Learning Action Duration and Synergy in Task Planning for Human-Robot Collaboration

Learning Action Duration and Synergy in Task Planning for Human-Robot Collaboration

Samuele Sandrini[1], Marco Faroni[1], Nicola Pedrocchi[1]

***Abstract*—A good estimation of the actions' cost is key in task planning for human-robot collaboration. The duration of an action depends on agents' capabilities and the correlation between actions performed simultaneously by the human and the robot. This paper proposes an approach to learning actions' costs and coupling between actions executed concurrently by humans and robots. We leverage the information from past executions to learn the average duration of each action and a synergy coefficient representing the effect of an action performed by the human on the duration of the action performed by the robot (and vice versa). We implement the proposed method in a simulated scenario where both agents can access the same area simultaneously. Safety measures require the robot to slow down when the human is close, denoting a bad synergy of tasks operating in the same area. We show that our approach can learn such bad couplings so that a task planner can leverage this information to find better plans.**

***Index Terms*—Human-Robot Interaction; Task And Motion Planning; Task planning; Learning for task planning.**

I. Introduction

Human-Robot Collaboration (HRC) often requires the system to make decisions based on human users' observed or predicted behavior. In such a context, planning and allocating tasks to the human and robot agents are two complex problems, even for tasks composed of a few activities. Two modelling issues are essential. First, human behavior is intrinsically unpredictable and partially uncontrollable [1]. Second, the coupling between the human and the robotic agents is characterized by a large variability. For example, the feasibility and the duration of an action may vary because of the interference between the human and the robot (*e.g.*, safety stops of the robot or even due to path-switch in motion re-planning [2]).

In recent years, task planning and allocation problems for HRC have been investigated. Existing works tried to model human preferences explicitly and ergonomics into planning [3]. For example, [4], [5] proposed a hierarchical agent-based task planner, where complex tasks can be decomposed into simpler actions. This approach can improve the collaborative experience by considering human preferences as social costs [6], [7]. Manufacturing-oriented works focus on process throughput by minimizing the expected duration [8] or planning contingent plans to reduce process errors [9], [10]. A promising approach to deal with uncertainty on the task duration is timeline-based planning [11]. Timeline-based planning explicitly models the duration variability of tasks and finds plans that are robust with respect to it. Timeline-based planning was used in HRC with pre-computed robot motions [12], and online motion planning [13], demonstrating robust plans allow for less frequent re-planning and shorter average task duration.

Most planning-based methods assume a guess of the action cost (e.g., the duration) is available from a domain expert. If this information is unreliable, the task planner reasons on wrong assumptions so that the resulting plans become sub-optimal or even infeasible (leading to frequent re-planning when using a motion re-planner such as [2]). Moreover, the duration guess does not consider possible couplings between the tasks executed concurrently by the human and the robot. Therefore, the task planner neglects each agent's positive or negative effects on the others. For example, if concurrent tasks require the human and the robot to work in the same area, the robot would probably slow down because of safety. If the task planner knew this effect, it could favour pairs of tasks that avoid robot safety slowdowns.

In this work, we propose a method to learn the expected duration of a joint plan. We leverage a minimum-time formulation of the task planning/allocation problem for HRC. Then, we estimate the expected duration of each task from previous executions and learn a synergy coefficient that represents the effect of one task over the other. A synergy coefficient greater than one means that the human task causes a slow down of the robot task. We show that it is possible to cast the estimation of the synergy coefficients into a set of linear regression problems (one for each task). The resulting synergy coefficients can be exploited in task planning and allocation method to select advantageous task couplings. The proposed approach can be used offline – to obtain a guess of the task duration and synergies – or iteratively as the number of executions grows to refine the initial guess over time. We demonstrate the proposed approach in an HRC scenario where a human and a collaborative robot shall perform a sequence of pick-and-place operations. Experiments show that pairs of tasks that drive the nearby agents lead to high synergy coefficients, recognizing favorable and unfavourable pairs.

