Stochastic Scheduling for Human-Robot Collaboration in Dynamic Manufacturing Environments

Anders Lager[1,2], Branko Miloradović[2], Giacomo Spampinato[1], Thomas Nolte[2], Alessandro V. Papadopoulos[2]

***Abstract*— Collaborative human-robot teams enhance efficiency and adaptability in manufacturing, but task scheduling in mixed-agent systems remains challenging due to the uncertainty of task execution times and the need for synchronization of agent actions. Existing task allocation models often rely on deterministic assumptions, limiting their effectiveness in dynamic environments. We propose a stochastic scheduling framework that models uncertainty through probabilistic makespan estimates, using convolutions and stochastic max operators for realistic performance evaluation. Our approach employs metaheuristic optimization to generate executable schedules aligned with human preferences and system constraints. It features a novel deadlock detection and repair mechanism to manage cross-schedule dependencies and prevent execution failures. This framework offers a robust, scalable solution for real-world human-robot scheduling in uncertain, interdependent task environments.**

I. Introduction

Collaborative robotics in manufacturing promises enhanced efficiency and adaptability by combining robotic precision with human dexterity [4]. While autonomous systems are effective, their high cost and inflexibility in dynamic settings have led to the rise of human-robot collaboration. This paradigm introduces significant scheduling challenges due to uncertainties in task durations and human preferences. In modern production environments, scheduling is further complicated by cross-schedule dependencies among multiple robots and human operators. For example, a human may need to complete an inspection before a robot proceeds with packaging, or several agents might coordinate transport tasks to avoid bottlenecks, or multi-agent tasks cannot start until all assigned agents have arrived. Mismanagement of these dependencies can lead to deadlocks and inefficient workflows. Additionally, uncertainties arise from robotic delays, environmental obstacles, and human factors such as fatigue, emphasizing the need for robust scheduling frameworks that integrate ergonomic constraints and individual preferences.

This work tackles the Multi-Robot Task Allocation (MRTA) [3] problem in a stochastic, human-in-the-loop context. Differently from traditional deterministic approaches, our method models task and routing durations as random variables with probability distributions. By analytically deriving the makespan distribution using convolutions and stochastic max operators, we obtain bounds that better reflect real-world variability. By incorporating human preferences, our framework not only enhances worker engagement but also achieves a more balanced workload distribution.

Prior work has explored similar challenges. Palmer et al. [15] computed stochastic objective functions under Gaussian assumptions, and our earlier work [6] extended planning to single-robot systems with non-restrictive distributions. Other studies, such as those using Probabilistic Simple Temporal Networks [16] or adaptive scheduling based on real-time human performance [19], address temporal constraints and reactive adaptation, yet they do not fully capture the stochastic and interdependent nature of collaborative manufacturing tasks. Markov Decision Processes models uncertainties in the effects of task executions [1], while we model uncertainties of task and routing durations, which may encompass some relevant aspects of such uncertainties, e.g., retries after failure. While some research has focused on individual agent performance [18], [9] or ergonomic risks [7], these efforts often overlook global task dependencies and the cross-schedule precedence constraints–namely, start-after-completion, start-after-start, complete-after-completion, and complete-after-start–as defined in [8].

Our primary contribution is a novel stochastic scheduling framework for MRTA that explicitly accounts for execution uncertainties and human-centric considerations. A novel deadlock detection and repair mechanism is introduced to manage the inherent cross-schedule dependencies. Finally, a novel heuristic is presented for guiding the search of a Greedy Randomized Adaptive Search Procedure (GRASP) [2] able to find optimized solutions to the scheduling problem in a limited time frame. We validate the correctness and accuracy of the proposed framework using Monte Carlo (MC) simulations, and a deterministic variant of the framework is compared. Further, the GRASP algorithm and a Genetic Algorithm (GA) [11] are evaluated across test instances of varying complexity. A hybrid approach, where GRASP is used to provide initial solutions for GA, is also investigated.

II. Motivating Example

In discrete manufacturing processes [14], activities such as assembly, inspection, and transport are orchestrated to process parts efficiently. Each part or batch is handled through a sequence of tasks–each assigned to an individual or a team of agents–that collectively form the basis for a multi-agent scheduling problem.

This work is funded by The Knowledge Foundation (KKS), projects ARRAY and MARC, by the Swedish Research Council (VR), project PSI.

[1]ABB AB, Västerås, Sweden. [2]Mälardalen University, Västerås, Sweden. Email: {name.surname}@se.abb.com; {name.surname}@mdu.se

Manufacturing Process Model

A typical process model outlines the series of activities needed to transform an order into a finished product. For example, an order may progress through:

- *Fetch*: Retrieving parts from storage locations and deliver them to a workstation.
- *Assemble*: Combining parts to form a product.
- *Inspect*: Evaluating product quality.
- *Transport*: Moving products between workstations.
- *Palletize*: Grouping finished products for shipment.

Each activity is characterized by an input queue (holding parts to be processed) and an output queue (storing processed parts), where the output of one activity becomes the input for the next. A *task* represents the execution of one such activity, including subtasks such as loading, processing, and unloading. The number of parts handled in a task depends on the order specifications and activity constraints. Importantly, aside from the initial order queue, all queues follow a First-In, First-Out (FIFO) rule, and certain activities (e.g., *Assemble*) may require strictly sequential processing due to limited space or resources.

Task Network: A Motivating Example

From the process model, a *task network* is constructed by simulating the flow of parts through the queues and activities. This network is modeled as a directed acyclic graph (DAG) where each node represents a task, and edges denote precedence constraints (PCs). Figure 1 illustrates a typical task network for sequential activities. Key features of this network include:

- The initial task τ_F, which marks the start of the process.
- A primary set of PCs enforcing FIFO access within the same activity—ensuring that, in sequential operations, a task cannot start until its predecessor completes (*start-after-completion*), while in concurrent setups, it may start once the preceding task has begun (*start-after-start*) and complete once the preceding task has finished (*complete after completion*). For the activity connected to the order queue, the start of tasks are not restricted.
- A secondary set of PCs governing transitions between consecutive activities; for instance, a task in the next activity cannot start until the previous activity has fully populated the common queue (*start-after-completion*).
- A tertiary set of PCs arising from queue capacity limits, where a task's completion may depend on subsequent activities removing parts to create the necessary space (*complete-after-start*).

