors of a domain feature that is represented by the set of tasks $V = \{v_i\} \in \mathcal{T}$. Such a set gathers all the valid TASKS that can be executed over time for that specific feature. T, D, and γ are defined as above.
x	$(v, [e, e'], [d, d'], \gamma(v))$ is a "token" where $v \in V$, $[e, e']$ is end-time interval, and $[d, d'] = D(v)$ is the duration interval for task v.
FTL_{SV}	$\{x_j\}$ is a flexible timeline of a state variable SV representing a temporal sequence of tokens x_j.

Solving a collaborative TAMP problem consists of identifying a task plan, a temporal schedule, and an assignment of the tasks considering that each task can be realized by a set of actions and each movement composing an action can be executed by an infinite set of trajectories. Each task plan is modeled through flexible timelines of tokens of state variables.

C. Task Planning Model for Human–Robot Cooperation

Given the notation in Section II-B, we define the production goals, each worker's possible behaviors, and the synchronization rules to model a human–robot scenario. Refer to [34] and [35] for a description of the formalization approach.

First, consider a set of high-level production targets $V^p = \{p_i\}$, where $p_i \in \mathcal{P}$. Each p_i can be further associated with a set $V^{p_i} = \{t_j^{p_i}\}$ that gathers TASKS to carry out p_i, where $t_j^{p_i} \in \mathcal{T}$.[3] Second, we consider a generic worker $w_k \in \mathcal{W}$ that may implement some of the needed tasks in V^{p_i}, and denote by $V^{w_k} = \{t_m^{p_i,w_k}\} \subset V^{p_i}$ the subset of tasks $t_j^{p_i}$ that a worker w_k can do according to F.

In addition to the set of tasks, we must define tasks' precedence constraints, task duration, and controllability.

To gather all this information, we denote by $SV^p = (V^p, T^P, D^p, \gamma^p)$ the production state variable associated with the high-level production targets $V^p = \{p_i\}$ and by $SV^{p_i} = (V^{p_i}, T^{p_i}, D^{p_i}, \gamma^{p_i})$ the production state variables associated with the production tasks $V^{p_i} = \{t_j^{p_i}\}$ necessary for a particular production target p_i. The transitions T^p and T^{p_i} are usually an input of the model, while duration and controllability come from the worker modeling and selection (see below). Then, we denote by $SV^{w_k} = (V^{w_k}, T^{w_k}, D^{w_k}, \gamma^{w_k})$ as the behavior state variables for the kth worker (often one robot and one human). Transitions T^{w_k}, duration D^{w_k}, and controllability γ^{w_k} are usually input of the model. The duration is an interval $D^{w_k}(t_m^{p_i,w_k}) = [d_m^{w_k} - \delta_m^{w_k},\ d_m^{w_k} + \delta_m^{w_k}]$ that describes the duration uncertainty when worker w_k performs $t_m^{p_i,w_k} \in V^{w_k}$. Controllability $\gamma^{w_k}(t_m^{p_i,w_k})$ depends on the nature of the worker. Task $t_m^{p_i,w_k}$ is uncontrollable if performed by humans since they may refuse to do a task or quit halfway through. Conversely, it is partially controllable if performed by a robot, as they may be unable to finish a task because humans obstruct all possible way outs.

As mentioned above, much information about the complete model is intertwined. For example, the execution time of the process task $t_j^{p_i}$ results

$$D^{p_i}\left(t_j^{p_i}\right) = \left[\min_{t_m^{p_i,w_k}}\left(d_j^{w_k} - \delta_m^{w_k}\right), \max_{t_m^{p_i,w_k}}\left(d_m^{w_k} + \delta_m^{w_k}\right)\right]$$
$$\text{s.t.} \quad t_m^{p_i,w_k} \text{ implements } t_j^{p_i}$$

otherwise, the problem is unfeasible. Similarly, the map of processes tasks $t_j^{p_i}$ into several feasible workers tasks of $t_i^{p_i,w_k}$ according to the different constraints (precedence constraints, resource constraints, etc.) is far to be simple. Proper SW tools autonomously compute such information (see Section V) that complete the model from user input, using proper synchronization rules R.

Given a set of state variables SV^P, SV^{p_i}, SV^{w_k} and synchronization rules R, we can now introduce the timeline-based planning formalization [28], [36]. We consider the generic state variable $SV^g = (V^g, T^g, D^g, \gamma^g)$ and denote by $x_j^g = (v_j^g, [e_j^{g-}, e_j^{g+}], [d_j^{g-}, d^g{}_j{}^+], \gamma(v_j^g))$ a "token" with $v_j^g \in V^g$, and $D^g(v_j^g) = [d_j^{g-}, d^g{}_j{}^+]$. We define a flexible timeline FTL_{SV^i} of a state variable SV^i as a sequence of tokens $\{x_j^g\}$ that span the process execution time and describes what the worker does over the process. Finally, a timeline-based plan π consists of a set of flexible timelines, one for each state variable, valid with respect to R. A timeline-based solver exploits different search techniques to find the optimal π among the feasible ones.

[3]The set V^p could be {*assembly A, check the quality of B, disassemble C*}, and V^{p_1} = {*Take the bottom of part A, pick and places the screws of A, Tight the screws of A*}.

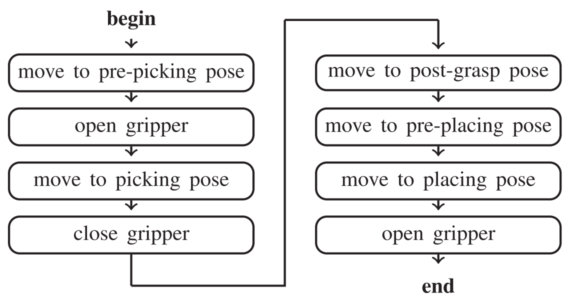

Fig. 2. Example: A pick-and-place action.

D. Action Planning Model for Human–Robot Cooperation

The action planner finds the best ACTION to execute a given TASK $t_j^{w_k}$. To do so, it gathers all the necessary geometric information from the scene descriptor (e.g., through queries to a database). Then, it determines all the actions that can realize task $t_j^{w_k}$ (i.e., all actions $a_i \in \mathcal{A}$ such that $t_j^{w_k} = S(a_i)$, according to Definition 5). For example, consider $t_j^{w_k}$ equal to "pick a blue object and place it in an empty box." The action planner uses the tags "blue object" and "empty box" to identify all the locations of the scene to which these labels are assigned. Then, the action planner finds the best sequence of ROBOTMOVEMENTS that connects those locations in the order specified in Fig. 2.

We formulate the action planning problem as the identification of the best path on a directed graph $\mathcal{G} = (U_g, E_g)$, where U_g is the set of vertices and E_g is the set of edges, and:

1) $U_{g,i}$ is a set of ROBOTCONFIGURATIONS, that corresponds to the goal of the ith ROBOTMOVEMENT;
2) $E_{g,j}$ connects two sets of ROBOTCONFIGURATIONS, that is, each edge is a set of ROBOTMOVEMENTS.

Indeed, $U_{g,i}$ includes all the ROBOTCONFIGURATIONS that are logically equivalent for the task (e.g., each one is the location of equivalent "blue objects"). Furthermore, if the representation of the location is in the Cartesian space, many joint configurations may correspond to each Cartesian pose.

For a generic action composed of n movements, the set of vertices is therefore given by

$$U_g = \bigcup_{i=0}^{n} U_{g,i}$$

where the starting vertex $U_{g,0}$ is a single ROBOTCONFIGURATION since the starting configuration is usually known.

The action planner shall find the shortest path on $\mathcal{G}$ from a vertex in $U_{g,0}$ to a vertex in $U_{g,n}$. The advantage of this formulation is that it maps a symbolic task into all its possible realizations. Recalling the example of the blue cubes, the best action is optimal for all possible inverse kinematic solutions of all grasping points of all available blue cubes.