The paper is organized as follows. Section II defines the task planning problem for HRC systems. Section III builds on that model to formulate the task expected duration in terms of average duration and synergy coefficients. Then, it casts the estimation of such coefficients into a set of linear regression problems. Section IV describes the software architecture to collect and process the data from task executions. Section V applies the proposed methodology to the manufacturing example and shows that the resulting coefficients reflect the good and bad coupling effects of robot safety stops. Finally, conclusions and future works are discussed in Section VI.

This work was partially supported by ShareWork project (H2020, European Commission – G.A. 820807).

[1] STIIMA-CNR - Institute of Intelligent Industrial Technologies and Systems, National Research Council of Italy {name.surname}@stiima.cnr.it

II. PRELIMINARIES

A. Task Planning Problem

A task planning problem can be formalized as an optimization problem. Given a set $\{H, R\}$ of agents, *i.e.*, human and robot, and a set of Tasks $\mathcal{T} = \{\tau_i\}$, the objective of the problem is to obtain a task plan and assignment π that minimizes the duration of the process.

We denote a task by a tuple $\tau = (l, d, t)$, in which:

- $l \in \{H, R\}$ is an assignment variable that specifies which agent can perform the specific task, *i.e.*, human, or robot;
- $d \in \mathbb{R}^+$ is a guess of the task duration;
- $t = \left[t^{\text{start}}\ ;\ t^{\text{end}}\right]$ is an interval with endpoints corresponding to the start and end time.

We refer to $\mathcal{T}^H$ and $\mathcal{T}^R$ as the subsets of $\mathcal{T}$ such that $l = H$ and $l = R$, respectively.

In this context, it is possible to introduce a binary assignment variable that defines the allocation of $\tau_i \in \mathcal{T}$ to the robot (a_i^R) or to the human (a_i^H):

$$\begin{aligned} a_i^R &= \begin{cases} 1, & \text{if task } \tau_i \text{ is assigned to the robot} \\ 0, & \text{otherwise} \end{cases} \\ a_i^H &= \begin{cases} 1, & \text{if task } \tau_i \text{ is assigned to the human} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{1}$$

The duration of a plan π is the maximum between the duration of the robot's and the human's plan, denoted by d_π^H and d_π^R respectively. The duration of each agent's plan can be calculated as:

$$\begin{aligned} d_\pi^H &= \sum_i d_i^H a_i^H \\ d_\pi^R &= \sum_i d_i^R a_i^R \end{aligned} \tag{2}$$

By defining a cost function J that represents the duration of a plan π as:

$$J = \max\{d_\pi^H, d_\pi^R\} \tag{3}$$

the optimal plan π^* is:

$$\pi^* = \underset{\pi}{\operatorname{argmin}}\, J \tag{4}$$

subject to the constraint that $\tau_i \in \mathcal{T}$ can only be assigned during π:

$$a_i^H + a_i^R = 1 \quad \forall\ i \ \text{ s.t. } \ \tau_i \in \mathcal{T} \tag{5}$$

and assignment, temporal, and causal constraints owed to the process requirements[1].

[1]If (4) only involves assignment constraints, the problem is a task allocation; if causal and/or temporal constraints are considered, the problem becomes a task planning&scheduling problem.

III. METHODOLOGY

The task planning problem assumes that a guess of the task duration exists. A domain expert usually provides this guess. However, this information is often unreliable and comes with a sizeable unmodeled uncertainty. Moreover, it does not account for the effect of the task performed by other agents. Indeed, the task duration is a function of the tasks executed simultaneously by the other agent, i.e.:

$$\begin{aligned} d_i^R &= d_i^R(\tau_j^H)\ \ \forall\ j \ \text{ s.t. } \ \tau_j^H \ : \ t_j^H \cap t_i^R \neq \emptyset \\ d_i^H &= d_i^H(\tau_j^R)\ \ \forall\ j \ \text{ s.t. } \ \tau_j^R \ : \ t_j^R \cap t_i^H \neq \emptyset \end{aligned}$$

where τ_j^H and τ_j^R are tasks assigned to humans and robots. If the coupling between tasks is not modelled, the optimal task (4) may be unreliable when executed on the real-world system. To overcome this problem, we extend the formulation given in II-A to include a synergy coefficient in the task duration.

