Automated Planning for Task-Based Cyber-Physical Systems under Multiple Sources of Uncertainty

RAQUEL SÁNCHEZ-SALAS, JAVIER TROYA, and JAVIER CÁMARA, ITIS Software, Universidad de Málaga, Spain

In smart Cyber-Physical Systems (sCPS), a critical challenge lies in task planning under uncertainty. There is a broad body of work in the area with approaches able to individually deal with different classes of constraints (e.g., ordering, structural) and uncertainties (e.g., in sensing, actuation, latencies). However, these uncertainties are rarely independent and often compound, affecting the satisfaction of goals and other system properties in subtle and often unpredictable ways. According to the *Uncertainty Interaction Problem* recently proposed in the literature, approaches are needed to identify multiple sources of uncertainty and quantify their impact. In this paper we deal with two types of uncertainty present in task-based sCPS, namely *temporal availability constraints* and *element reliability*. The former refers to the availability of a given system element required to perform a task, which may be unavailable for certain periods of time, while the latter is related to system elements that may fail at some point with some probability. This paper presents an approach to consider both uncertainties, employing genetic algorithms to incorporate them effectively into planning for deciding how to best adapt the system to changes at run time. Our method is evaluated in the domains of electric vehicle charging and healthcare robotics. Our evaluation shows that: (i) the proposed approach outperforms a baseline mixed-integer linear programming (MILP) algorithm capable of generating optimal solutions in the absence of uncertainty, providing more robust solutions to failures, changes in temporal availability, or both sources of uncertainty combined; (ii) both sources of uncertainty have a strong and compound impact on the quality of the solutions provided; and (iii) the proposed approach significantly reduces computational cost, with respect to the MILP-based optimization.

CCS Concepts: • **Software and its engineering** → *Extra-functional properties*;

Additional Key Words and Phrases: planning, adaptation, uncertainty, availability constraints, CPS

ACM Reference Format:
Raquel Sánchez-Salas, Javier Troya, and Javier Cámara. 2024. Automated Planning for Task-Based Cyber-Physical Systems under Multiple Sources of Uncertainty. *J. ACM* 37, 4, Article 111 (August 2024), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

ACKNOWLEDGMENTS

This work was partially supported by the Spanish Government (FEDER/Ministerio de Ciencia e Innovación – Agencia Estatal de Investigación) under projects SoCUS [TED2021-130523B-I00] and IPSCA [PID2021-125527NB-I00].

Authors' address: Raquel Sánchez-Salas, raquelsalanchez@uma.es; Javier Troya, jtroya@uma.es; Javier Cámara, jcamara@uma.es, ITIS Software, Universidad de Málaga, Málaga, Spain.

ACM 0004-5411/2024/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Self-adaptive, smart Cyber-Physical Systems (sCPS) [4] are becoming increasingly prevalent in various domains, such as healthcare, transportation, and industrial automation, among others. The complexity and uncertainty inherent to sCPS pose significant challenges for their effective and efficient operation. Indeed, uncertainty is an inherent property of any system that operates in a real environment, which often includes interaction with physical elements or humans. There are some proposals for representing uncertainty in software systems at a model level [46] and to deal with it in different forms (objective, subjective, epistemic, aleatory) [45], for which different notations and representations are used, such as mathematical and numerical models [37], probabilities [18], Fuzzy set theory [41, 55], variability analysis [43] or risk assessment [40]. Also, several taxonomies of uncertainty have been proposed in the context of self-adaptive systems [19, 30, 38, 39], and a substantial body of work exists on methods to manage uncertainty [22]. These methods are able to individually represent and mitigate uncertainties from various sources. However, these uncertainties are rarely independent and often compound, affecting the satisfaction of goals and other system properties in subtle and often unpredictable ways.

To address this gap, the *Uncertainty Interaction Problem* was introduced in [6, 11]. It highlights the limited understanding about the specific ways in which uncertainties of different types and from various sources are combined and, ultimately, affect the goals and other properties of CPS. Representing the combination of multiple sources of unceratinty, analyzing the impact of their emergent effects on the satisfaction of requirements, and mitigating them remain open challenges. In our approach, we do not treat these sources of uncertainty independently. Instead, our method incorporates them in a combined manner within the planning process, ensuring that scheduling decisions take into account both sources of uncertainty.