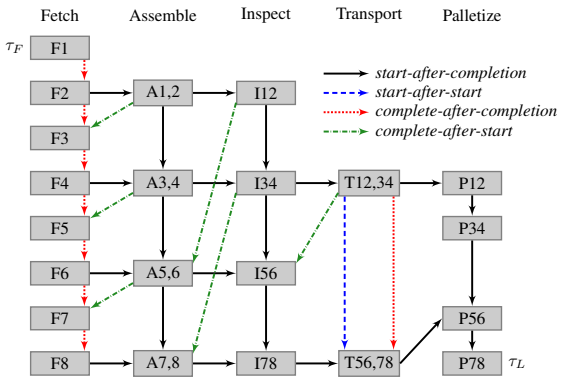

Fig. 1: Task network for a manufacturing scenario.

The process is considered complete when the final task τ_L of the last activity is finished. This motivating example highlights the complexity inherent in task scheduling for manufacturing systems, where managing sequential and interdependent tasks is critical to avoiding bottlenecks and ensuring smooth operations.

III. Defining the Scheduling Problem

This paper formulates the scheduling problem as an optimization model that assigns tasks to agents in a feasible and efficient manner. The model must respect PCs, handle uncertainty in task and routing durations, avoid deadlocks, and incorporate human preferences. The problem falls within the MRTA taxonomy [3], [13], [12] as an XD [ST-MR-TA] (Single-Task robots (ST), Multi-Robot tasks (MR), Time-extended Assignment (TA) with cross-schedule dependencies (XD) as defined in [5]). In the following, we introduce the general assumptions and notation, describe the stochastic modeling of durations and the computation of plan makespans, and finally define the objective function.

General Assumptions and Notation: Let T be the set of all tasks to be completed, and A be the set of all available agents (robots and humans). Each task $\tau \in T$ can be executed by a team of agents. For every task τ, a decision variable

$$f_\tau \in \{1, \dots, m_\tau\}$$

selects one of the m_τ predefined *team formations*. These represent alternative ways to perform τ, specifying the number and types of agents required and their given *roles* in carrying out the task. For example, $f_\tau = 1$ requires a robot and a human performer, while $f_\tau = 2$ requires a robot and a human supervisor. The set of agents assigned to τ is denoted by

$$F_\tau \subseteq A,$$

which must satisfy the requirements of the chosen formation. Agents execute tasks sequentially. For each agent $a \in A$, let

$$\tau_i^a \in T, \quad i = 0, \dots, n_a,$$

represent the i-th task in agent a's sequence (with τ_0^a denoting the starting location or initial state). We define the set of immediate predecessor tasks for task τ as

$$\mathrm{P}_\tau^{\mathrm{apt}} = \{\tau_{i-1}^a \mid \tau_i^a = \tau,\ a \in F_\tau\}.$$

PCs are imposed by the edges of the process model, as illustrated in Figure 1, categorized as follows:

- $\mathrm{P}_\tau^{\mathrm{cps}}$: Tasks whose completions precede the start of τ.
- $\mathrm{P}_\tau^{\mathrm{sps}}$: Tasks whose starts precede the start of τ.
- $\mathrm{P}_\tau^{\mathrm{cpc}}$: Tasks whose completions precede the completion of τ.
- $\mathrm{P}_\tau^{\mathrm{spc}}$: Tasks whose starts precede the completion of τ.

In addition, the overall plan must be free of deadlocks, which can occur if the agents' scheduled routes, in combination with PCs, create cyclic dependencies between tasks.

A. Stochastic Modeling of Durations

Task execution and agent routing times are modeled as independent, nonnegative random variables with generic probabilistic distributions. We define the routing duration for agent $a \in A$ moving from task t to task τ as $R_{a,t,\tau}$. The duration required to execute task τ under team formation f_τ with agents F_τ is defined as A_{τ,f_τ,F_τ}. Initial distributions may be set as uniform (based on expert estimates of minimum and maximum values) and later refined with empirical data from observations in simulation or reality. In the following, we provide some stochastic definitions needed by the proposed scheduling framework.

Definition 1 (Random Variable)**.** *A random variable X on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is a measurable function $X : \Omega \to \mathbb{R}$ such that $\{\omega \in \Omega : X(\omega) = x\} \in \mathcal{F}$ for all $x \in \mathbb{R}$.*

Definition 2 (Probability density function (PDF))**.** *The PDF $f_X(x)$ of a random variable X is defined as:*

$$f_X(x) = \mathbb{P}[\omega \in \Omega \mid X(\omega) = x]$$

Definition 3 (Cumulative distribution function (CDF))**.** *The CDF $F_X(x)$ of a random variable X is defined as:*

$$F_X(x) = \mathbb{P}[\omega \in \Omega \mid X(\omega) \leq x]$$

Definition 4 (Percentile)**.** *The k-th percentile of a probabilistic distribution $f_X(x)$ is defined as:*

$$p_k = \inf\{x : F_X(x) \geq k\}, \quad 0 < k < 1.$$

Definition 5 (First-Order Stochastic Dominance [17])**.** *Consider two random variables, X and Y, with CDFs F_X and F_Y. X has a first-order stochastic dominance over Y, if and only if $\forall x$, $F_X(x) \leq F_Y(x)$, and $\exists x$, $F_X(x) < F_Y(x)$. The stochastic dominance is denoted as $X \geq_{st} Y$.*

Definition 6 (Independence)**.** *Two random variables X and Y are independent if the pair of events $\{X = x\}$ and $\{Y = y\}$ are independent for all $x, y \in \mathbb{R}$. Formally,*

$$\mathbb{P}[X = x, Y = y] = \mathbb{P}[X = x]\mathbb{P}[Y = y], \quad \forall\, x, y \in \mathbb{R}.$$

Definition 7 (Convolution or sum of random variables)**.** *If X and Y are independent random variables on $(\Omega, \mathcal{F}, \mathbb{P})$, then $Z = X + Y$ has probability density function, when X and Y are discrete random variables*

$$\mathbb{P}[Z = z] = \sum_{x=-\infty}^{\infty} f_X(x) f_Y(z - x), \quad \forall\, z \in \mathbb{Z}, \tag{1}$$

and for continuous random variables:

$$\mathbb{P}[Z = z] = \int_{-\infty}^{\infty} f_X(x) f_Y(z - x)\, \mathrm{d}x. \tag{2}$$