A. Task Planning Problem with Explicit Task Coupling

For each couple of task indices (i, j), a synergy term is introduced for each agent and denoted with $s_{i,j}^R$ for the robot agent and $s_{i,j}^H$ for the human agent. The synergy term denotes the increment of the duration of task τ_i when executed by the robot while the human is executing task τ_j. We define this coefficient as:

$$s_{i,j}^R = \frac{d_{i,j}^R}{\hat{d}_i^R} \tag{6}$$

where $d_{i,j}^R$ is the expected duration of task τ_i when the human executes τ_j and $\hat{d}_i^R$ is the expected value of the duration of τ_i^R for all concurrent tasks τ_j^H. Thus, the duration of a plan π becomes:

$$d_\pi^R = \sum_i \hat{d}_i^R a_i^R \left(\sum_j s_{i,j}^R \delta_{i,j}^R a_j^H \right) \tag{7}$$

where $\delta_{i,j}^R$ represents the ratio of the overlapping time between τ_i^R and τ_j^H with respect to the duration of the task τ_i^R, thus defined as:

$$\delta_{i,j}^R = \frac{D(t_j^H \cap t_i^R)}{D(t_i^R)} \tag{8}$$

and D is a function that calculates the duration of a temporal interval $t = [t^{start}, t^{end}]$:

$$D(t) = \begin{cases} t^{end} - t^{start}, & \text{if } t \neq \emptyset \\ 0, & \text{if } t = \emptyset \end{cases} \tag{9}$$

Equations (6), (7), (8) can be rewritten for the human agent as:

$$s_{i,j}^H = \frac{d_{i,j}^H}{\hat{d}_i^H} \tag{10}$$

$$\delta_{i,j}^H = \frac{D(t_j^R \cap t_i^H)}{D(t_i^H)} \tag{11}$$

$$d_\pi^H = \sum_i \hat{d}_i^H a_i^H \left(\sum_j s_{i,j}^H \delta_{i,j}^H a_j^R \right) \tag{12}$$

Using (7) and (12) in (3), a task planning can minimize the process duration taking into account the coupling effect between concurrent tasks.

B. Synergy and duration estimation

We estimate both task duration and synergy coefficients from experience. Given n executions of a task τ_i, we approximate its duration with its expected value:

$$\begin{aligned} \hat{d}_i^R &\approx \mathbb{E}\left[d_i^R|_k\right] \quad \forall\ k = \{1,\dots,n\} \\ \hat{d}_i^H &\approx \mathbb{E}\left[d_i^H|_k\right] \quad \forall\ k = \{1,\dots,n\} \end{aligned} \tag{13}$$

where $d_i^R|_k$ and $d_i^H|_k$ are the task duration measured in execution k.

Furthermore, it is possible to formalize the problem of estimating task synergy coefficients as a least-square regression problem. For each sample k:

$$D(t_i^H)\Big|_k = \sum_{j=1}^{m} \delta_{i,j}^H\Big|_k s_{i,j}^R d_i^R + T_{\text{idle}}\Big|_k \tag{14}$$

where T_{idle} is the time when the robot is not assigned to any task during τ_i^H and $m = |\mathcal{T}^R|$. Equation (14) can be written in matrix form as:

$$\mathbf{D_i^H} = \mathbf{R^R S_i^R} + \mathbf{T}_{\text{idle}} \tag{15}$$

where:

- $\mathbf{D_i^H} \in \mathbb{R}^{n\times 1}$ is a column vector containing the human execution task duration $D(t_i^H)\Big|_k$;
- $\mathbf{R^R} \in \mathbb{R}^{n\times m}$ is the regression matrix, which takes the following form:

$$\mathbf{R^R} = \begin{bmatrix} \delta_{i,1}^H\Big|_1 & \cdots & \delta_{i,m}^H\Big|_1 \\ \vdots & \ddots & \vdots \\ \delta_{i,1}^H\Big|_k & \cdots & \delta_{i,m}^H\Big|_k \\ \vdots & \ddots & \vdots \\ \delta_{i,1}^H\Big|_n & \cdots & \delta_{i,m}^H\Big|_n \end{bmatrix} d_i^R \tag{16}$$