Attaining satisfaction of system requirements in the presence of uncertainty is a challenge for which autonomy and self-adaptation have been proved crucial over the last decade [51]. In particular, the MAPE-K (Monitoring, Analysis, Planning, Execution and Knowledge) control loop has been identified as an extraordinarily effective model for realizing self-adaptation in sCPS [3].

In fact, recent years have witnessed a remarkable growth of the body of work for automated approaches to planning that can enable sCPS to make informed decisions and adapt their behavior in an effective manner under prescribed levels of uncertainty, e.g. [5, 10, 16, 33]. Among these proposals, we contributed in previous work [42], an approach to synthesize self-adaptation plans for sCPS under uncertainty in temporal availability constraints. Indeed, many real-world scenarios involve elements of the system that are available only during specific time periods. For instance, a robot may need to conduct a task at a physical location that is not always accessible, or an electrical vehicle may be available for charging at a station only during a limited time window. These temporal constraints introduce an additional layer of complexity to the planning and decision-making process. Existing methods often struggled to effectively incorporate this type of uncertainty, so we contributed with a novel approach to automated planning that explicitly addresses the uncertainty in temporal availability constraints in task-based sCPS scenarios. We explored the use of genetic algorithms [27] to efficiently consider the space of possible plans to identify solutions that are not only feasible given the uncertain time constraints, but also close-to-optimal with respect to a set of quantitative objectives of the sCPS (e.g., cost, time).

Despite their effectiveness, a notable limitation of these approaches is the ignorance of multiple sources of uncertainty. In fact, one of the challenges described in [11], namely *Challenge E1: Complementary uncertainty exploration*, calls for the development of complementary uncertainty exploration techniques that consider multiple sources of uncertainty with varying types and degrees of impact. In this paper we contribute towards addressing this challenge by considering,

together with the uncertainty associated with temporal availability constraints, another source of uncertainty, namely *element reliability* (i.e., CPS elements may fail at some point with some probability). For instance, a robot or electric vehicle chargers might be down for some time intervals due to different failures, which inevitably affects the attainment of system goals. Thus, we study how the interaction of both uncertainties (i.e., in temporal availability constraints and reliability) affects system goals. To this end, we extend the formalization of the problem, algorithms, and implementation described in [42], so that it can consider both sources of uncertainty.

To evaluate our approach, we apply it on two case studies that correspond to distinct domains, namely a shared smart infrastructure for the charging of electric vehicles, and healthcare robotics. For studying how the various uncertainties affect the system, we compare our implementation with a baseline method to solve the planning problems that employs mixed-integer linear programming (MILP) so that it guarantees a globally optimal solution in the absence of uncertainty.

We focus on two properties to evaluate our approach, namely effectiveness and efficiency, for which we define three research questions (RQs):

- **RQ1: Effectiveness**. We aim to study how the different uncertainties, and their combination, affect the systems. We divide this RQ into four subquestions:
  - **RQ1.1** How effective is our approach in comparison to an optimal algorithm with no uncertainties considered?
  - **RQ1.2** How effective is our approach in comparison to an optimal algorithm under prescribed levels of uncertainty in temporal availability constraints?
  - **RQ1.3** How effective is our approach in comparison to an optimal algorithm under prescribed levels of uncertainty in elements reliability?
  - **RQ1.4** How effective is our approach in comparison to an optimal algorithm under prescribed levels of uncertainty in temporal availability constraints and elements reliability?
- **RQ2: Impact**. How much do the different sources of uncertainty impact of system goals?
- **RQ3: Efficiency**. How efficient is our approach with respect to an optimal algorithm?

Our results demonstrate that, while the baseline approach fares better than our proposal under no uncertainty, our proposed approach yields solutions that outperform the baseline under increasing levels of uncertainty, rendering the system more resilient. This applies both to uncertainty in temporal availability constraints, as well as to reliability. Moreover, the computational cost of our approach is a fraction of the computational cost of the baseline, with time reductions above 70% for large problem instances. Finally, our study indicates that the different sources of uncertainty have a compound effect on system objectives, exhibiting an *augmentation behavior* (uncertainty sources are strongly correlated but neither dominates the other) [6].