III. OPTIMIZATION AND SOLVERS

This section proposes algorithms to solve task and action planning problems despite the high computational complexity of these problems in real-world scenarios. Without loss of generality, we will refer to the case where the workers are one robot and one human.

A. Optimization of Collaborative Processes

Consider a set of interaction windows $\mathcal{Y} = \{y_j\}$ during which the human and the robot perform some task $t_i^{p_k}$ associated with a production target p_k. Let $b_{i,j}^H$ and $b_{i,j}^R$ be binary control variables such that

$$b_{i,j}^R = \begin{cases} 1, & \text{if the robot does } t_i^{p_k} \text{ during } y_j \\ 0, & \text{otherwise} \end{cases}$$
$$b_{i,j}^H = \begin{cases} 1, & \text{if the human does } t_i^{p_k} \text{ during } y_j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The robot and the operator can perform only one task during a particular interaction window, namely

$$\forall\, j \text{ s.t. } y_j \in \mathcal{Y}, \quad \sum_i b_{i,j}^R = 1, \quad \text{and} \quad \sum_i b_{i,j}^H = 1. \tag{2}$$

Each task can be assigned only once during a process and thus executed only by the human or the robot, that is

$$\forall\, i \text{ s.t. } t_i^{p_k} \in \mathcal{T}, \quad \sum_j b_{i,j}^R + b_{i,j}^H = 1. \tag{3}$$

A duration cost function f_d can be therefore defined as

$$f_d = \sum_i \sum_j d_i^R b_{i,j}^R + d_i^H b_{i,j}^H \tag{4}$$

where d_i^H and d_i^R are the expected duration of task $t_i^{p_k} \in \mathcal{T}$ when performed by the human and by the robot separately.

However, (4) does not consider coupling effects between the robot and the human. For example, if the robot and the human move to the same area concurrently, the robot will either stop or slow down for safety reasons. To capture this synergy, we define $\Delta d_{i,j}^R$ and $\Delta d_{i,j}^H$ as

$$\Delta d_{i,j}^R = d_{i,j}^R - d_i^R \quad \text{and} \quad \Delta d_{i,j}^H = d_{i,j}^H - d_i^H \tag{5}$$

where $d_{i,j}^R$ is the expected duration of task $t_i^{p_k}$ performed by the robot while the human is performing $t_j^{p_k}$ (and vice versa for $d_{i,j}^H$). The synergy cost function f_s is therefore defined as

$$\begin{aligned} f_s &= \sum_i \sum_j \sum_k \Delta d_{i,j}^R b_{j,k}^H b_{i,k}^R + \Delta d_{i,j}^H b_{j,k}^R b_{i,k}^H \\ &= \sum_i \sum_j \left(s_{i,j} \sum_k b_{j,k}^H b_{i,k}^R \right) \end{aligned} \tag{6}$$

where $s_{i,j} = \Delta d_{i,j}^R + \Delta d_{i,j}^H$ is a synergy coefficient, that is, an index of the simultaneous tasks coupling, as shown in Table I.

In conclusion, the solution plan π is a solution to the following multiobjective optimization problem:

$$\text{minimize}_\pi \ \{f_d, f_s\} \tag{7}$$

subject to constraints (2) and (3). This article focuses on time-efficiency criteria, but the extension to other objectives is straightforward and does not undermine the search strategy.

TABLE I
SYNERGY MATRIX. EACH ELEMENT $s_{i,j}$ REPRESENTS THE INCREMENT OR DECREMENT OF DURATION GIVEN BY THE SIMULTANEOUS EXECUTION OF TASKS i AND j

R \ H	0	1	2	...	n-1	n
0	∞	$s_{0,1}$	$s_{0,2}$	...	$s_{0,n-1}$	$s_{0,n}$
1	$s_{1,0}$	∞	$s_{1,2}$	...	$s_{1,n-1}$	$s_{1,n}$
...	...	...	...	...	...	...
n-1	$s_{n-1,0}$	$s_{n-1,1}$	$s_{n-1,2}$	...	∞	$s_{n-1,n}$
n	$s_{n,0}$	$s_{n,1}$	$s_{n,2}$	...	$s_{n,n-1}$	∞

Algorithm 1 Timeline-Based Plan Synthesis

```
Input: $SV$, $R$, $S^{RH}$
Output: $\pi = (FTL, R)$
1: $\Pi \leftarrow \emptyset$
2: $\pi \leftarrow initialize(SV, R)$
3: while $\neg\, isSolution(\pi, SV, R)$ do
4:     $\Phi \leftarrow flaws(\pi, SV, R)$
5:     $\Phi^* \leftarrow chooseFlaws\ (\Phi, R)$
6:     for $\phi \in \Phi^*$ do
7:         $\Pi \leftarrow refine\ (\pi, \phi.\text{resolvers})$
8:     $\pi \leftarrow choosePlan\ (\Pi, S^{RH})$
9: return $\pi$
```

B. Task Planning as Multiobjective Search

The synthesis of a plan π uses a domain-independent refinement search, briefly described in Algorithm 1. Timelines are refined iteratively to solve inconsistencies. We detect flaws in the current partial plan at each iteration, select which flaw to solve, and refine the plan by applying possible solutions. Each solution determines an alternative refinement and, thus, an alternative partial plan. Unexplored partial plans compose the fringe of the search space and are collected into a dedicated data structure in case of backtracking. A solution is found when the partial plan extracted from the fringe does not contain flaws.

Three points are crucial in Algorithm 1: 1) the selection of the flaw to solve for plan refinement (line 5); 2) the selection of the next partial plan (line 8); and 3) the computation of the objective functions.

1) Flaw Selection and Refinement: We refer to "flaws" as conditions that affect the completeness or validity of timelines. Flaws may concern tokens to be added to timelines (planning flaws) or tokens of a timeline to be ordered because they overlap (scheduling flaws). This choice determines the way the solving process interleaves planning and scheduling decisions. We implement a hierarchy-based heuristic that considers the synchronization rules of a domain specification [37].[4]

2) Refinement of Partial Plans: Given the multiobjective nature of the problem, we pursue a Pareto optimality approach [38] and apply the *dominance relationship* to partial plans.

[4]Intuitively, a higher hierarchical level is assigned to SVs appearing in the rules' trigger (i.e., SVs influencing behaviors of other SVs). A lower hierarchical level is assigned to SVs appearing in the rules' body (i.e., SVs whose behavior depends on other SVs).

Definition 6: Given a set $\{f_1, \ldots, f_n\}$ of cost functions, a partial plan π_i *dominates* a partial plan π_j (with $i \neq j$) if

$$f_k(\pi_i) < f_k(\pi_j) \quad \forall\ k = 1, \ldots, n.$$

The dominance condition is used to compare partial plans to heterogeneous objective functions and identify the Pareto set of the search space. The Pareto set comprises partial plans and represents a suitable tradeoff between objective functions f_k. Such partial plans are compared on a priority assigned to the objectives. This work's objectives are makespan, f_d, and synergy, f_s. Among the dominant plans, we prioritize synergy.

3) Cost Estimation of Partial Plans: The implementation of (7) according to the timeline framework needs some intermediate steps. The objective function must be composed of a "cost term" and a "heuristic term." Specifically, the cost term models the scheduled tokens of the timelines of a plan $FTL_i \in \pi$. Instead, the heuristic term considers possible *projections* of the timelines. A projection ξ_j^i represents a particular sequence of tokens $x_k \in \xi_j^i$ that may complete the timeline FTL_i in future refinements of a plan π. All the possible projections of the timeline FTL_i define a set $\Xi_i \ni \xi_j^i$ for all j, and the minimization passes through the computation of partial plans whose timelines are not necessarily complete.