- $\mathbf{S_i^R} \in \mathbb{R}^{m\times 1}$ is a column vector containing the synergy coefficients to be estimated:

$$\mathbf{S_i^R} = \begin{bmatrix} s_{i,1}^R & s_{i,2}^R & \cdots & s_{i,m}^R \end{bmatrix}^T \tag{17}$$

- $\mathbf{T}_{\text{idle}} \in \mathbb{R}^{n\times 1}$ is a column vector containing the robot idle times $T_{\text{idle}}\Big|_k$.

In conclusion, the solution of the regression problem in (15) is obtained from the following:

$$\mathbf{S_i^R} = \left(\mathbf{R^R}^T\mathbf{R^R}\right)^{-1}\mathbf{R^R}^T\left[\mathbf{D_i^H} - \mathbf{T}_{\text{idle}}\right]. \tag{18}$$

and using the same nomenclature for the human agent:

$$\mathbf{S_i^H} = \left(\mathbf{R^H}^T\mathbf{R^H}\right)^{-1}\mathbf{R^H}^T\left[\mathbf{D_i^R} - \mathbf{T}_{\text{idle}}\right]. \tag{19}$$

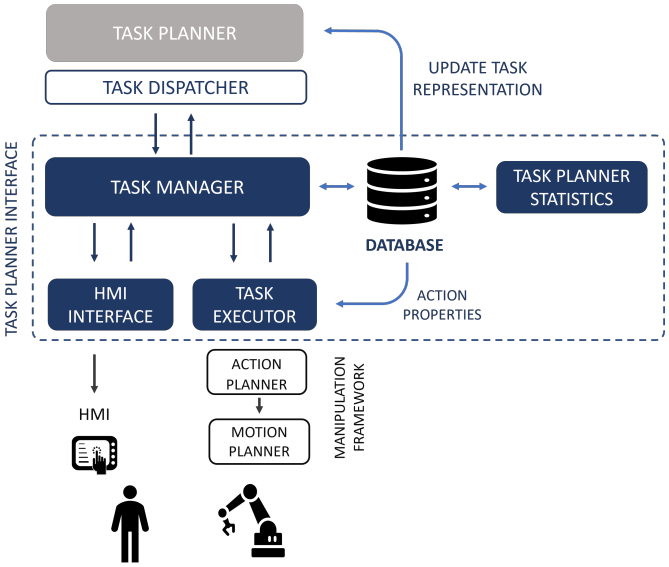

Fig. 1: Proposed framework architecture.

IV. Proposed framework architecture

Figure 1 shows the software architecture developed to measure and estimates action duration and synergies for real-world deployment of our approach.

The *Task planner* is at the highest level and has a symbolic knowledge of the tasks to execute. Based on the temporal constraints between tasks (*i.e.*, precedence) and a-priori durations information, it computes the optimized plan, the scheduling, and the assignment of tasks to agents. This information is used by the *dispatcher*, which receives the plan and sends the request at a lower level; finally, it waits for the result to proceed with the remaining part of the plan.

The *task planner interface* module is at the middle layer. The first module component is the *task manager*, which receives the task request from the dispatcher and queries the database to check for properties associated with the requested symbolic task. In a positive case, the *task manager* sends the task execution request to the task executor of the agent specified by the task planner. When feedback comes from the underlying layer, the *task manager* module interacts with the *database* to update the task results. The core of the *task planner interface* is a database representing the high-level updated Knowledge-Base. The database stores information used at different levels: the definition of high-level task properties, run-time information of duration execution, statistical information of expected duration, and synergy of concurrent tasks between agents.