In the remainder of this paper, Section 2 introduces the motivating scenarios for our approach and Section 3 formalizes the problem. Next, Section 4 introduces our approach, centered on the planning stage of the MAPE-K loop, and Section 5 describes its evaluation, including threats to validity. Section 6 discusses related work. Finally, Section 7 concludes the paper.

2 MOTIVATING SCENARIOS

We describe two motivating scenarios where multiple sources of uncertainty must be considered simultaneously. These examples demonstrate how different types of uncertainty, such as temporal availability constraints and element reliability, affect decision-making and increase the complexity of planning

2.1 Electric Vehicle Charging Infrastructure

Most electric vehicles (EVs) available in the market offer limited autonomy. For this reason, many EV owners need to install home chargers, yet limited access to charging stations in shared parking areas (e.g., at the workplace) still poses significant obstacles to electromobility. Shared EV charging infrastructures can help to alleviate this problem, but are difficult to exploit in an efficient manner due to factors such as limited vehicle availability (i.e., a vehicle may be available for charging only during a limited time window), resource failures (e.g., a charger may fail from time to time), and diverse energy needs (some users may need to cover longer distances than others to return home). To address this situation and promote EV adoption, there is a pressing need for shared charging stations capable of automatically planning the order and duration of the charging of multiple EVs under uncertain and changing conditions.

To motivate our approach, we describe a scenario involving a parking lot at the Ada Byron research building in the University of Malaga, where users park a variety of EVs with different energy needs (Figure 1). Some of them only need to cover a few kilometers to commute back home from work, while others need to cover longer distances.

Fig. 1. Electric vehicle charging infrastructure at the Ada Byron research building, University of Malaga.

The ability to adapt and optimize charging according to individual user requirements and daily schedule (available from a database updated periodically via an app installed in the user's mobile phone) is crucial to ensure that everyone can use their electric vehicles conveniently and efficiently, without concerns about charger availability. In this setting, the need for a robust and adaptive vehicle charging system becomes even more evident when considering the unpredictability of real-life scenarios, where the system may have to deal with chargers unexpectedly going offline, vehicles arriving earlier or later than specified in the user's mobile app, and fluctuating energy requirements due to unexpected travel needs. Concretely, the planning system should generate feasible solutions (i.e., solutions that respect the constraints on EV availability and satisfy the charging needs of all cars) while mimimizing the cost and timespan required to charge all cars.

To summarize, this scenario involves two key sources of uncertainty: (i) the temporal availability of electric vehicles, which may arrive at charging stations earlier or later than expected, and (ii) the failure probability of chargers, which may become unavailable due to technical issues. These uncertainties should not be treated as independent; for example, it is necessary to take into account in the planning process that, if a vehicle arrives late, the charger assigned at that point may fail with a probability that is different from the failure probability at the intended arrival time of the vehicle.

2.2 RoboMAX: Food Logistics

RoboMAX [2] is a repository of robotic mission adaptation exemplars co-designed by robotic application researchers, developers, operators, and end-users. The repository captures key sources of uncertainty, adaptation concerns, and other distinguishing characteristics that pose multiple adaptation challenges.

Food Logistics is one of such mission adaptation exemplars in which the system should deliver food from the kitchen to an inpatient room. The delivery is made on an order-by-order basis in response to kitchen delivery requests. The food can be delivered into a room table by the robot (requires a special manipulation robot skill) or the food can be fetched from the robot's tray. Fetching

from the robot tray requires cooperation with the inpatient, a companion, a nurse, or another robot. Some inpatients could be unable to fetch the food from the tray, and a companion visitor or nurse should be available at the moment of the delivery. The information about if a companion visitor or nurse is able to retrieve the food from the tray during a given time period can be obtained based on the inpatient record according to information about the patient and the presence of a companion into the room. This information is subject to uncertainty, since companions could arrive or leave inpatient rooms at different times from the ones stated in the inpatient record. Moreover, robots may fail and become inoperative from time to time. In this scenario, robots should minimize the disruption to inpatients by trying to avoid food delivery attempts when there is not a companion or nurse available to receive the food, as well as the timespan required to deliver all food orders.