Definition 8 (Maximum between random variables [6])**.** *If X and Y are independent random variables on $(\Omega, \mathcal{F}, \mathbb{P})$, then $Z = \max(X, Y)$ has cumulative probability function*

$$F_Z(z) = F_X(z) F_Y(z)$$

B. Plan Duration Computation

The completion time of a task has a probability distribution which is determined by the durations along the *critical paths*–ordered sequences of routing and task durations from the start of the plan to that task. We denote this completion time as $K(\tau)$ and define:

$$K(\tau) = \max\big(\mathcal{C}(\tau)\big), \tag{3}$$

where $\mathcal{C}(\tau)$ is the set of computed completion durations along all critical paths for task τ. To compute these durations, we recursively define the start and completion duration operators. The start duration operator $\mathcal{S}(\tau)$ is given by:

$$\mathcal{S}(\tau) = \mathcal{M}\Bigg(\mathcal{F}\Bigg(\bigcup_{t \in \mathrm{P}_\tau^{\mathrm{apt}}} \Big(\mathcal{C}(t) + \max\Big(\bigcup_{a \in F_\tau \cap F_t} R_{a,t,\tau}\Big)\Big) \cup \bigcup_{t \in \mathcal{K}(\mathrm{P}_\tau^{\mathrm{cps}})} \mathcal{C}(t) \cup \bigcup_{t \in \mathcal{K}(\mathrm{P}_\tau^{\mathrm{sps}})} \mathcal{S}(t)\Bigg)\Bigg), \tag{4}$$

and the completion duration operator $\mathcal{C}(\tau)$ is defined as:

$$\mathcal{C}(\tau) = \mathcal{M}\Bigg(\mathcal{F}\Bigg(\mathcal{S}(\tau) + A_{\tau,f_\tau,F_\tau} \cup \bigcup_{t \in \mathcal{K}(\mathrm{P}_\tau^{\mathrm{cpc}})} \mathcal{C}(t) \cup \bigcup_{t \in \mathcal{K}(\mathrm{P}_\tau^{\mathrm{spc}})} \mathcal{S}(t)\Bigg)\Bigg), \tag{5}$$

with initial durations, $\{\mathcal{S}(\tau), \mathcal{C}(\tau) | \tau = \tau_0^a\} = 0\ \forall a \in A$, and where a sum of a set and a single element is performed element-wise. $\mathcal{M}(\cdot)$ is an operator that merges durations from a set into a smaller representative set. $\mathcal{F}(\cdot)$ is an operator that prunes inferior durations from a set, where X is inferior if $\exists Y | \max(X, Y) = Y$. The operator $\mathcal{K}(\cdot)$ selects the subset of preceding tasks that are critical due to causality, e.g., a preceding task is not critical if assigned to the same agent. The purpose of these operators is to enhance the computational efficiency and accuracy by reducing $\mathcal{S}(\tau)$ and $\mathcal{C}(\tau)$. Further efficiency is gained by avoiding the recomputation of $\mathcal{S}(\tau)$ and $\mathcal{C}(\tau)$ with the same τ. Table I provides examples illustrating the action of the merge operator, $\mathcal{M}(\cdot)$, where X and Y are composed of independent durations. A merge of them occurs if their combined duration, $\max(X, Y)$, can be expressed with independent operands only.

C. Plan Duration Bounds

Exact analytical computation of $K(\tau)$ is challenging due to the possibility of dependencies among completion dura-

TABLE I: Examples of the merge operator, $\mathcal{M}(\cdot)$

#	X	Y	$\mathcal{M}(\{X, Y\})$
1	$A + B$	$A + C$	$\{A + \max(B, C)\}$
2	A	$A + B$	$\{Y\}$
3	$B + D$	$\max(B, C)$	$\{\max(B + D, C)\}$
4	B	$\max(B + C, D) + E$	$\{Y\}$
5	$B + D$	$\max(B, C) + E$	$\{X,\ Y\}$

tions, $\mathcal{C}(\tau)$. Instead, the estimate $K_e(\tau)$ is computed as

$$K_e(\tau) = \max\big(\mathcal{C}_{\text{ind}}(\tau)\big), \tag{6}$$

where $\mathcal{C}_{\text{ind}}(\tau)$ is the same set as $\mathcal{C}(\tau)$ under the assumption of independence. We define a lower bound $K_l(\tau)$ with CDF

$$F_{K_l(\tau)} = \min_{x \in \mathcal{C}_{\text{ind}}(\tau)} F_x, \tag{7}$$

where F_x is the CDF of x. Note that the computed completion durations of all critical paths $\mathcal{C}(\tau)$ may generally have shared tasks that will make the duration of the individual path dependent on the others. We show that considering the duration of the different critical paths as independent random variables, $\mathcal{C}_{\text{ind}}(\tau)$, provides *safe bounds*, in terms of stochastic dominance, for the case of dependent variables.

Theorem 1. *Assume that $K(\tau)$ from Eq. 3 represents the exact task duration and $K_e(\tau)$ from Eq. 6 is its estimate. Further, let $K_l(\tau)$ be defined as above. Then,*

$$K_l(\tau) \leq_{st} K(\tau) \leq_{st} K_e(\tau),$$

where $\leq_{st}$ denotes the stochastic order.

Proof. Since $\mathcal{C}(\tau)$ is composed of nonnegative, continuous random variables combined via $+$ and $\max$ operators, the resulting durations are exact and positively dependent (i.e., they are non-decreasing functions of any shared random variable). Eq. (7) and Theorem 2 (in the Appendix) imply that if $\mathcal{C}(\tau)_{com}$ denotes a comonotonic version of $\mathcal{C}(\tau)$, then

$$\max(\mathcal{C}(\tau)_{com}) \leq_{st} \max(\mathcal{C}(\tau)) \leq_{st} \max(\mathcal{C}_{\text{ind}}(\tau)).$$