At the lowest level, there is the single-agent *task executor*, which converts symbolic information associated with tasks into geometric targets. First, it interacts with the database to retrieve the action type (*e.g.*, pick, place, go to) and the symbolic goal associated with the task. This layer has a geometric knowledge of the system and translates the symbolic goal into a sequence of robot movements. To do so, it knows the locations of symbolic goals, integrates a motion planner, and

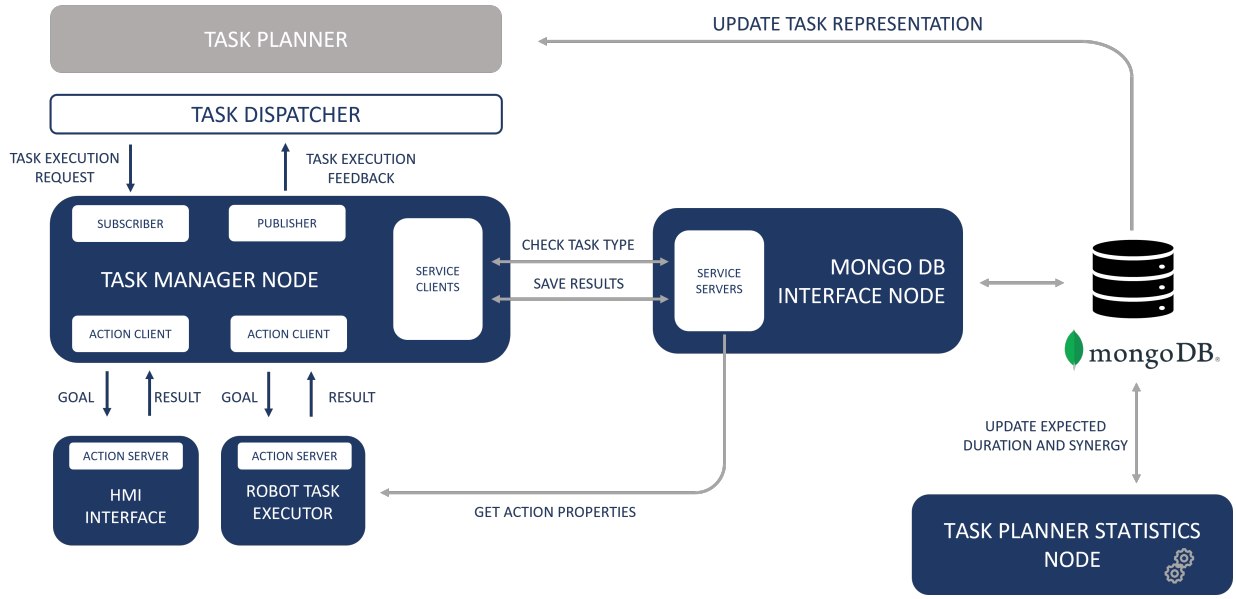

Fig. 2: Software Implementation.

may include an action planner such as in [13]. Once the task execution is finished, the feedback is sent to the layer above.

The *task manager* sends the task request to an *HMI interface*, which communicates to the HMI the information of the task to be executed and waits for the confirmation of execution. Then, it sends the feedback to the *task manager*.

The advantage of this architecture is that, through the presence of the database, it is possible to keep the tasks' representation up-to-date, on which the high-level planning is based. So, it is possible to update actions' costs based on the experience of executions. For this purpose, the *task planner statistics* module calculates the estimated duration and task synergy as described in Section III-B.

A. Software implementation

The functional architecture presented in Section IV was developed as a hybrid Python/C++ library based on ROS [14].

Software development focused on the task planner interface level. The crossroads component is the *task service manager*. As shown in Figure 2, this node exploits the Publisher-Subscriber communication to receive the task execution request from the dispatcher and then send back the feedback.

The communication with the database occurs through an interface node that acts as a service server for the task service manager: a service checks for the existence of task properties associated with the symbolic task, and another service saves the execution results from the layer below.

Communication with the underlying layer, *i.e.*, *task execution* nodes, takes place according to the client-server action model. The task service manager sends a goal containing a symbolic task name of the task to be executed and waits for the results, *e.g.*, duration, task type, and agent information.