Summarizing, in this scenario, food delivery is subject to two sources of uncertainty: (i) the availability of companions to assist patients in retrieving their meals, and (ii) the failure probability of robots, which may become non-operational at any time. These uncertainties are not independent and complicate the planning process. For instance, it is essential to take into consideration that if a robot fails and arrives late to deliver a meal, a companion could be unexpectedly absent because she had to leave early. This presence of multiple uncertainties requires a robust planning mechanism that can handle the emergent effects of such disruptions effectively.

3 PROBLEM FORMALIZATION

A set $R = \{r_1, \ldots, r_n\}$ of resources (e.g., robots, chargers) is deployed in a physical environment and available to perform a set of tasks $T = \{t_0, \ldots, t_m\}$ (e.g., charging, delivering food) for a set of consumers $C = \{c_1, \ldots, c_l\}$ (e.g., EVs, patients).

However, tasks cannot be performed at all times, given that they have to be carried out in a specific physical location and require the availability of resources and consumers, which may not always be present. For instance, electric chargers and robots can sometimes be offline, vehicles are only available during a given time window, and in a hospital setting, inpatient companions may not always be present to receive the food delivered by a robot.

Definition 1 (Availability window). *We define the availability window of a CPS element $e \in E = \{R \cup C\}$ as $W : E \rightarrow \mathbb{R}_0^+ \times \mathbb{R}^+$, that is, the time period during which the element is available. For any given element $e \in E$, with $W(e) = (t_e^\alpha, t_e^\omega)$, we refer to t_e^α and t_e^ω as the* availability start time *and* availability end time *of e, respectively. Note that our problem formalization considers a single availability window per CPS element for presentation clarity, but it can be trivially extended to consider multiple windows.*

The definition of availability window is described in deterministic terms. In the real world, however, there is limited certainty about the actual availability window, so we also characterize the availability of CPS elements in probabilistic terms.

Definition 2 (Availability probability). *Let $f_{t_e^\star}(t)$ be the probability density function (PDF) of the availability start/end of an element e, where $\star \in \{\alpha, \omega\}$:*

$$f_{t_e^\star}(t) = \frac{1}{\sigma_{t_e^\star}\sqrt{2\pi}} \exp\left(-\frac{(t - \mu_{t_e^\star})^2}{2\sigma_{t_e^\star}^2}\right),$$

where $\mu_{t_e^\star}$ is the mean and $\sigma_{t_e^\star}$ is the standard deviation of the availability start/end (Gaussian) distributions. The corresponding cumulative distribution functions (CDFs) are designated by:

$$F_{t_e^\star}(t) = \int_{-\infty}^{t} f_{t_e^\star}(x)\, dx.$$

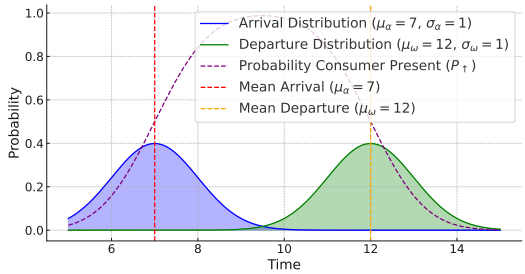

Fig. 2. Example of probability availability function for an electric vehicle.

The probability that the CPS element is present on location at time t, denoted as $P_{\uparrow e}(t)$, is then given by the difference between the availability start (α) and end (ω) CDFs:

$$P_{\uparrow e}(t) = F_{t_e^\alpha}(t) - F_{t_e^\omega}(t).$$

Example 1. *Figure 2 illustrates a sample timeline for the arrival-departure of an electric vehicle e. The blue and green curves depict the probability density functions for the arrival and departure of the vehicle (with means in $\mu_\alpha = 7$ and $\mu_\omega = 12$, respectively). The dashed curve corresponds to the probability of the vehicle being present over the timeline $P_{\uparrow e}(t)$. Note that the function peaks when the arrival (availability start) CDF is maximal, but the departure (availability end) CDF is still close to zero.*

Definition 3 (Problem instance). *A problem instance is characterized by a tuple $(R, C, F_{\uparrow E}, F_X, T, \Lambda, \Theta, O)$, where R is a set of resources, C is a set of consumers, $F_{\uparrow E}$ is a set of consumer availability probability functions for CPS elements in E, $F_X : R \times \mathbb{R}^2 \rightarrow [0, 1]$ is a function that maps time intervals to failure probabilities of resources, T is a set of tasks, $\Lambda : T \rightarrow C$ is a function that maps tasks to consumers, $\Theta \in \mathbb{R}^+$ designates a time frame $[0, \Theta]$ during which tasks have to be completed, and $O : S_v \rightarrow \mathbb{R}^k$ is a function that maps solutions (cf. Definition 5) of a problem instance v to k optimization criteria.*