Thus, $K_l(\tau) \leq_{st} K(\tau) \leq_{st} K_e(\tau)$. □

D. Human Preferences and Ergonomic Constraints

For a human agent a, task allocation considers ergonomic and motivational factors. An idle time quota, $Q_a^I \in [0,1]$, can be specified, defining a guaranteed minimum level of free time that can be spent on breaks or other (non-planned) tasks. Preferred activities and roles are specified with indicator $\mathcal{L}_{a,\tau,f_\tau}$ to be 1 if task τ with team formation f_τ is preferred, and 0 otherwise. An activity time quota, $Q_a^P \in [0,1]$, defines the desired proportion of the active time to be spent on preferred activities and roles. The total activity time for an agent, including planned tasks and routing, is given by

$$T_a = \sum_{i=1}^{n_a} \Big(R_{a,\tau_{i-1},\tau_i} + A_{\tau_i,f_{\tau_i},F_{\tau_i}} \Big),$$

and the total preferred activity time is

$$P_a = \sum_{i=1}^{n_a} \mathcal{L}_{a,\tau_i,f_{\tau_i}} \Big(R_{a,\tau_{i-1},\tau_i} + A_{\tau_i,f_{\tau_i},F_{\tau_i}} \Big).$$

We require that the estimated median idle time quota for human agent a, denoted by

$$Q_a^{I*} = \frac{p_{50}(K_l(\tau_L)) - p_{50}(T_a)}{p_{50}(K_e(\tau_L))}, \tag{8}$$

satisfies

$$Q_a^{I*} \geq Q_a^I, \quad \forall\, a \in H, \tag{9}$$

where $p_{50}(\cdot)$ returns the median and $H \subseteq A$ is the set of human agents. Similarly, the median preferred activity time quota

$$Q_a^{P*} = \frac{p_{50}(P_a)}{p_{50}(T_a)}$$

should satisfy

$$Q_a^{P*} \geq Q_a^P, \quad \forall\, a \in H, \tag{10}$$

Unlike constraint Eq. (9), this is considered a soft constraint.

E. Objective Function

The primary objective is to minimize the makespan, represented by $K_e(\tau_L)$—the upper bound completion time of the final task τ_L. Given a specified risk level $k \in [0, 100]$, we aim to minimize the k-th percentile of the makespan, $p_k(K_e(\tau_L))$. A low k promotes plans with better chances to reach lower makespans (higher risk) while a high k promotes plans with better chances to avoid higher makespans (lower risk). To account for human-centric constraints, penalty terms are added when the estimated quotas in Eqs. (9) and (10) are not met. The objective function to be minimized is:

$$J = p_k(K_e(\tau_L)) + \sum_{a \in \{H | n_a > 0\}} \Big[\big[Q_a^P - Q_a^{P*}\big]^+ \, p_{50}(K_e(\tau_L)) + \big[Q_a^I - Q_a^{I*}\big]^+ \, p_{50}(T_a) \Big], \tag{11}$$

where $[x]^+ := \max(x, 0)$.

IV. Solving the Scheduling Problem

Due to the stochastic representation of the makespan, developing an exact solution algorithm for the scheduling problem is highly challenging. Consequently, we adopt variants of two metaheuristic approaches to search for sub-optimal solutions, namely a GRASP algorithm with a novel efficient search heuristics and a GA approach inspired by [10]. These approaches are designed to explore the solution space while handling uncertainty, PCs, deadlocks and human preferences.

A. GRASP Algorithm

The GRASP algorithm builds a solution incrementally, starting from an empty solution and in each step selecting and scheduling a *combination* of one task, one related team formation, and a matching agent set. A task is selectable if unscheduled and all of the task's predecessors are already scheduled in the partially built solution. Selection is guided by a heuristic function (see Eq. (12)) that for any selectable combination estimates the impact of scheduling the related task at the end of the sequences of the related agents in the partial solution. From the best-ranked alternatives, a randomized selection is made, with higher-ranked combinations having higher selection probabilities. The selected task is denoted by τ_i for step $i \in \{1, 2, \ldots, |T|\}$. After the last step $|T|$, the resulting solution is repaired for deadlocks (see Section IV-C) and compared against the current best solution using the objective function defined in Eq. (11).

GRASP Heuristic Function: In step i, the heuristic function $\mathcal{H}(\tau_i, f_{\tau_i}, F_{\tau_i})$ estimates the incremental cost of scheduling a selectable combination for τ_i, f_{τ_i} and F_{τ_i}. It sums estimates of makespan increase and deviations from preferred idle and activity times using (non-stochastic) median values of task and routing durations, defined as:

$$\mathcal{H}(\tau_i, f_{\tau_i}, F_{\tau_i}) = \left[C(\tau_i) - \max_{\{\tau_j | j<i\}} C(\tau_j)\right]^+ + \sum_{a \in F_{\tau_i} \cap H} \left[\left[T_a - C(\tau_i)(1-Q_a^I)\right]^+ + \left[Q_a^P T_a - P_a\right]^+\right], \quad (12)$$

where

$$C(\tau) = \max\left\{S(\tau) + p_{50}(A_{\tau,f_\tau,F_\tau}),\ \max_{t \in \mathrm{P}_\tau^{\mathrm{cpc}}} C(t),\ \max_{t \in \mathrm{P}_\tau^{\mathrm{spc}}} S(t)\right\},$$

$$S(\tau) = \max\left\{\max_{t \in \mathrm{P}_\tau^{\mathrm{apt}}}\left(C(t) + \max_{a \in F_\tau \cap F_t} p_{50}(R_{a,t,\tau})\right),\ \max_{t \in \mathrm{P}_\tau^{\mathrm{cps}}} C(t),\ \max_{t \in \mathrm{P}_\tau^{\mathrm{sps}}} S(t)\right\}$$

and $S(\tau_0^a) = C(\tau_0^a) = 0\ \forall a \in A$.

B. Genetic Algorithm (GA)

In the GA framework, an *individual* represents a feasible solution and is encoded by a chromosome that includes grounded decision variables, such as the sequence of tasks for each agent (with multi-robot tasks appearing in multiple agent sequences) and the selected team formation for each task. The algorithm proceeds as follows:

1) **Initialization:** An initial *generation*, i.e., a first population of individuals, is generated by assigning tasks in a random order to randomly selected team formations with corresponding agent sets. These assignments are then adjusted to ensure that all intra-schedule PCs are satisfied. Deadlocks are resolved as described in Section IV-C.
2) **Creating Next Generation:** A set of mutation operators is applied to randomly selected individuals from the current generation. Although these mutations preserve intra-schedule PCs, they may temporarily introduce deadlocks, which are subsequently repaired.
3) **Evaluation and Sorting:** The new generation is evaluated using the objective function (see Eq. 11) and sorted accordingly. An elitism strategy retains a small fraction of the best-performing individuals from the previous generation, replacing the least fit individuals.