Consider (4), the objective function $f_d(\pi)$ turns in

$$f_d(\pi) = \max_{FTL_i \in \pi} \left(\sum_{x_j \in FTL_i} d_j + \max_{\xi_j^i \in \Xi_i} \sum_{x_k \in \xi_j^i} d_k \right). \tag{8}$$

Given FTL_i, the makespan turns into the sum of the duration d_j of its tokens $x_j \in FTL_i$ with the maximum sum of the duration d_k of tokens x_k belonging to the projections $\xi_j^i \in \Xi_i$.

Consider (6), the objective function $f_s(\pi)$ turns in

$$f_s(\pi) = \sum_{x_i^R \in FTL_R} \sum_{x_j^H \in \Omega(x_i^R)} s_{i,j} + \max_{\xi_j^R \in \Xi_R} \sum_{x_k^R \in \xi_j^R} \max_{\substack{x_m^H \in V^H \\ m \neq k}} \left(s_{k,m} \right) \tag{9}$$

where FTL_R is the robot timeline, $\Omega(x_i^R) = \{x_1^H, \ldots, x_n^H\}$ the set of tokens of the human timeline FTL_H. The execution of x_j^H may overlap in time with x_i^R, and the synergy term $s_{i,j}$ is computed for each pair of overlapping tokens, x_i^R and x_j^H and extracted from the matrix S^{RH} (Table I). The first term of (9) is the cost term, while the second is the heuristic term. This last term considers possible projections of the robot timeline $\xi_j^R \in \Xi_R$ as the maximum expected synergy of the plan according to the worst synergy that corresponds to the maximum value of $s_{k,m}$ of the tokens $x_k^R \in \xi_j^R$ and $x_m^H \in V^H$ with $m \neq k$, that is, the maximum value of all the possible combinations.

C. Action Planning as Multigoal Motion Planning Problem

According to Section II-D, the optimal action planning problem consists of finding the shortest path from the current configuration $q_0 \in U_{g,0}$ to a vertex in $U_{g,n}$ on a graph $\mathcal{G} = (U_g, E_g)$. Each edge in E_g corresponds to a motion planning problem between the configurations associated with the edge vertices.

Algorithm 2 Refining Tasks Into Motion Plans

```
Input: action_model, robot_tree
Output: motion_plans
 1: $q_{\text{current}} \leftarrow$ getCurrentConfiguration()
 2: $(U_{g,1}, \ldots, U_{g,n}) \leftarrow$ getGoalsFromScene(action_model, robot_tree)
 3: for $U_{g,i}$ in $(U_{g,1}, \ldots, U_{g,n})$ do
 4:     $(\sigma, c) \leftarrow$ motionPlanning(robot_tree, $q_{\text{current}}$, $U_{g,i}$)
 5:     if isEmpty($\sigma$) then
 6:         break
 7:     else
 8:         motion_plans.append($\sigma$)
 9:         cost $\leftarrow$ cost+$c$
10:         $q_{\text{current}} \leftarrow \sigma(1)$
11: return motion_plans
```

Solving the shortest path on $\mathcal{G}$ is often not viable because evaluating all the edge weights is time consuming (solving a single motion planning problem may take seconds in a realistic scenario). To achieve online implementation, we pursue an approximated approach. We decompose the action planning problem into a sequence of subproblems, that is, one subproblem for each ROBOTMOVEMENT of the ACTION. Then, we optimize the sequence of ROBOTMOVEMENTS step by step. The procedure is described in Algorithm 2.

Procedure *getGoalsFromScene* gets the ROBOTCONFIGURATIONS, $U_{g,1}, \ldots, U_{g,n}$ (line 2), where $U_{g,i}$ is the set of configurations goals associated to the *i*th movement. Then, the algorithm solve a motion planning problem from q_{current} to $U_{g,i}$ (line 4). The resulting motion plan is a curve $\sigma : [0, 1] \rightarrow \mathcal{C}$ such that $\sigma(0) = q_{\text{current}}$ and $\sigma(1) \in U_{g,i}$. The curve σ is appended to the array motion_plans (line 8), and the total cost is updated (line 9). Finally, the current configuration is updated with the final configuration of the chosen trajectory (line 10), and the procedure is repeated.

Function *getGoalsFromScene* queries the database containing all locations in the environment. If the locations are expressed as Cartesian poses, the function converts them into ROBOTCONFIGURATIONS by applying inverse kinematics. Simplifying, the function gets all the configuration goals associated with each movement of an ACTION.

This approximated approach turns a subaction into a multigoal motion planning problem (line 4), for which efficient solvers exist. Procedure *motionPlanning* outputs the minimum-cost path from a starting configuration to any configuration in $V_{g,i}$. This problem may be solved by running the motion planner for each configuration in $V_{g,i}$ and by selecting the best solution, although this approach is inefficient and does not scale well to the number of goals. Informed sampling-based planners [39] solve the problem efficiently in a single query.

IV. QUALITATIVE ASSESSMENT

Existing TAMP approaches for HRC fit different requirements and assumptions, leading to the lack of a shared standard. A quantitative, fair comparison between existing works is difficult because each method is designed to comply with different constraints. In this section, we resort to examples of real-world problems to argue that our methodology can address a wider variety of cases than existing methods.

TABLE II
COMPARISON BETWEEN THE PROPOSED TAMP APPROACH AND OTHER WORKS IN THE LITERATURE

	Logic constraints	Geometric constraints	Temporal constraints	Hierarchical	Task optimization	Optimal robot traj.	Optimal robot actions	Contingency strategies	Robust w.r.t. duration	Online geom. reasoning
[3], [4], [5], [6]	✓	✓		✓						
[9], [8]	✓	✓		✓	✓	✓^{1}	✓^{1}			
[10]	✓	✓	✓	✓						
[19], [20]	✓	✓		✓						
[21], [22]	✓	✓		✓				✓		
[24], [25]	✓	✓		✓	✓			✓		
[26], [27]	✓	✓		✓				✓		
[32]	✓	✓	✓	✓	✓	✓^{1}		✓	✓	
Our approach	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

A. Case Studies

We consider a typical end-of-line packaging application, where two collaborative lightweight robots are installed, and a human operator can access the cell [40]. The robots must pick the packs from a ball-transfer table and place them into boxes. Storing packs in the boxes must follow certain rules (e.g., a mosaic composition). If needed, the human inspects the packing quality and handles the packs. The packs on the ball-transfer table are randomly positioned. An external camera and an eye-in-hand camera give a raw and refined localization. In this scenario, we analyze the effectiveness of the most relevant works mentioned in Section I-A. The results are summarized in Table II.

1) Temporally Bounded Process Execution: Consider the noncollaborative case (that is, no human intervention) in which the robot cannot overcome a maximum execution time owing to plant constraints. The methods in Table II that do not model *temporal constraints* cannot guarantee the plan constraints. This could not be granted even if they integrate *optimal robot trajectories* and *optimal robot actions* because action optimality does not imply the plan is minimum time.

2) Execution in Dynamic Environments: Consider the noncollaborative case in which the position of the remaining packs can change after each grasping operation. The methods in Table II that do not implement any *contingency strategies* or *online geometric reasoning* need replanning when an unforeseen event occurs. Suppose an action takes longer than expected (e.g., the camera takes longer to identify the grasping position). In that case, *temporal constraints* as in [10] may be violated unless the plan is robust to action duration. Conversely, the algorithms with *contingency strategies* or *online geometric reasoning* need a local replanning.

3) Execution With Multiple Agents: In the case that two robots are used, a delay in the execution of a task (e.g., due to slow processing of a sensor, occlusions of a camera, etc.) may introduce synchronization issues between the agents.