Definition 4 (Resource assignment). *A resource assignment is characterized as a tuple $(r, t, \theta_s, \theta_e) \in R \times T \times \mathbb{R}^2$ that allocates a resource r to the execution of a task t, which is to be carried out during the time period $[\theta_s, \theta_f]$.*

Definition 5 (Problem solution). *A problem solution is a set of* resource assignments *A in which there is non-zero probability of the task resource and consumer being present ($\forall a \in A \bullet \exists \theta \in [a.\theta_s, a.\theta_f],\ s.t.\ P_{\uparrow \Lambda(a.t)}(\theta) > 0 \wedge P_{\uparrow a.r}(\theta) > 0$). We designate the set of possible problem solutions for problem instance v by S_v.*

Definition 6 (Problem). *Let i be a problem instance, and $Y = \{y \in \mathbb{R}^k \ : \ y = i.O(s),\ s \in S_v\}$ be the set of criterion vectors in $\mathbb{R}^k$ that correspond to the mapping of solutions to optimization criteria $i.O$. The planning problem consists in approximating the Pareto front of non-dominated solutions $S_v^\star$:*

$$S_v^\star = \{s \in S_v : i.O(s) \in Pareto(Y)\},\ with Pareto(Y) = \{y' \in Y : \{y'' \in Y : y'' \succ y',\ y' \neq y''\} = \emptyset\}.$$

4 APPROACH

In this section, we first introduce an overview of the run-time adaptation loop that frames the operational context for our approach (Section 4.1). Next, we present a planning approach that does not consider uncertainty sources, and is based on mixed-integer linear programming (Section 4.2). This approach yields optimal results in the absence of uncertainty and is used as baseline for our

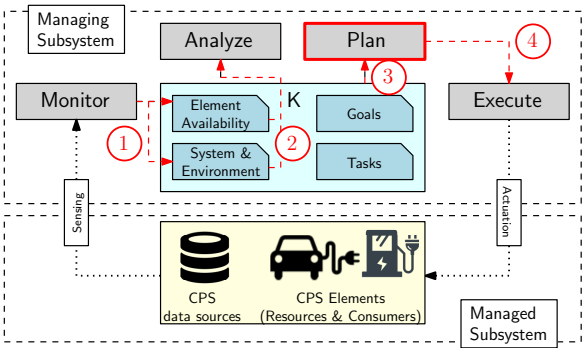

Fig. 3. Run-time context overview of our approach.

evaluation (cf. Section 5). Finally, we describe the details of our planning approach under interacting sources of uncertainty (Section 4.3).

4.1 Operational Context

Our approach is designed to work at run-time in the planning stage of the adaptation loop of MAPE-K based systems. Figure 3 shows the operational context of our approach.

4.1.1 Knowledge. Incorporates various models that contain information about the tasks to be performed, system goals (e.g., optimization objectives – cf. O in Definition 6), resource and consumer availability – cf. Definition 1), as well as system and environment information. We assume that all these models are continually updated at run-time (cf. monitoring and analysis), except for the one capturing system goals, which we assume to be fixed.

4.1.2 Monitor. The CPS and its environment, including elements such as consumers (e.g., EVs, people) and resources (e.g., chargers, robots) are monitored on an ongoing basis through sensors. Updated information is placed ① in the system and environment model. Moreover, information about CPS element availability that resides in CPS data sources is also ① continually monitored and incorporated into the element availability model. Relevant data sources to extract element availability information are for instance, the database updated through the mobile app in the EV charging scenario, and the inpatient record database in the RoboMax food logistics scenario.

4.1.3 Analyze. During analysis, the system ② checks whether there are any deviations between the information about expected element availability in the knowledge base and information monitored from the system and environment at run-time (e.g., whether a given EV that should be on location for charging is not actually available). If such deviations exist, the planning stage is triggered for replanning.