Steps 2-3 are repeated until convergence is achieved. To enhance diversity, four mutation operators are used. *Task-to-Idle* moves a task from one agent to another with most surplus idle time, inserting it at a valid position. *Team Formation* changes a task's team formation—retaining current agents if possible, otherwise reassigning them. *Task Insert* transfers a task to a different agent compatible with the team formation. *Task Swap* exchanges tasks between two agents while potentially changing their team formations.

Algorithm 1: Detect and repair deadlocks

function DETECTANDREPAIRCYCLES
  $DOfound \leftarrow$ **false**
  while $\neg DOfound$ **do**
    for $t \in T$ **do** $L_t \leftarrow \emptyset$ ▷ Reset lock sets
    for $t \in T$ **do** $D_t \leftarrow$ **false** ▷ Reset DO flags
    $d \leftarrow 0$ ▷ Reset number of ordered tasks
    SEARCHDO(τ_F) ▷ Search from the first task
    if $\exists a, x, t | x = \tau_i^a \wedge t = \tau_j^a \wedge i < j \wedge x \in L_t \wedge x \not\gtrdot t$ **then**
      if $\exists$ alternative such a, x, t combinations: a', x', t' **then**
        Select $a, x, t | x \not\succ x'$ for all combinations
      SWAPTASKORDER(a,x,t) ▷ Remove cycle with agent a
      for $\{a', x', t' | x' = x \wedge a' \neq a \wedge (t \prec t' \vee t = t')\}$ **do**
        SWAPTASKORDER(a',x,t') ▷ Avoid new cycle with a'
    else $DOfound \leftarrow$ **true** ▷ Done. No cycles in schedule
function SEARCHDO(τ)
  $D_\tau \leftarrow$ **true** ▷ DO for τ is decided
  $d \leftarrow d + 1$
  $\mathcal{D}_d = \tau$ ▷ Store the dependency order of tasks
  if $\exists \tau_o | \tau \in \mathrm{P}_{\tau_o}^{\mathrm{cps}} \wedge \neg$SAMEACTIVITY($\tau_o$,$\tau$) **then**
    $\chi = \{t | t \in \mathrm{P}_{\tau_o}^{\mathrm{apt}} \cup \mathrm{P}_{\tau_o}^{\mathrm{cps}} \cup \mathrm{P}_{\tau_o}^{\mathrm{sps}} \cup \mathrm{P}_{\tau_o}^{\mathrm{cpc}} \cup \mathrm{P}_{\tau_o}^{\mathrm{cps}} \wedge \neg D_t\}$
    for $t \in \chi$ **do** $L_t \leftarrow L_t \cup \tau_o$ ▷ τ_o is locked by t
    if $\chi = \emptyset$ **then** SEARCHDO(τ_o) ▷ Search the unlocked τ_o
  if $\exists \tau_s | \tau \in \mathrm{P}_{\tau_s}^{\mathrm{cps}} \cup \mathrm{P}_{\tau_s}^{\mathrm{sps}} \cup \mathrm{P}_{\tau_s}^{\mathrm{cpc}} \wedge$SAMEACTIVITY($\tau_s$,$\tau$) **then**
    $\chi = \{t | t \in \mathrm{P}_{\tau_s}^{\mathrm{apt}} \cup \mathrm{P}_{\tau_s}^{\mathrm{cps}} \cup \mathrm{P}_{\tau_s}^{\mathrm{sps}} \cup \mathrm{P}_{\tau_s}^{\mathrm{cpc}} \cup \mathrm{P}_{\tau_s}^{\mathrm{cps}} \wedge \neg D_t\}$
    for $t \in \chi$ **do** $L_t \leftarrow L_t \cup \tau_s$ ▷ τ_s is locked by t
    if $\chi = \emptyset$ **then** SEARCHDO(τ_s) ▷ Search the unlocked τ_s
  for $t \in L_\tau$ **do**
    $L_\tau \leftarrow L_\tau \setminus t$ ▷ τ no longer locks t
    if $\nexists x | t \in L_x$ **then** SEARCHDO(t) ▷ Search the unlocked t

C. Deadlock Search and Repair

Deadlocks occur when agent schedules introduce cyclic dependencies that violate the acyclic structure of the task network. To eliminate these deadlocks (see Algorithm 1 for detailed procedure), we employ a two-stage procedure.

a) Dependency Order (DO) Search:: Starting from the initial task τ_F, the algorithm recursively tries to determine a valid dependency order for all tasks, from τ_F to τ_L, taking into account both PCs and agent schedules. This process identifies *lock-sets* L_t, which consist of tasks that cannot be scheduled until task t's dependency order is decided.

b) Detect and Repair Cycles:: Starts with a DO search. If circular dependencies are detected (i.e., some lock-sets remain unresolved), a repair step is invoked to identify a pair of tasks t and $x \in L_t$ that are assigned to the same agent and lack a definitive ordering. Their order is swapped in the agent's schedule and if x is a multi-agent task, the order of x with a corresponding task t' in another agent's schedule may also be swapped to avoid creating a new cycle. The DO search is reinitiated and the process is rerun iteratively, until no lock-sets remain and a valid dependency order is found.

V. EVALUATION

This section evaluates the proposed stochastic computation approach using Monte Carlo simulations and a deterministic counterpart. It also compares GRASP with two variants of the GA approach, for solving the scheduling problem. The

TABLE II: Use case settings.

Use case	#T	#R	#H	Use case	#T	#R	#H
1A	11	2	1	1B	11	3	2
2A	22	2	1	2B	22	3	2
3A	33	2	1	3B	33	3	2
4A	44	2	1	4B	44	3	2
5A	55	2	1	5B	55	3	2

*(#T - no of tasks; #R - no of robots; #H - no of humans)

Python code used to obtain the results is available online[1].