The database is composed of different collections, each dedicated to storing different information:

- *task properties*: for each task, it stores a symbolic identifier, the type of action associated with it (*e.g.*, pick, place, go to), a textual description, the agents that can execute it, and the symbolic goals associated with it.
- *Task results*: it is updated in real-time with information from the lower layers about the execution time and the success or failure of the task.
- *Task duration*: it contains the information of expected durations and standard deviation associated with each symbolic task.
- *Task synergy*: for each pair of tasks, it contains the result of the estimation described in Section III-B.

Notice that both *Task duration* and *Task synergy* can be updated synchronously or asynchronously.

In this framework, MongoDB was used as the database [15]. In addition, PyMongo library was used for interaction with MongoDB database in the so-called *Mongo-DB interface node* shown in Figure 2.

At the lowest level, the *task-execution* node interacts with the database interface to retrieve the geometric goal associated with the task. It acts as a client of the manipulation framework [16], to which it sends requests, specifying the type of action to be performed. The manipulation framework defines the sequence of movements required to perform the requested action, solves the corresponding motion planning problem, and finally executes the action.

Finally, the *task planner statistics node* is independent of the framework information flow and provides ROS-Services to update the *task duration* and the *task synergy* collections and make statistical charts. These services can be called synchronously or asynchronously concerning the information flow of the framework.

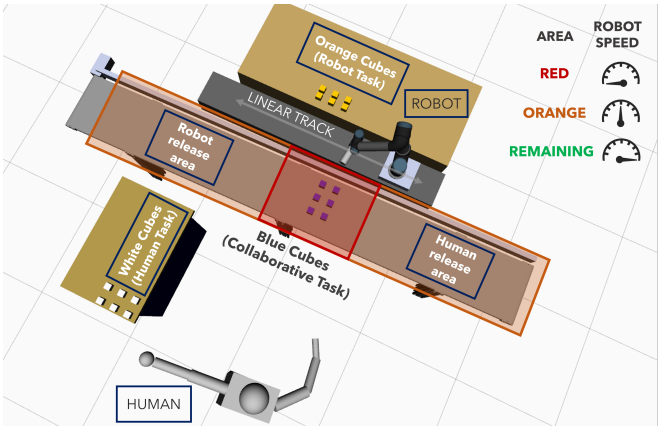

Fig. 3: Simulated cllaborative workcell

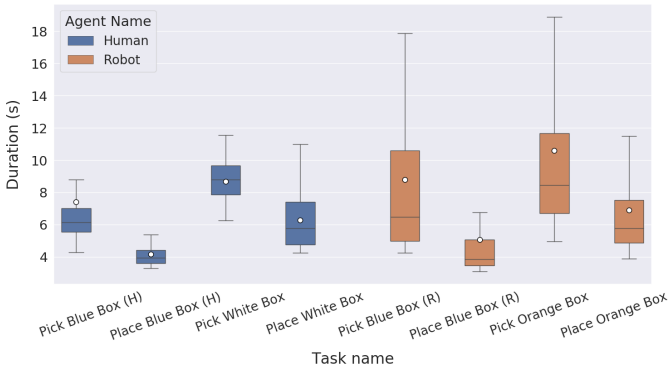

Fig. 4: Task durations grouped by agent, white circles denote the mean.

V. Experiments

We test our approach in a simulated case study of a collaborative workcell (Figure 3) composed of a collaborative robot, UR5, mounted on a linear axis and a human operator. The agents share a portion of the workspace, defining a collaborative workspace. There are six white cubes on the work table accessible only by the human operator, six orange cubes on the work table accessible by the robot, and six blue cubes within the collaborative work area accessible by both agents. Thus, humans can handle only white and blue objects, while robots can handle only orange and blue objects.

Each agent has a dedicated object release area, as illustrated in Figure 3. The process goal is that: (i) the robot picks four orange and two blue boxes by placing them in its release area; (ii) the human picks four white and two blue boxes by arranging them in its release area. Therefore, the set of tasks that the human and the robot can perform is defined by:

$$\mathcal{T}^H = \{\text{Pick Orange Box}, \text{Place Orange Box}, \text{Pick Blue Box (H)}, \text{Place Blue Box (H)}\} \quad (20)$$

$$\mathcal{T}^R = \{\text{Pick White Box}, \text{Place White Box}, \text{Pick Blue Box (R)}, \text{Place Blue Box (R)}\} \quad (21)$$

where $\mathcal{T} = \mathcal{T}^H \cup \mathcal{T}^R$.