All the methods in Table II that integrate neither *temporal constraints* nor *contingency strategies* can be highly inefficient in the execution. On the one hand, methods that integrate only *contingency strategies* could cope with the coordination of multiple agents, but a delay in the task execution may violate precedence constraints. On the other hand, methods that integrate only *temporal constraints* [10] may fail in the execution when a worker overcomes the time limits due to unmodeled events (e.g., occlusions, the hold of the movement for safety reasons, etc.). Increasing temporal constraints to compensate for this behavior is not helpful since this information is used only in the planning phase and not during the execution. This issue can be mitigated if the temporal modeling used in the plan computation is robust to the task execution latency.

4) Optimal Execution With Multiple Agents: Consider that throughput must be maximized to guarantee the economic return on the robotic cell. The methodologies in [8] and [9] compute an optimal initial plan. However, synchronization issues would arise when a task takes longer than expected, as discussed in Section IV-A3. Consequently, a delay in grasping (e.g., due to slow camera perception) leads to an optimality loss or the need to recompute the whole plan.

5) Optimal Execution in Dynamic Environments: Consider throughput as the goal, and consider that the packs' position can change after each grasping. Toussaint [8] and Zhang and Shah [9] cannot grant the plan's feasibility because the trajectory must be recomputed online without excessive idle times. Although [32] exploits the timelines, it fails because the optimal plan is computed offline, based on a probabilistic model. The plan computation should be continuously updated to overcome this limitation in parallel with the task execution. However, the latency of the continuous update is high due to the high computational load of the methodology. Darvish et al. [24], [25] grant local optimality, but movable objects in the scene do not allow for global optimization. Finally, [26], [27] embody a contingency strategy to overcome failures related to misalignment between models and reality. These methods do not generalize to temporal constraints and do not allow for trajectory and action optimization.

B. Discussion

According to the analysis above, our TAMP approach is the most adequate to deal with typical real-world requirements. The capability of considering execution issues at both task and action/motion planning levels is a crucial advantage. This capability enables reliable coordination of agents while preserving the optimality of task assignment, action implementation, and the resulting collaboration.

The combination of temporal flexibility, optimal task allocation, and optimal online motion trajectories allows us to preserve coordination and production efficiency through reliable task plans and behaviors. Specifically, timeline-based planning effectively integrates reasoning on allocating tasks to the agents and optimizing production while considering possible deviations at execution time. Furthermore, the action planner evaluates the state of the environment online and computes optimal trajectories of motions on the fly.

V. Experimental Assessment

A. Case Study

We consider a case study derived from the EU-funded project *ShareWork* (http://www.sharework-project.eu) where a robot arm (Universal Robots UR5 on an actuated linear track) and a human operator have to assemble a mosaic (see Fig. 3). We consider four mosaics composed of 4, 9, 16, and 50 cubes of different colors (blue, orange, and white). Each slot has a label given by its column letter and its row number (that is, *A1, A2, ...*). A common condition in HRC is that some operations can be performed only by the robot or the human. In this example, the following allocation constraints are imposed: orange cubes shall be moved only by the robot; white cubes shall be moved only by the human; both can move blue cubes.

1) Process Planning Model: The case study is traced back to the timeline-based formalism described in Section II-C. The process is modeled as a production state variable $SV^P = (V^P, T^P, D^P, \gamma^P)$, where each value of V^P represents the assembly of a row, that is, $V^P = \{\text{DoRow}_1, \text{DoRow}_2, \ldots\}$. The precedence of some rows over others may be set at this level as synchronization rules. Two behavior state variables, $SV^H = (V^H, T^H, D^H, \gamma^H)$ and $SV^R = (V^R, T^R, D^R, \gamma^R)$, model the low-level tasks that the human and the robot can perform. In this case study, the human and the robot perform tasks of the type PickPlace_x, which consists of picking a cube and placing it in the slot with label x. Hence, $V^H = \{\text{PickPlace}_y\}$ and $V^R = \{\text{PickPlace}_z\}$, where y is the labels corresponding to white and blue cubes, and z are labels corresponding to orange and blue cubes. Note that $V^R \cap V^H \neq \emptyset$ because blue tiles can be assigned to humans and robots.

According to the controllability notion given in Section II-C, the human behavior is modeled as uncontrollable ($\gamma^H(v_k) = u \; \forall v_k \in V^H$) and the robot behavior of is partially controllable ($\gamma^R(v_k) = pc \; \forall v_k \in V^R$). The duration of each task (D^R and D^H) is estimated as in [41]. This duration estimation method considers the human interference on the robot paths and estimates the duration for all possible obstruction and safety stop cases. Based on this estimation, each element $s_{i,j}$ of the synergy matrix is computed as in (6).

Each task $\text{PickPlace}_z \in V^R$ corresponds to a set of actions to be performed by the robot. Each action boils down to a sequence of RobotMovements, as shown in Fig. 2. When the action planner receives a task, the action planner decodes the cube's color and the goal slot from a database. Then, it solves the action planning problem described in Algorithm 2.

2) Software Implementation: The task planner in Section III has been integrated into the timeline-based planner PLATINUm [42], using a *hierarchy-based heuristics* for flaw selection and an *HR-balancing search strategy* for search expansion [43]. The rest of the framework is implemented in ROS and connected to PLATINUm by *rosbridge_suite* [44]. PLATINUm dispatches tasks to the robot action planner and waits for feedback. In real-world tests, an HMI system would communicate the task to humans

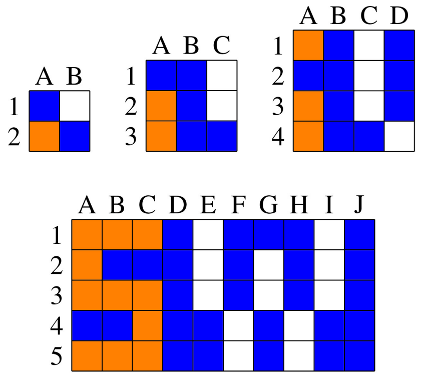
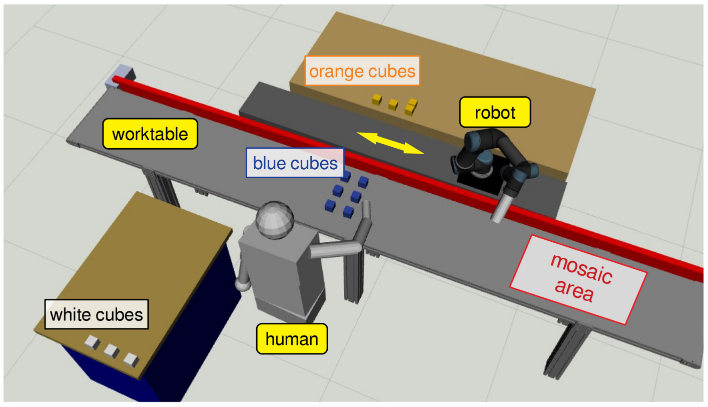

Fig. 3. Case study: A 7-degree-of-freedom robot and a human worker collaborate to assemble a mosaic. Four mosaics are considered, with the number of cubes ranging from 4 to 50.