A. Numerical Results

Ten use cases, summarized in Table II, are set up with different numbers of tasks and agents. Routing and task durations are modeled as uniform distributions with randomized intervals. While these data are simplified, the methodology supports a real-world scenario with unrestricted distributions modeled from empirical data. All use cases are based on a process model exemplified in Section II. For use cases 2A and 2B, each with 22 tasks, the constructed task network is illustrated in Figure 1. For larger problem instances, the increase of tasks is accomplished by an increased length of the order list, which expands the task network vertically. All tasks have 3 alternative team formations with different agent combinations: a single robot, a single human, or a robot with a human supervisor. Assemble tasks have a 4th team formation with a collaborating robot-human pair. Desired human idle time is 40%, and all humans prefer *Assemble* activities in any team formation and the role of supervisor for all activities except *Fetch*. In a real-world scenario, the methodology enables different value selection strategies, e.g., from direct human input. The selected risk value, $k = 50$, i.e., a medium risk in the objective Eq. (11).

MC simulations are used to validate the makespan computation approach by representing the ground truth. An MC distribution is generated from 100 thousands of deterministic makespan computations, using Algorithm 2, where input task and routing durations are randomized from their modeled distributions. To account for PCs, Algorithm 2 accumulates task and routing durations in the dependency order identified

[1] https://github.com/anders-lager/stochastic-scheduling

Algorithm 2: Deterministic makespan computation

function ComputeMakespan
  $S_0 \leftarrow 0$; $C_0 \leftarrow 0$; $i \leftarrow 0$
  while $i \leq |T|$ **do**
    $i \leftarrow i + 1$
    $\tau \leftarrow \mathcal{D}_i$ ▷ Visit tasks in dependency order
    for $a \in F_\tau$ **do**
      $t \leftarrow$ AgentPreviousTask(a,τ)
      $S_i = \max(C_t + R_{a,t,\tau}, S_i)$
    for $t \in \mathrm{P}_\tau^{\mathrm{cps}}$ **do** $S_i = \max(C_t, S_i)$
    for $t \in \mathrm{P}_\tau^{\mathrm{sps}}$ **do** $S_i = \max(S_t, S_i)$
    $C_i = S_i + A_{\tau,f_\tau,F_\tau}$
    for $t \in \mathrm{P}_\tau^{\mathrm{cpc}}$ **do** $C_i = \max(C_t, C_i)$
    for $t \in \mathrm{P}_\tau^{\mathrm{spc}}$ **do** $C_i = \max(S_t, C_i)$
  return $C_{|T|}$

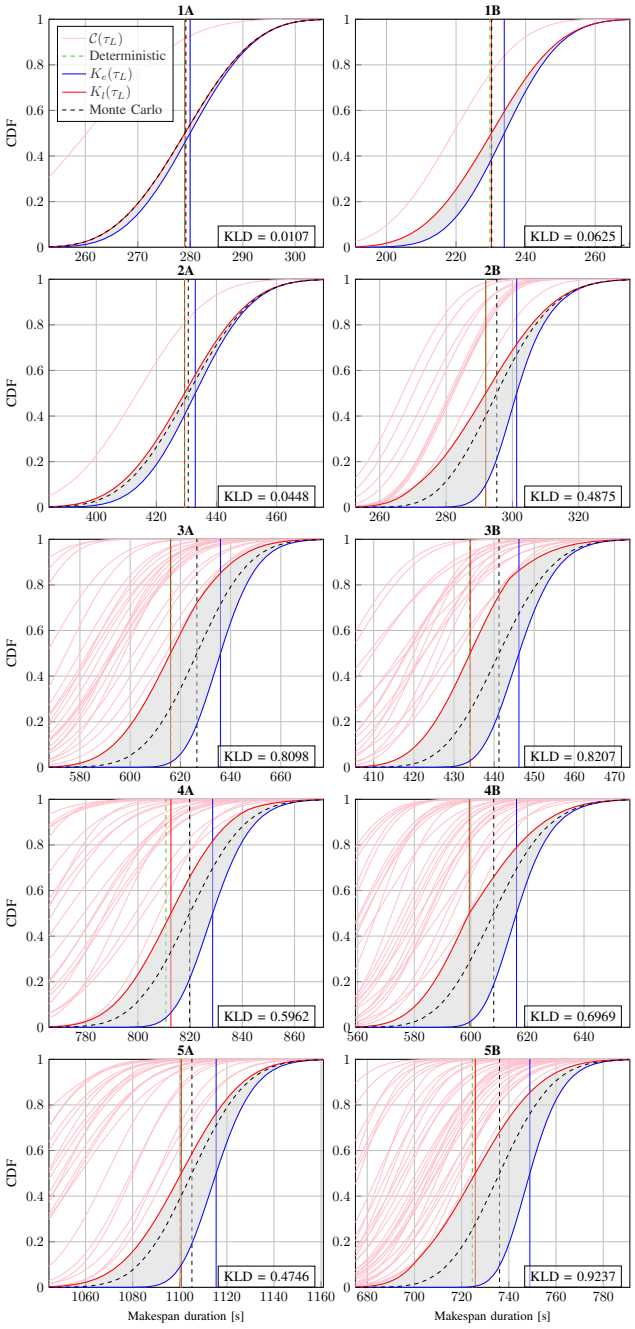

Fig. 2: Makespans of best schedules found.

by the deadlock Algorithm 1. Makespans for the best schedules found (with any algorithm) are illustrated in Figure 2. They include makespan bounds, completion durations for critical paths, and MC distributions. The difference of the makespan bounds, $K_l(\tau_L)$ and $K_e(\tau_L)$, is illustrated by the gray area between their CDFs and by their Kullback-Leibler Divergence (KLD). Additionally, the graphs indicate the makepan of a deterministic approach computed with Algorithm 2 using median values of the modeled input task and routing durations.

The parameters of GRASP and GA were manually tuned to limit the convergence time while providing good solutions. GRASP randomly selects a combination from the 4 best

heuristically ranked alternatives, with selection probabilities 8/15, 4/15, 2/15, and 1/15 from the best to the fourth best. GA uses a population of 200 with 5% elites, and each mutation type occurs with a probability of 30%. The algorithms are run 20 times for each use case with different seeds. Each run is limited to 270 minutes. A comparative analysis of algorithmic convergence over time is presented in Figure 3. It includes a warm-started variant of GA, denoted GAW, whose initial generation is created with GRASP solutions.