The process requires access to the same work area, especially during a simultaneous execution of the *Pick Blue Box* and *Place Blue Box* tasks. The symbolic goal associated with each task does not refer to a specific pick or place slot. Thus, the choice of the best slot associated with a symbolic goal is performed by the action planner of the manipulation framework [16] at runtime by solving a multi-goal motion planning problem.

The ISO/TS 15066 [17] is applied to ensure safety. Three collaborative areas are defined. If any human body part enters inside the red area, the robot is stopped. If the human enters the orange area, the robot moves at 50 % of its nominal speed. Finally, if the human operator moves into the remaining area, the robot is free to move at its nominal speed.

A. Results

In order to estimate the synergy coefficients between tasks, 50 plans are randomly generated respecting the assignment constraints imposed by the agent's domains (20) and (21) and precedence constraints between a pick and place tasks. A simulation run is performed for each recipe, using the framework proposed in Section IV: tasks are executed, and the results are saved to the database.

We exploit the *task planner statistics* node to calculate the expected duration and standard deviation for each task (grouped by type and agent). Results are shown in Figure 4.

For each type of task, outliers are removed using an isolation forest algorithm to reduce noise from the task results before synergy coefficients estimation [18].

We use the estimated duration of each task and saved task results to apply the regression described in (15) and obtain the synergy coefficients estimation between robot and human tasks. We obtain the column vector in (17) by applying the regression for each task. Repeating this procedure for each task, we obtain the synergy matrix shown in Figure 5 (a heatmap graphically highlighting the positive or negative synergy between concurrent tasks). The heatmap shows the synergy coefficients of the robot agent task versus the human tasks. The tasks of the robot (21) are placed in the rows of the heatmap, and on the columns are the human tasks (20). An $s^R_{i,j}$ element of the heatmap reports the duration increment of the robot's task τ_i when simultaneous with the human task τ_j.

If the value $s^R_{i,j}$ is greater than 1, it means that task τ_i performed by the robot is penalized when it is in parallel with task τ_j performed by the human. Vice versa, if $s^R_{i,j}$ is smaller than 1, it means that the coupling between tasks τ_i and τ_j is advantageous in terms of execution time.

The heatmap analysis shows that all the robot tasks are penalized when the human simultaneously performs the *Pick Blue Box* and *Place Blue Box* tasks. Indeed, when the human operator executes *Pick Blue Box*, he accesses a *red area* close to the robot, causing the robot to stop. Conversely, the robot tasks do not slow down or even improve performance when they are executed in parallel with the human tasks *Pick White*

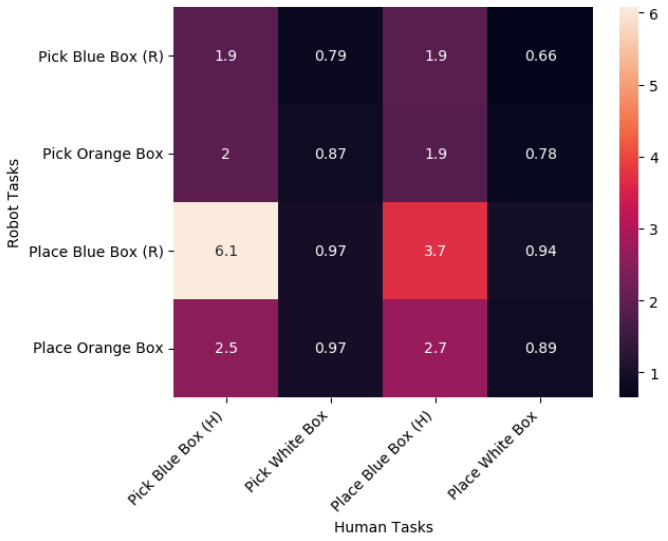

Fig. 5: Robot task Synergy Matrix (each row corresponds to the vector S_i^R obtained from (17))