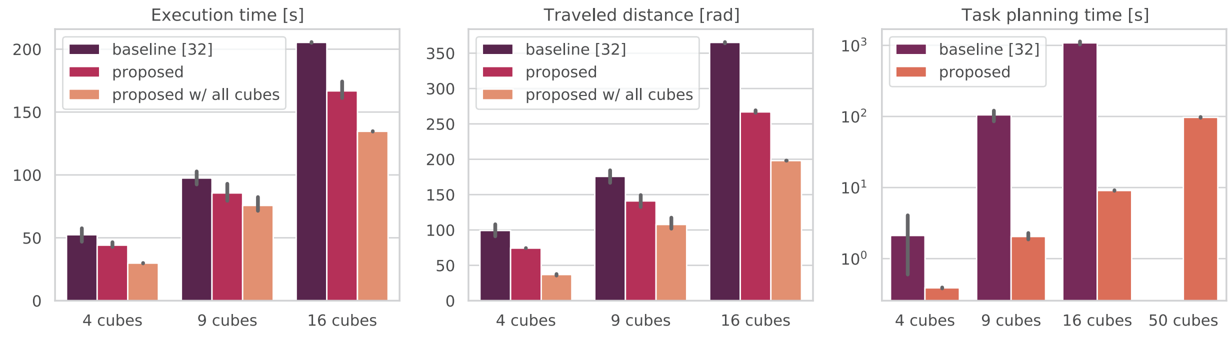

Fig. 4. Results of Experiment 1a. The proposed method (*proposed*) is compared with [32] (*baseline*). The *proposed* method reduces the execution time (left plot), the robot's traveled distance (middle plot), and the planning time of the task planner (right plot). The difference is even more evident when more cubes than strictly necessary are used (*proposed with all cubes*).

and receive feedback from them. In simulated tests, the human is modeled through a mannequin[5] commanded by a second instance of the action planner used for the robot.[6]

The action planner is implemented by using the library [45], which builds high-level skills on top of *MoveIt!*. Because of the multigoal nature of the proposed action planning and the online requirement, we use MI-RRT* [46], a fast variant of Informed-RRT* [39], to solve the motion planning problems.

B. Experiments

We discuss three experiments to evaluate the different components of the proposed approach. The first two analyze the reasoning capabilities of the task planner and action planner alone. The third one then evaluates their integration within the proposed TAMP approach. We conclude the section with a final discussion of the results emphasizing the main advantages and strengths of the approach. A video of the experiments is attached to this manuscript.

1) Experiment 1 (Task Planning Performance): We compare our approach with a timeline-based approach [32] and an action-based approach [10] (according to Table II, [32] and [10] are the only approaches that can manage temporal constraints).

a) Comparison with timeline-based approaches: Reference [32] has two main limitations. First, the task planner reasons at a low level of abstraction (i.e., each point-to-point movement is modeled as a TASK), putting the task planner in charge of finding the optimal ordering and assignment of all the ROBOTMOVEMENTS. Second, it is based on the a-priori modeling of the trajectories, that is, all trajectories are precomputed, and the task planner reasons on the estimated costs of such plans.

To demonstrate that our approach scales better than [32] to complex processes, we consider the mosaics in Fig. 3. Pellegrinelli et al. [32] modeled each ROBOTMOVEMENT from a cube to each slot (and vice versa) as a TASK. On the contrary, the proposed method only requires one TASK for each slot. As the planning time roughly grows exponentially in the number of TASKS, the planning time of the proposed method is around one order of magnitude smaller than that of [32], as shown in Fig. 4 (right plot). Note that [32] could not solve the 50-cube mosaic within the maximum planning time of 15 min.

Similar reasoning holds for the motion planning phase. Pellegrinelli et al. [32] precomputes all trajectories from all cubes to all slots and vice versa. Suppose the number of cubes in the scene is equal to the number of slots of the mosaic, and

[5]The modeled movements are: trunk (2 translations and 3 rotations), shoulders (3 rotations each), elbow (1 rotation each), and wrist (3 rotations).

[6]Using a simulation model of the environment and the human worker is fundamental to ensure high repeatability of the tests, erasing the effects of measurement uncertainty, differences between human subjects, and human–machine communication. This allows for a fair comparison between different methods, focusing only on the effects of TAMP.

TABLE III
RESULTS OF EXPERIMENT 1.2. COMPARISON OF THE PROPOSED METHOD AND [10]

		Proposed	Edelkamp et al. [10]
Planning time [ms]	4 cubes	**392**(6.1)	263(34)
	9 cubes	**1991**(240)	947(91)
	16 cubes	**8942**(176)	5099(756)
Execution time [s]	4 cubes	**43**(5.7)	65(6.1)
	9 cubes	**85**(6.3)	180(8.5)
	16 cubes	**166**(7.2)	197(6.1)

let $\tau_{\max}$ be the maximum planning time after which a motion planning query is stopped and b the number of slots. Then, the offline phase can take up to $2\tau_{\max}b^2$ s. Considering that, in our tests, $\tau_{\max} = 5$ s, this corresponds to 90, 360, 1690, and 14440 s for each mosaic. On the contrary, our approach computes the trajectories online, dealing with uncertainty and changing goals.

Concerning the execution phase, we compare the performance of the two approaches for the 4-cube, 9-cube, and 16-cube mosaics. Fig. 4 shows that the execution time of the proposed method is significantly shorter (−14%, −12%, −20% for the three mosaics) because the action planner chooses the most convenient movement online and based on the current robot and human state. Indeed, the robot's traveled distance is much shorter (−26%, −39%, −27% for the three mosaics). This difference is even more evident when more cubes than necessary are available. For example, we consider the case where 50 cubes are available although only 4, 9, and 16 are necessary (method "proposed w/ all cubes" in Fig. 4). Because our method can choose among a broad set of objects at each PickPlace$_x$ task, it leads to a further improvement of the execution time (−44%, −22%, −34% for the three mosaics) and the robot's traveled distance (−63%, −39%, −46% for the three mosaics).

b) Comparison with action-based approaches: A direct comparison with other approaches mentioned in Table II is difficult because of the intrinsic differences in planning formalism and models.

Nonetheless, we consider [10] as it supports *temporal constraints* and *hierarchical decomposition*. Thus, the comparison focuses on how the two models deal with uncertain and strict temporal requirements.

Edelkamp et al. [10] proposed an integration of TAMP capabilities based on PDDL2.1 [47]. It uses an action-based representation with so-called *durative actions* that do not consider scheduling aspects. PDDL2.1 planners do not pursue makespan optimization but use temporal constraints for plan consistency. Furthermore, neither *temporal flexibility* nor *temporal uncertainty* are considered by such planners. Therefore, the task planning models pick-and-place tasks with a fixed duration and consider the robot and worker controllable. We run our implementation of [10] for the 4-cube, 9-cube, and 16-cube mosaic and show the results in Table III. As expected, [10] achieves the best planning time for all scenarios because the action-based planner focuses on process decomposition without considering optimization aspects. Scheduling decisions do not impact the reasoning, while the lack of flexibility reduces the number of choices considered during the search. Then, we execute the plans obtained for the 4-cube, 9-cube, and 16-cube mosaic, simulating the uncertainty of the worker's actions ($\delta = \pm 5$ time units). The objective is to evaluate the reliability of synthesized plans in a realistic scenario where human workers behave uncontrollably. As shown in Table III, the execution time of [10] is greater than the execution of our approach (up to +110%). The lack of temporal flexibility with respect to uncertainty does not allow to deal with the uncontrollable dynamics of the worker effectively. Consequently, the frequent need for replanning increases the execution time of the plan, leading to less efficient (and effective) collaborations.

2) Experiment 2 (Action Planning Performance): Consider the 50-cube mosaic of Fig. 3 and the following action-planning configurations.

1) *Precomputed:* Motion plans from and to each point are computed offline, as in [32]. Before each movement, the algorithm chooses the closest goal to the current robot position. Referring to Table II, this approach displays *optimal robot trajectories*, but neither *optimal actions* nor *online geometric reasoning*. It cannot deal with dynamic environments as paths are computed *a priori*.
2) *Single-Goal:* Motion plans are computed before execution. The action planner always selects the closest goal to the current robot position. Referring to Table II, this approach has *optimal robot trajectories* and *online geometric reasoning*, but no *optimal actions*.
3) *Multigoal:* The proposed action planning. Motion plans are computed just before their execution. The multigoal optimal motion planner considers all the goals with the desired properties, yielding *optimal robot trajectories*, *online geometric reasoning*, and *optimal actions*.