B. Evaluation discussion

The makespan graphs show the MC distribution being wedged between $K_l(\tau_L)$ and $K_e(\tau_L)$ for all use cases, as expected from Theorem 1. Consequently, the usage of $K_e(\tau_L)$ in the objective, Eq 11, guarantees the makespan

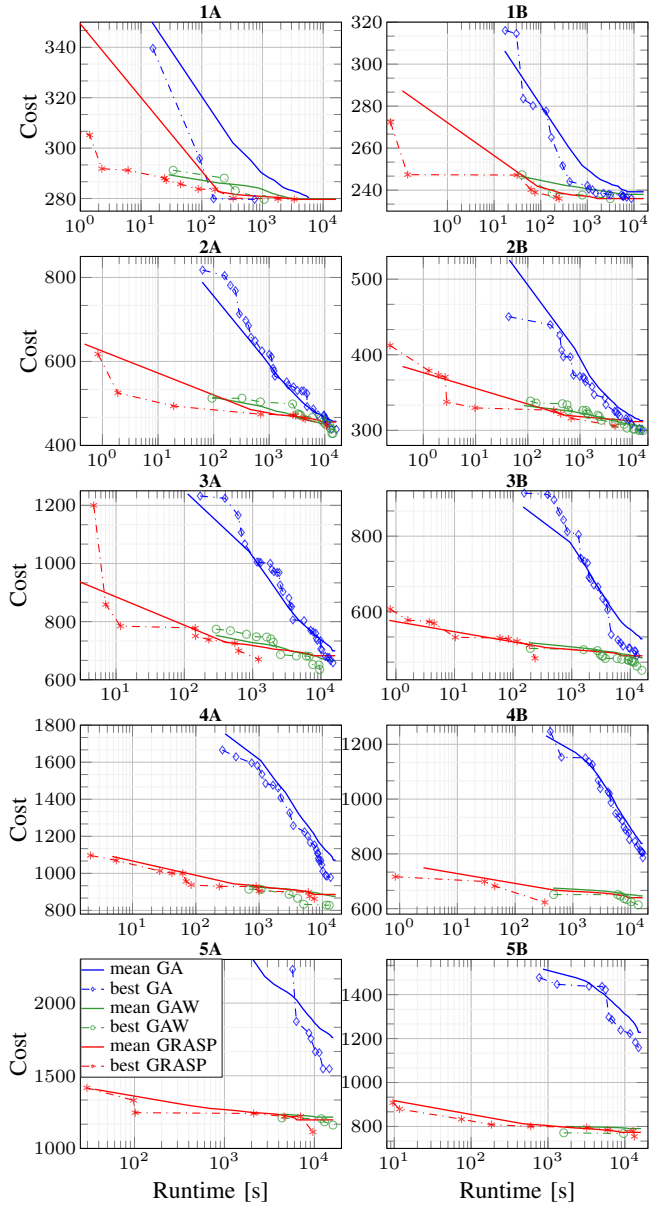

Fig. 3: Convergence of scheduling algorithms.

is never underestimated for a schedule. Additionally, the usage of $K_l(\tau_L)$ for the estimate of idle time quota, Eq. 8, guarantees human idle time never is overestimated. The estimation accuracy, e.g., represented by the KLD value, depends on the $\mathcal{C}(\tau_F)$ distributions and their degree of mutual dependence. For these examples, the accuracy has a variation among use cases without a clear correlation with problem size. The deterministic approach tends to underestimate the median makespan and generally suffers from the missing representation of uncertainty. The fast convergence of GRASP compared to GA demonstrates the efficiency of the proposed GRASP heuristics. The capacity of GRASP to find good solutions faster is demonstrated in all use cases. The GAW approach clearly benefits from the GRASP performance, with a convergence starting from a more optimized population than GA. In general, GA is outperformed by both GRASP and GAW. For the smallest use cases $1A$ and $1B$, all algorithms converge to the solutions of equal cost, but the actual plans are not necessarily the same. GAW finds the best solutions for intermediate use cases $2A$ through $4B$, while GRASP finds the best solutions for $5A$ and $5B$. On average, the difference between GRASP and GAW is generally small without a clear advantage to either of them unless a really fast solution is needed, which would make GRASP the better alternative.

VI. Conclusion

In this paper, we presented a framework for scheduling in human-robot collaborative manufacturing environments. Our approach models the complex interplay between stochastic task and routing durations, precedence constraints, and human-centric factors such as ergonomic preferences and idle time requirements. By formulating the scheduling problem as an optimization model that accounts for uncertainty and potential deadlocks, we have established a robust basis for effective multi-agent task allocation.

Our contributions demonstrate that the integration of stochastic modeling with human-centric scheduling can significantly enhance the reliability of scheduling strategies in resilient manufacturing systems by producing realistic plans. Future work will focus on extending the framework to support real-time scheduling adjustments and incorporating more detailed risk models, as well as validating the approach in real-world industrial settings.

Overall, the proposed framework advances the state of the art in collaborative robotics by providing a scalable, adaptable solution to the challenges inherent in human-robot manufacturing systems.

References

[1] S. Choudhury, J. K. Gupta, M. J. Kochenderfer, D. Sadigh, and J. Bohg. Dynamic multi-robot task allocation under uncertainty and temporal constraints. *Autonomous Robots*, 46(1):231–247, 2022.

[2] T. A. Feo, M. G. Resende, and S. H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 1994.

[3] B. P. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The Int. Journal of Robotics Research*, 23(9):939–954, 2004.