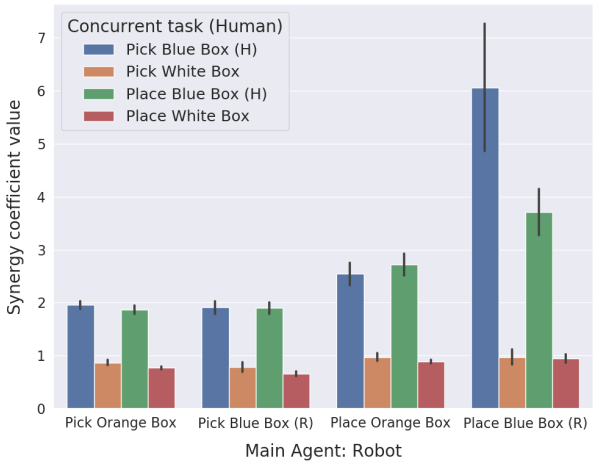

Fig. 6: Synergy coefficients and their standard deviation

Box (which does not cause an access to the collaborative area), and *Place White Box*.

Figure 6 also shows the synergy coefficients and their standard deviation estimates by the regression to validate the statistical significance of the estimate.

VI. Conclusions and Future Works

In this paper, a synergy term that takes into account the coupling between agent tasks is defined. The synergy coefficient is contextualized in a generic task planning problem by including it in the duration cost function.

We validated the approach in a simple HRC scenario where the positive of negative couplings were intuitive in order to show that the proposed method can learn the expected good or bad synergy of pairs of tasks.

Future works will focus on integrating the estimated synergy term in a task planner to demonstrate the reduction of task plan duration in complex scenarios.

Alternative strategies will be compared to access the advantages of the proposed approach.

References

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

[2] C. Tonola, M. Faroni, N. Pedrocchi, and M. Beschi, "Anytime informed path re-planning and optimization for human-robot collaboration," in *Proceedings of the IEEE International Conference on Robot and Human Interactive Communication*, Vancouver (Canada), 2021.

[3] 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 T. Cybernetics*, pp. 1–14, 2020.

[4] 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, 2017.

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

[6] G. Milliez, R. Lallement, M. Fiore, and R. Alami, "Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring," in *ACM/IEEE Int. Conf. Hum.-Rob. Inter.*, 2016.

[7] E. Sebastiani, R. Lallement, R. Alami, and L. Iocchi, "Dealing with online human-robot negotiations in hierarchical agent-based task planner," in *Int. Conf. Automated Plan. Sched.*, 2017.

[8] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *IEEE International Conference on Robot and Human Interactive Communication*, 2021, pp. 1017–1023.

[9] 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.

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

[11] M. Cialdea Mayer and A. Orlandini, "An executable semantics of flexible plans in terms of timed game automata," in *The 22nd International Symposium on Temporal Representation and Reasoning (TIME)*. IEEE, 2015.

[12] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, and T. Tolio, "Motion planning and scheduling for human and industrial-robot collaboration," *CIRP Annals*, vol. 66, pp. 1–4, 2017.

[13] M. Faroni, M. Beschi, S. Ghidini, N. Pedrocchi, A. Umbrico, A. Orlandini, and A. Cesta, "A layered control approach to human-aware task and motion planning for human-robot collaboration," in *Proceedings of the IEEE International Conference on Robot and Human Interactive Communication*, Naples (Italy), 2020.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[15] C. Győrödi, R. Győrödi, G. Pecherle, and A. Olah, "A comparative study: Mongodb vs. mysql," in *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, 2015, pp. 1–6.

[16] E. Villagrossi, N. Pedrocchi, and M. Beschi, "Simplify the robot programming through an action-and-skill manipulation framework," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–6.

[17] "ISO/TS 15066-2016 Robots and robotic devices -: Collaborative robots," International Organization for Standardization, Geneva, CH, Standard, 2016.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.