4.1.4 Plan. If planning is triggered, a new planning problem instance is created ③. The process to create the instance uses information from all four models and is built similarly to the previous problem instance (i.e., following Definition 3), but removing tasks, resources, and consumers that are scheduled to participate in a task and unavailable, or that are already participating in a task. The details of planning, which is the focus of our approach, are described in Section 4.3.

4.1.5 Execution. Once a plan is generated ④ (cf. Definition 5), it is enacted through effectors at the managed subsystem level.

In the remainder of this section, we first describe in detail an algorithm based on mixed-integer linear programming (MILP) that we have developed to solve our problem when temporal availability constraints are known with certainty (Section 4.2), and follow with a description of our genetic algorithm to solve the problem under uncertainty in temporal availability constraints (Section 4.3) and element reliability is guaranteed.

4.2 Planning in the Absence of Uncertainty

This section introduces an algorithm that solves planning with well-defined temporal availability constraints (cf. Definition 1) and no element failures (cf. Definition 3) in an optimal fashion.

Algorithm 1 provides as output a problem solution as a set of resource assignments according to Definition 5, and receives as input a deterministic problem instance, which is equivalent to the one described in Definition 3, but including deterministic availability windows of resources and consumers, instead of their availability probability functions:

Definition 7 (Deterministic problem instance). *A deterministic problem instance is a tuple* $(R, C, W, T, \Lambda, \Theta, O)$*, where* R *is a set of resources,* C *is a set of consumers,* W *maps CPS elements in* E *to their availability windows,* T *is a set of tasks,* $\Lambda : T \rightarrow C$ *is a function that maps tasks to consumers,* $\Theta \in \mathbb{R}^+$ *designates a time frame* $[0, \Theta]$ *during which tasks have to be completed, and* $O : S_v \rightarrow \mathbb{R}^k$ *is a function that maps solutions of a problem instance* v *to* k *optimization criteria.*

To solve the linear programming problem, we divide the timeline into a discrete set of time slots according to a time granularity parameter t_g. For instance, if we consider the timeline to be one day and set $t_g = 1$ *hour*, we have 24 slots that we can refer to by their indexes (1..24).

Definition 8 (Linear programming problem instance). *A linear programming problem instance* (X, O, ρ) *is characterized by a set of binary decision variables* $X : R \times C \times \mathbb{N} \leftarrow \{0, 1\}$ *that specify whether a resource* $r \in R$ *is assigned to a consumer* $c \in C$ *during a given time slot* $i \in \mathbb{N}$*, a function* $O : S_v \rightarrow \mathbb{R}^k$ *that maps solutions of a problem instance* v *to* k *optimization criteria, and a set of constraints* ρ *that apply to valid problem solutions.*

Based on the deterministic problem instance i received as input, the algorithm progressively builds a linear programming problem instance γ (initialized in line 3), that incorporates a set of binary decision variables to express the assignment of a consumer to a resource during a given time slot (initialized in line 4, cf. Definition 8), and the set of optimization objectives from the deterministic problem instance $i.O$. Initially, the set of constraints $\gamma.\rho$ is empty (line 3), and incorporates a constraint capturing that no two assignments can involve the same resource during the same time slot (line 5). Then, the set of constraints is further extended to incorporate constraints that capture that for every resource, consumer, and timeslot, the fact that a consumer and a resource are allocated together to a given timeslot implies that the timeslot should be within the availability windows of both the resource and the consumer (line 9). In line 9, function $slots : W \rightarrow \mathbb{P}(\mathbb{N})$ returns the set of slots that fall within the availability window w of a resource or consumer:

$$slots(w) = \{n \in \{1, \ldots, \lceil \frac{\Theta}{t_g} \rceil\} \mid (n-1) \cdot t_g \geq w.t^{\alpha} \wedge n \cdot t_g \leq w.t^{\omega}\}$$

The algorithm then generates the solution to the linear programming problem instance γ (line 13), where function *solve* abstracts the underlying algorithm used to return MILP problem solutions (for which different solvers can be used, cf. Section 5.1).

The last part of the algorithm (lines 15-22) maps decision variables in $\gamma.X$ to a problem solution in terms of resource assignments (cf. definitions 4 and 5). To achieve this, an initially empty set of assignments A is created (line 14), and then progressively extended by iterating over all the

Algorithm 1: Uncertainty-Agnostic Planning (Mixed-Integer Linear Programming).