[4] L. Gualtieri, I. Palomba, E. J. Wehrle, and R. Vidoni. The opportunities and challenges of sme manufacturing automation: safety and ergonomics in human–robot collaboration. *Industry 4.0 for SMEs: Challenges, opportunities and requirements*, pages 105–144, 2020.
[5] G. A. Korsah, A. Stentz, and M. B. Dias. A comprehensive taxonomy for multi-robot task allocation. *The Int. Journal of Robotics Research*, 32(12):1495–1512, 2013.
[6] A. Lager, B. Miloradović, G. Spampinato, T. Nolte, and A. V. Papadopoulos. Risk-aware planning of collaborative mobile robot applications with uncertain task durations. In *IEEE Int. Conf. Robot and Human Interactive Communication*, pages 1191–1198, 2024.
[7] Y. Y. Liau and K. Ryu. Genetic algorithm-based task allocation in multiple modes of human–robot collaboration systems with two cobots. *The Int. Journal of Advanced Manufacturing Technology*, 119(11):7291–7309, 2022.
[8] M. Lombardi and M. Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 2012.
[9] C. Messeri, G. Masotti, A. M. Zanchettin, and P. Rocco. Human-robot collaboration: Optimizing stress and productivity based on game theory. *IEEE Robotics and Automation Letters*, 6(4):8061–8068, 2021.
[10] B. Miloradović, B. Çürüklü, M. Ekström, and A. V. Papadopoulos. A genetic algorithm approach to multi-agent mission planning problems. In *Operations Research and Enterprise Systems*, pages 109–134, 2020.
[11] B. Miloradović, B. Çürüklü, M. Ekström, and A. V. Papadopoulos. GMP: A genetic mission planner for heterogeneous multirobot system applications. *IEEE Transactions on Cybernetics*, 52(10), 2021.
[12] B. Miloradović, M. Frasheri, B. Cürüklü, M. Ekström, and A. V. Papadopoulos. TAMER: Task allocation in multi-robot systems through an entity-relationship model. In *Int. Conf. Principles and Practice of Multi-Agent Systems*, pages 478–486, 2019.
[13] E. Nunes, M. Manner, H. Mitiche, and M. Gini. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, 90:55–70, 2017.
[14] C. Ocampo-Martinez et al. Energy efficiency in discrete-manufacturing systems: Insights, trends, and control strategies. *Journal of Manufacturing Systems*, 52:131–145, 2019.
[15] A. W. Palmer, A. J. Hill, and S. J. Scheding. Stochastic collection and replenishment (SCAR): Objective functions. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3324–3331, 2013.
[16] M. Saint-Guillain, T. Vaquero, S. Chien, J. Agrawal, and J. Abrahams. Probabilistic temporal networks with ordinary distributions: Theory, robustness and expected utility. *Journal of Artificial Intelligence Research*, 71:1091–1136, 2021.
[17] M. Shaked and J. G. Shanthikumar, editors. *Univariate Stochastic Orders*, pages 3–79. Springer New York, 2007.
[18] S. B. Stancliff, J. Dolan, and A. Trebi-Ollennu. Planning to fail: Reliability needs to be considered a priori in multirobot task allocation. In *IEEE Int. Conf. Systems, Man and Cybernetics*, pages 2362–2367, 2009.
[19] S. Zhang, Y. Chen, J. Zhang, and Y. Jia. Real-time adaptive assembly scheduling in human-multi-robot collaboration according to human capability. In *IEEE Int. Conf. Robotics & Automation*, pages 3860–3866, 2020.

APPENDIX

Definition 9 (Comonotonicity)**.** *Two random variables X and Y are said to be* ***comonotonic*** *if there exists a common underlying random variable U and two non-decreasing functions f and g such that:*

$$X = f(U) \quad \textit{and} \quad Y = g(U),$$

For comonotonicity, there are a few relevant properties, including the following.

Joint Cumulative Distribution Function: The joint cumulative distribution function of X and Y is given by:

$$F_{X,Y}(x, y) = \min(F_X(x), F_Y(y)),$$

where $F_X(x)$ and $F_Y(y)$ are the marginal cumulative distribution functions of X and Y, respectively.

Copula: The copula of comonotonic random variables corresponds to the *Fréchet–Hoeffding upper bound*:

$$C(u, v) = \min(u, v), \quad \text{for } u, v \in [0, 1].$$

Maximum Positive Dependence: Comonotonicity represents the strongest form of positive dependence:

- The rank correlation (Spearman's ρ) and linear correlation (ρ) are maximal.
- The conditional distribution of Y given $X = x$ is deterministic.

Theorem 2 (Stochastic Dominance of Maxima)**.** *Let X and Y be nonnegative random variables (not necessarily identically distributed) with continuous cumulative distribution functions (CDFs) F_X and F_Y, respectively. Consider three random vectors (X_{ind}, Y_{ind}), (X_{com}, Y_{com}), and (X_{dep}, Y_{dep}) such that:*

1) *$X_{ind} \sim X$, $Y_{ind} \sim Y$, and X_{ind} and Y_{ind} are independent.*
2) *(X_{com}, Y_{com}) is a comonotone coupling of X and Y.*
3) *(X_{dep}, Y_{dep}) is any other positively dependent coupling of X and Y with copula C_{dep} satisfying $\forall u, v \in [0, 1]$:*

$$C_{ind}(u, v) = uv \leq C_{dep}(u, v) \leq \min(u, v) = C_{com}(u, v).$$

Let us define the maxima $Z_{ind} := \max(X_{ind}, Y_{ind})$, $Z_{com} := \max(X_{com}, Y_{com})$, and $Z_{dep} := \max(X_{dep}, Y_{dep})$.
Then the following stochastic order holds:

$$Z_{com} \leq_{st} Z_{dep} \leq_{st} Z_{ind},$$

which means that for all $z \geq 0$:

$$P(Z_{ind} > z) \geq P(Z_{dep} > z) \geq P(Z_{com} > z).$$

Proof. Independent case:

$$P(Z_{\text{ind}} \leq z) = P(X_{\text{ind}} \leq z, Y_{\text{ind}} \leq z).$$

Since X_{ind} and Y_{ind} are independent:

$$F_{Z_{\text{ind}}}(z) = F_X(z) F_Y(z).$$

Comonotonic case:

$$P(Z_{\text{com}} \leq z) = P(F_X^{-1}(U) \leq z, F_Y^{-1}(U) \leq z).$$

Since $F_X^{-1}(U) \leq z \iff U \leq F_X(z)$ and similarly for Y, it follows that:

$$F_{Z_{\text{com}}}(z) = \min(F_X(z), F_Y(z)).$$

Positively Dependent case: For any positively dependent coupling:

$$F_{Z_{\text{dep}}}(z) = C_{\text{dep}}(F_X(z), F_Y(z)).$$

Given the bounds on the copula:

$$F_X(z) F_Y(z) \leq C_{\text{dep}}(F_X(z), F_Y(z)) \leq \min(F_X(z), F_Y(z)).$$

Therefore, we conclude that:

$$F_{Z_{\text{ind}}}(z) \leq F_{Z_{\text{dep}}}(z) \leq F_{Z_{\text{com}}}(z),$$

which implies:

$$P(Z_{\text{ind}} > z) \geq P(Z_{\text{dep}} > z) \geq P(Z_{\text{com}} > z), \forall z \geq 0.$$

□