We evaluate the following indexes: 1) total execution time of the robot tasks (in seconds); 2) joint-space distance traveled by the robot (in radians); and 3) planning time for the motion planning algorithm, *that is*, the sum of the planning times of all movements to perform a pick-and-place action (in seconds).

We run 20 tests for each configuration by using a task plan generated by the *feasible* configuration described in Section V-B3 (the chosen plan assigns 25 tasks to the robot and 25 tasks to the human). Results are in Fig. 5. The *multigoal* configuration outperforms the *precomputed* and the *single-goal* variants with a reduction of around 48% in the traveled distance (*precomputed*: mean = 1487 rad, stdev = 23.7 rad; *single-goal*: mean = 1480 rad, stdev = 27.1 rad; *multigoal*: mean = 768 rad, stdev = 2.21 rad). All configurations use the same optimal path planner; the improvement of the solution is due to the choice of the goal: *precomputed* and *single-goal* direct the search toward the closest goal, which is often suboptimal. Other heuristics could be adopted to select the goal, but the results would strongly depend on the geometric properties of the workspace. For example, the "closest-one" heuristic would perform even worse in a cluttered environment. On the contrary, *multigoal* always finds the best solution regardless of the geometry of the workspace. Shorter paths reduce the execution time, as shown in Fig. 5. Note that the difference in the execution time is less pronounced than the one in the traveled distance. The reason is that the execution time also

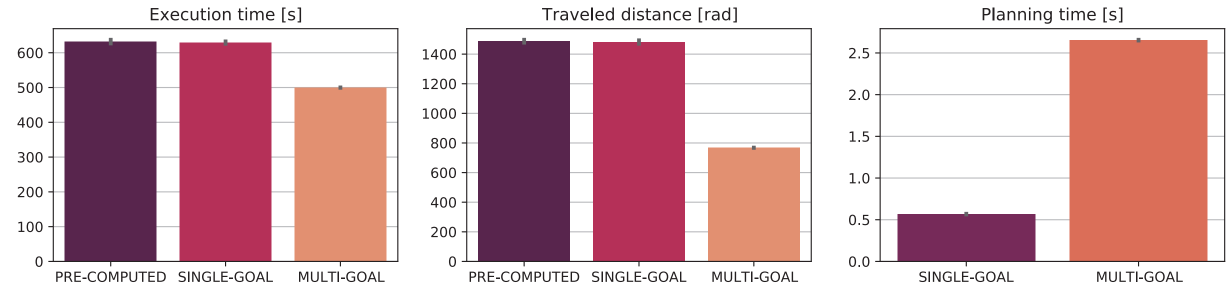

Fig. 5. Results of Experiment 2. Comparison of the proposed *multigoal* action planner and a *precomputed* motion planner and a *single-goal* planner. The path length is reduced (left plot) as the robot execution time (middle plot). Planning times are larger but suitable for online planning (right plot; **precomputed** is not shown because it relies on offline planning).

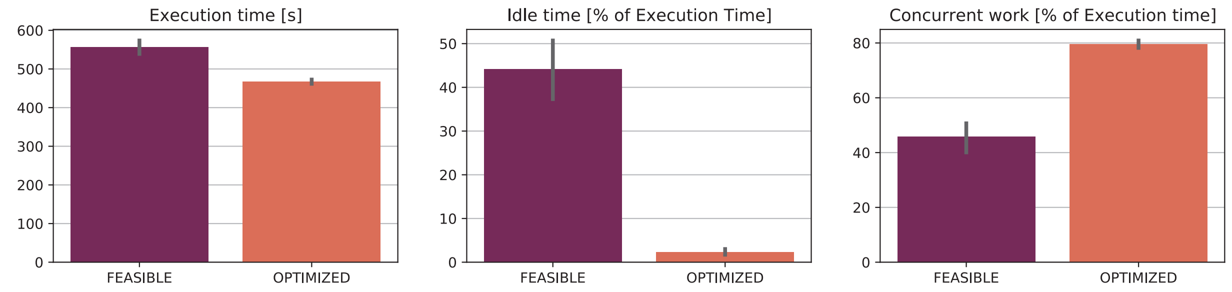

Fig. 6. Results of Experiment 3. The proposed method (*optimized*) is compared with a feasibility-oriented approach (*feasible*): *optimized* reduces the process execution time (left plot) and the idle time of human and robot (middle plot) and increases the time the robot and the human work simultaneously (right plot).

considers planning latency, safety slowdown, and communication overhead, and it is affected by the path parametrization algorithm. Nonetheless, *multigoal* leads to a significant reduction (around 21%) in the robot execution time (*precomputed*: mean = 632 s, stdev = 12.0 s; *single-goal*: mean = 629 s, stdev = 8.05 s; *multigoal*: mean = 500 s, stdev = 1.27 s). Note that the *multigoal* approach results in a minor variance of the traveled distance and execution time because the multigoal search is less affected by the robot state at the beginning of each action.

Finally, the motion planning times are shown in the right plot of Fig. 5 (*precomputed* is not shown as all paths are computed offline). As expected, *multigoal* has planning times higher than *single-goal* (*single-goal*: mean = 0.567 s, stdev = 0.057 ms; *multigoal*: mean = 2.65 s, stdev = 0.211 s). This difference is intrinsic to the multigoal nature of the planner, which has to plan toward all available goals. Nonetheless, planning times of *multigoal* are still in the order of a few seconds and, therefore, suitable for online planning. Moreover, this discrepancy is expected to decrease thanks to the constant advances in multigoal motion planning algorithms.

3) Experiment 3 (TAMP Performance): Consider the 50-cube mosaic of Fig. 3 and the following cases.

1) *Feasible:* We generate feasible random task plans with respect to allocation and precedence constraints but do not optimize duration and synergy. The number of tasks assigned to the human is chosen as a uniform random variable between 12 and 39 (i.e., the smallest and largest number of cubes the human can move). Referring to Table II, this configuration reproduces those methods that do not implement any *task optimization*, *contingency strategy*, or *temporal robustness*.
2) *Optimized:* We use the proposed multiobjective optimization approach. The task planner decides the order and allocation of the tasks.

Both configurations use the proposed action planner to highlight the differences owing to the task plan generation.

For *optimized*, the task planner decides the number of tasks for each worker in such a way as to minimize the process duration. This is a key feature of our approach, while existing works either assume that the number of assignments is given or they find a feasible assignment, disregarding its optimality. To reproduce this issue, we let the *feasible* approach randomly decide the number of tasks assigned to each worker as long as the assignment is feasible. The following indexes are evaluated.

1) Process Execution Time, $\mathrm{ET}_P = \max(\mathrm{ET}_R, \mathrm{ET}_H)$.
2) Idle Time $\mathrm{IT} = 100\,|\mathrm{ET}_R - \mathrm{ET}_H|/\mathrm{ET}_P$ [%].
3) Concurrent working Time of human and robot, $\mathrm{CT} = 100 \cdot (\min(\mathrm{ET}_R, \mathrm{ET}_H) - \mathrm{ST})/\mathrm{ET}_P$ [%].

where ET_R and ET_H are the execution time of the robot and the human, and ST is the robot holding time because of safety (i.e., when the human is close to the robot). The first index measures the throughput of the process, and the second and the third measure the quality of collaboration.