1 **Input:** Deterministic problem instance $i = (R, C, W, T, \Lambda, \Theta, O)$ (cf. Definition 7).
2 **Output:** A problem solution assigning task to resources and consumers (cf. Definition5).
3 $\gamma \leftarrow (X = \emptyset, O = i.O, \rho = \emptyset)$
4 $\gamma.X \leftarrow \{(c, r, 0) \mapsto 0 \mid c \in i.C, r \in i.R\}$
5 $\gamma.\rho \leftarrow \{\forall x, x' \in \gamma.X \bullet x.r = x'.r \Leftrightarrow (x = x' \vee x.s \neq x'.s)\}$
6 **foreach** $r \in i.R$ **do**
7 **foreach** $c \in i.C$ **do**
8 **foreach** $s \in \{1, \ldots, \lceil \frac{\Theta}{t_g} \rceil\}$ **do**
9 $\gamma.\rho \leftarrow \gamma.\rho \cup \{\gamma.X(c, r, s) = 1 \Rightarrow s \in slots(i.W(r)) \cap slots(i.W(c))\}$
10 **end**
11 **end**
12 **end**
13 $solve(\gamma)$
14 $A \leftarrow \emptyset$
15 **foreach** $x \in \gamma.X$ **do**
16 **if** $\nexists\ a \in A \bullet a.r = x.r \wedge a.t = \Lambda^{-1}(x.c)$ **then**
17 $A \leftarrow A \cup \{(x.r, \Lambda^{-1}(x.c), [x.s \cdot t_g, (x.s + 1) \cdot t_g])\}$
18 **end**
19 **else**
20 $a.\theta_s \leftarrow min(a.\theta_s, x.s \cdot t_g)$
21 $a.\theta_f \leftarrow max(a.\theta_f, x.s \cdot t_g)$
22 **end**
23 **end**
24 **return** A

decision variables x (line 15). For each iteration, if there is not an assignment a in A that coincides in resource and consumer with those of x (line 16), we initialize an assignment with a time interval that coincides with the upper and lower limits of the time slot for x (line 17). Otherwise (i.e., there is an assignment in A that coincides with x in resource and consumer), we update the time values of the assignment to incorporate the new time slot within the assignment's time window to expand it further (lines 20-22). The algorithm guarantees that correct solutions with no gaps between slot assignments will be returned (line 24), given that we are assuming a single availability window for elements in the CPS, as discussed in Section 3.

4.3 Planning under Uncertainty in Temporal Availability Constraints and Element Reliability

Once we have presented how to solve a simplified version of our problem in which temporal availability constraints are well known and element reliability is guaranteed, in this section we describe a genetic algorithm capable of providing solutions under uncertainty arising from: (i) temporal availability constraints, (ii) reliability issues derived from resource failures, and (iii) a combination of the two previous sources of uncertainty.

4.3.1 Main Planning Algorithm. Algorithm 2 provides as output a problem solution as a set of resource assignments according to Definition 5, and receives as input a problem instance (cf. Definition 3), as well as a set of parameters to control the behavior of the genetic algorithm that includes: population size s_{pop}, the maximum number of generations g_{max}, the parent selection and mutation rates r_{sel} and r_{mut}, as well as the number of samples to generate for the evaluation of fitness of individuals (κ for the number of variants with respect to consumer-resource availability,

and κ_X for the number of variants related to resource failures), which are discussed in more detail later in this section.

The algorithm starts by initializing a population P (initially empty, line 4), and populating it with individuals that are formed as a set of assignments A of the form $(r, t, \theta_s, \theta_f)$ for every task in the problem instance (i.e., in $i.T$), where resource r is randomly picked (line 8), and the time bounds for the assignment are also randomly picked (through function $rnd_w : R \times C \rightarrow \mathbb{R}_{\geq 0}$) among values that respect the temporal availability constraints of resource r and the consumer associated with task t (i.e., $i.\Lambda^{-1}(t)$). Concretely, for a pair resource-consumer (r, c), function rnd_w generates a random value in the interval $[min(f^{\alpha}_{t^{\alpha}_r}.\mu, f^{\alpha}_{t^{\alpha}_c}.\mu), max(f^{\omega}_{t^{\omega}_r}.\mu, f^{\omega}_{t^{\omega}_c}.\mu)]$. Note that the invocation to rnd_w to obtain a value for θ_f (line 10) is constrained to yield values higher than that of θ_s (line 9).