We run 20 tests for each configuration; results are in Fig. 6. The *optimized* approach outperforms the *feasible* one by reducing ET_P of around 16% (*optimized*: mean = 467 s, stdev = 11.9 s; *feasible*: mean = 557 s, stdev = 39.3 s) and IT of around 95% (*optimized*: mean = 2.36%, stdev = 1.50%; *feasible*: mean = 44.2%, stdev = 15.2%), while increasing CT of around 74% (*optimized*: mean = 79.6%, stdev = 3.26%; *feasible*: mean = 45.8%, stdev = 12.1%). Note that the *optimized* approach displays a balanced assignment of tasks to the robot and the human. In this case, the task planner assigns 27 tasks to the human and 23 tasks to the robot, so the two expected makespans are similar. As a result, the execution time and the idle time are shorter.

C. Discussion

The outcomes of these experiments confirm the conclusions of the qualitative assessment. The task planning assessment shows that the timeline-based approach achieves a higher level of reliability during plan execution in the case of uncontrollable delays or temporal deviations in the execution of tasks. It also shows that the proposed approach scales to more complex tasks than previous timeline-based methods (Fig. 4).

The action planning assessment shows that the dynamic selection of the motion goal significantly increases the flexibility and reliability of robot motions and achieves shorter execution time and robot traveled distance (Fig. 5).

The third experiment focuses on integration. It shows the advantages of integrating the two planning approaches. The results of Fig. 6 clearly show that the combination of optimal reasoning at the two levels of abstraction significantly improves the synergetic behaviors of the agents in terms of both idle time and concurrency.

VI. Conclusions and Future Works

This article proposed a TAMP approach for hybrid collaborative processes. The proposed method follows a multiobjective optimization approach to maximize the throughput of the process. We demonstrated the advantages of the method compared with state-of-the-art techniques, both from a qualitative and numerical point of view. Future works will focus on integrating learning techniques to refine the process model through experience and speed up the search for optimal plans [48]. For example, [49] presents a preliminary study on learning task duration and human–robot synergy via linear regression. Further investigation will also address the use of other optimization objectives (e.g., taking into account human factors and user preferences [50] or agent–agent communication [51]). Finally, real-world tests of human workers will assess the system's performance and dependability.

References

[1] U. E. Ogenyi, J. Liu, C. Yang, Z. Ju, and H. Liu, "Physical human–robot collaboration: Robotic systems, learning methods, collaborative strategies, sensors, and actuators," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 1888–1901, Apr. 2021.

[2] C. R. Garrett et al., "Integrated task and motion planning," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 4, pp. 265–293, May 2021.

[3] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 639–646.

[4] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *Proc. IEEE/RSJ Int. Conf. Int. Robots Syst.*, 2014, pp. 3684–3691.

[5] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *Int. J. Robot. Res.*, vol. 37, no. 10, pp. 1134–1151, 2018.

[6] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "FFRob: Leveraging symbolic planning for efficient task and motion planning," *Int. J. Robot. Res.*, vol. 37, no. 1, pp. 104–136, 2018.

[7] F. Rovida et al., "SkiROS—A skill-based robot control platform on top of ROS," in *Robot Operating System (ROS)* (Studies in Computational Intelligence), vol. 707, A. Koubaa, Ed. Cham, Switzerland: Springer, 2017, pp. 121–160.

[8] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 1–7.

[9] C. Zhang and J. A. Shah, "Co-optimizing task and motion planning," in *Proc. IEEE/RSJ Int. Conf. Int. Robots Syst.*, 2016, pp. 4750–4756.

[10] S. Edelkamp, M. Lahijanian, D. Magazzeni, and E. Plaku, "Integrating temporal reasoning and sampling-based motion planning for multigoal problems with dynamics and time windows," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3473–3480, Oct. 2018.

[11] A. Casalino, A. M. Zanchettin, L. Piroddi, and P. Rocco, "Optimal scheduling of human–robot collaborative assembly operations with time Petri nets," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 1, pp. 70–84, Jan. 2021.

[12] A. Ham and M.-J. Park, "Human–robot task allocation and scheduling: Boeing 777 case study," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1256–1263, Apr. 2021.

[13] M. Faccio, I. Granata, and R. Minto, "Task allocation model for human–robot collaboration with variable cobot speed," *J. Intell. Manuf.*, vol. 2023, pp. 1–14, Jan. 2023.

[14] Y. Cheng, L. Sun, and M. Tomizuka, "Human-aware robot task planning based on a hierarchical task model," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1136–1143, Apr. 2021.

[15] W. Wang, R. Li, Z. M. Diekel, and Y. Jia, "Robot action planning by online optimization in human–robot collaborative tasks," *Int. J. Intell. Robot. Appl.*, vol. 2, no. 2, pp. 161–179, 2018.

[16] K. Li, Q. Liu, W. Xu, J. Liu, Z. Zhou, and H. Feng, "Sequence planning considering human fatigue for human–robot collaboration in disassembly," *Procedia CIRP*, vol. 83, pp. 95–104, Jul. 2019.

[17] J. Chen, Y. Ding, B. Xin, Q. Yang, and H. Fang, "A unifying framework for human–agent collaborative systems—Part I: Element and relation analysis," *IEEE Trans. Cybern.*, vol. 52, no. 1, pp. 138–151, Jan. 2022.

[18] Y. Ding, B. Xin, J. Chen, Q. Yang, and H. Fang, "A unifying framework for human–agent collaborative systems—Part II: Design procedure and application," *IEEE Trans. Cybern.*, vol. 52, no. 11, pp. 11990–12002, Nov. 2022.

[19] S. Lemaignan, M. Warnier, E. A. Sisbot, A. Clodic, and R. Alami, "Artificial cognition for social human–robot interaction: An implementation," *Artif. Intell.*, vol. 247, pp. 45–69, Jun. 2017.

[20] L. De Silva, R. Lallement, and R. Alami, "The HATP hierarchical planner: Formalisation and an initial study of its usability and practicality," in *Proc. IEEE/RSJ Int. Conf. Int. Robots Syst.*, 2015, pp. 6465–6472.

[21] G. Milliez, R. Lallement, M. Fiore, and R. Alami, "Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring," in *Proc. ACM/IEEE Int. Conf. Human.-Robot Inter.*, 2016, pp. 43–50.

[22] E. Sebastiani, R. Lallement, R. Alami, and L. Iocchi, "Dealing with on-line human-robot negotiations in hierarchical agent-based task planner," in *Proc. Int. Conf. Autom. Plan. Schedul.*, 2017, pp. 549–557.

[23] G. Evangelou, N. Dimitropoulos, G. Michalos, and S. Makris, "An approach for task and action planning in human–robot collaborative cells using AI," *Procedia CIRP*, vol. 97, pp. 476–481, Feb. 2021.

[24] K. Darvish, B. Bruno, E. Simetti, F. Mastrogiovanni, and G. Casalino, "Interleaved online task planning, simulation, task allocation and motion control for flexible human–robot cooperation," in *Proc. IEEE Int. Symp. Robot Human Interact. Commun.*, 2018, pp. 58–65.

[25] K. Darvish, E. Simetti, F. Mastrogiovanni, and G. Casalino, "A hierarchical architecture for human–robot cooperation processes," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 567–586, Apr. 2021.

[26] A. Akbari, Muhayyuddin, and J. Rosell, "Knowledge-oriented task and motion planning for multiple mobile robots," *J. Exp. Theor. Artif. Intell.*, vol. 31, no. 1, pp. 137–162, 2019.

[27] A. Akbari, M. Diab, and J. Rosell, "Contingent task and motion planning under uncertainty for human–robot interactions," *Appl. Sci.*, vol. 10, no. 5, p. 1665, 2020.

[28] N. Muscettola, "HSTS: Integrating planning and scheduling," in *Intelligent Scheduling*, M. Zweben and M. S. Fox, Ed. Burlington, MA, USA: Morgan Kauffmann, 1994.

[29] A. Ceballos et al., "A goal-oriented autonomous controller for space exploration," in *Proc. ASTRA*, 2011, pp. 1–8.

[30] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi, "MrSPOCK: Steps in developing an end-to-end space application," *Comput. Intell.*, vol. 27, no. 1, pp. 83–102, 2011.

[31] P. H. Morris, N. Muscettola, and T. Vidal, "Dynamic control of plans with temporal uncertainty," in *Proc. Int. Joint Conf. Artif. Intell.*, 2001, pp. 494–499.

[32] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, and T. Tolio, "Motion planning and scheduling for human and industrial-robot collaboration," *CIRP Ann.*, vol. 66, no. 1, pp. 1–4, 2017.

[33] E. Gat, "On three-layer architectures," in *Artificial Intelligence and Mobile Robots*. Cambridge, MA, USA: MIT Press, 1998, pp. 195–210.

[34] J. A. Marvel, J. Falco, and I. Marstio, "Characterizing task-based human–robot collaboration safety in manufacturing," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 260–275, Feb. 2015.

[35] J. T. C. Tan, F. Duan, Y. Zhang, and T. Arai, "Task decomposition of cell production assembly operation for man-machine collaboration by HTA," in *Proc. IEEE Int. Conf. Autom. Logist.*, 2008, pp. 1066–1071.

[36] M. Cialdea Mayer, A. Orlandini, and A. Umbrico, "Planning and execution with flexible timelines: A formal account," *Acta Informatica*, vol. 53, nos. 6–8, pp. 649–680, 2016.

[37] A. Umbrico, A. Orlandini, and M. C. Mayer, "Enriching a temporal planner with resources and a hierarchy-based heuristic," in *Advances in Artificial Intelligence*. Cham, Switzerland: Springer, 2015, pp. 410–423.

[38] Y. Censor, "Pareto optimality in multiobjective problems," *Appl. Math. Optim.*, vol. 4, no. 1, pp. 41–59, 1977.

[39] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 966–984, Aug. 2018.

[40] A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, A. Fernandez, and J. Molina, "Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning," *Appl. Sci.*, vol. 9, no. 2, p. 348, 2019.

[41] S. Pellegrinelli and N. Pedrocchi, "Estimation of robot execution time for close proximity human–robot collaboration," *Integr. Comput.-Aided Eng.*, vol. 25, no. 1, pp. 81–96, 2018.

[42] A. Umbrico, A. Cesta, M. C. Mayer, and A. Orlandini, "PLATINUm: A new framework for planning and acting," in *Advances in Artificial Intelligence* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2017, pp. 498–512.

[43] M. Faroni et al., "A layered control approach to human-aware task and motion planning for human–robot collaboration," in *Proc. IEEE Int. Conf. Robot Human Interact. Commun.*, 2020, pp. 1204–1210.

[44] R. Toris et al., "Robot Web tools: Efficient messaging for cloud robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 4530–4537.

[45] E. Villagrossi, N. Pedrocchi, and M. Beschi, "Simplify the robot programming through an action-and-skill manipulation framework," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2021, pp. 1–6.

[46] M. Faroni, N. Pedrocchi, and M. Beschi, "Accelerating sampling-based optimal path planning via adaptive informed sampling," 2022, *arXiv:2208.09318*.

[47] M. Fox and D. Long, "PDDL2.1—An extension to PDDL for expressing temporal planning domains," *J. Artif. Intell. Res.*, vol. 20, pp. 61–124, Dec. 2003.

[48] G. Xiang and J. Su, "Task-oriented deep reinforcement learning for robotic skill acquisition and control," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 1056–1069, Feb. 2021.

[49] S. Sandrini, M. Faroni, and N. Pedrocchi, "Learning action duration and synergy in task planning for human–robot collaboration," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2022, pp. 1–6.

[50] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *Proc. IEEE Int. Conf. Robot Human Inter. Comm.*, 2021, pp. 1017–1023.

[51] M. Raković, N. F. Duarte, J. Marques, A. Billard, and J. Santos-Victor, "The gaze dialogue model: Nonverbal communication in HHI and HRI," *IEEE Trans. Cybern.*, early access, Nov. 29, 2022, doi: 10.1109/TCYB.2022.3222077.

Marco Faroni received the M.S. degree in industrial automation engineering and the Ph.D. degree in mechanical and industrial engineering from the University of Brescia, Brescia, Italy, in 2015 and 2019, respectively.

He is currently a Research Fellow with the Department of Robotics, University of Michigan at Ann Arbor, Ann Arbor, MI, USA. From 2018 to 2022, he was a Researcher with the Institute STIIMA-CNR, National Research Council of Italy, Milan, Italy. His research focuses on planning for robots in challenging scenarios and human–robot collaboration.

Alessandro Umbrico received the M.S. degree in computer science engineering and the Ph.D. degree in computer science and automation from University of Roma TRE, Rome, Italy, in 2012 and 2017, respectively.

He is currently a Researcher with the Institute of Cognitive Sciences and Technologies, National Research Council of Italy, Milan, Italy. His research focuses on human-aware robot controllers pursuing the integration of artificial intelligence planning, knowledge reasoning, and cognitive theories.

Manuel Beschi (Member, IEEE) received the B.S. and M.S. degrees and the Ph.D. degree in control systems from the University of Brescia, Brescia, Italy, in 2008, 2020, and 2014, respectively.

In 2014, he was with STIIMA-CNR, Milan, Italy, and the University of Brescia. Since 2022, he has been an Associate Professor with the Dipartimento di Ingegneria Meccanica e Industriale, University of Brescia and affiliated with CNR-STIIMA. His research focus is on the control and motion planning of mechatronics systems.

Andrea Orlandini received the M.S. degree in computer science engineering and the Ph.D. degree in computer science and automation from the University of Roma TRE, Rome, Italy, in 2002 and 2006, respectively.

He is currently a Senior Researcher with the Institute of Cognitive Science and Technology, National Research Council of Italy, Milan, Italy. He is investigating human-aware task and motion planning issues in human–robot collaboration designing and developing temporally flexible task planning systems. His main research interests are related to long-term autonomy and reliable human–robot interaction investigating the connections between formal methods and automated planning.

Amedeo Cesta received the M.S. degree in electronic engineering and the Ph.D. degree in computer science from the Sapienza University of Rome, Rome, Italy, in 1983 and 1992, respectively.

He is a Research Director of the Institute of Cognitive Sciences and Technologies, National Research Council of Italy, Rome, and a Group Leader with ISTC, Rome. He has been working in artificial intelligence for over 30 years and has founded and still coordinates the Laboratory on Planning and Scheduling Technologies (PSTLab) activities. He has conducted research in multiagent systems, intelligent human–computer interaction, planning and scheduling, and has always pursued the synthesis of innovative decision support systems. His work in AI also emphasizes the real-world aspects of automated reasoning, particularly focusing on research areas like planning for the space domain, technology-based support to older people, and integration of AI techniques in robotics. He was a President from 2013 to 2017 and the Vice-President of AIxIA from 2018 to 2021, the Italian Association for Artificial Intelligence. He has been included in the list of EurAI Fellows (EurAI is the European Association of Artificial Intelligence).

Nicola Pedrocchi received the M.S. degree in mechanical engineering and the Ph.D. degree in applied mechanics and robotics from the University of Brescia, Brescia, Italy, in 2004 and 2008, respectively.

He is currently a Senior Researcher with the Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing, National Research Council of Italy, Milan, Italy. Since 2017, he has been coordinating the Robot Motion Control and Robotized Processes Laboratory. His research field is on motion control for industrial robots for advanced manufacturing.

Open Access funding provided by 'Consiglio Nazionale delle Ricerche-CARI-CARE' within the CRUI CARE Agreement