Algorithm 2: Planning under uncertainty (Genetic Algorithm).

1 **Inputs:** Problem instance $i = (R, C, F_{\uparrow E}, T, \Lambda, \Theta, O)$ (cf. Definition 3); Population size s_{pop}; Maximum number of generations g_{max}; Parent selection rate r_{sel}; Mutation rate r_{mut}; Number of samples for probabilistic evaluation of resource-consumer availability κ; Number of samples for probabilistic evaluation of resource failures κ_X.
2 **Output:** A problem solution assigning task to resources and consumers (cf. Definition5).
3 **GA**$(i, s_{pop}, g_{max}, r_{sel}, r_{mut}, \kappa)$:
4 $P \leftarrow \emptyset$
5 **foreach** $ind \in \{1, \ldots, s_{pop}\}$ **do**
6 $A \leftarrow \emptyset$
7 **foreach** $t \in i.T$ **do**
8 $r \leftarrow random(i.R)$
9 $\theta_s \leftarrow rnd_w(i.F_{\uparrow}(r), i.F_{\uparrow}(i.\Lambda^{-1}(t))$
10 $\theta_f \leftarrow rnd_w(i.F_{\uparrow}(r), i.F_{\uparrow}(i.\Lambda^{-1}(t)),\ s.t.\ \theta_f > \theta_s$
11 $A \leftarrow A \cup \{(r, t, \theta_s, \theta_f)\}$
12 **end**
13 $P \leftarrow P \cup \{A\}$
14 **end**
15 $RFS \leftarrow generateRFS(i, \kappa_X)$
16 **foreach** $gen \in \{1, \ldots, g_{max}\}$ **do**
17 $FP \leftarrow \{p \mapsto \langle 0 \rangle^k \mid p \in P\}$
18 **foreach** $p \in P$ **do**
19 $FP_{\kappa} \leftarrow \emptyset$
20 **for** $samp \in \{1, \ldots, \kappa\}$ **do**
21 $p_{\kappa} \leftarrow generate_{\kappa}(p)$
22 $p'_{\kappa} \leftarrow consolidate(p_{\kappa}, select(RFS))$
23 $FP_{\kappa} \leftarrow FP_{\kappa} \cup \{p'_{\kappa} \mapsto i.O(p'_{\kappa})\}$
24 **end**
25 $FP[p] \leftarrow mean(FP_{\kappa})$
26 **end**
27 $P_{sel} \leftarrow selection(FP)$
28 $P \leftarrow crossover(P_{sel})$
29 $P \leftarrow mutation(P)$
30 **end**
31 **return** $\{p \in P : i.O(p) \in Pareto(FP)\}$

Once the initial population has been built, the algorithm generates a set of *resource failure schedules* (function *generateRFS*, line 15), each of which captures a potential scenario of resource failures throughout the execution timeline.

Definition 9 (Resource Failure Schedule). *Given a problem instance* $i = (R, C, F_{\uparrow E}, F_X, T, \Lambda, \Theta, O)$*, a resource failure schedule* $RFS \subseteq i.R \times [\mathbb{R}_{\geq 0}, \mathbb{R}_{\geq 0}]^{\lceil \frac{i.\Theta}{t_g} \rceil}$ *is a set of assignments of resources to sets of time slots in which the resource is assumed to be in a failed state (and therefore not usable).*

This set of time slots in which resources are in a failed state will be used in the probabilistic evaluation of the fitness function, within the main loop of the algorithm, which is described in the remainder of this section. Section 4.3.2 provides the details of resource failure schedule generation.

The main loop of the algorithm iterates over a maximum number of generations g_{max} (lines 16-29) by evaluating the current population of assignments (lines 17-26), and then selecting the parents (line 27) to generate the offspring for the next generation (line 28 crossover, and line 29 mutation). We provide details about these stages of the algorithm in the following.

Concerning evaluation of individuals in the population (lines 17-29), the process begins by creating a dictionary FP that associates every individual in the population $p \in P$ with a zero-vector of k real numbers to store the fitness of p along the k optimization objectives of the problem instance ($i.O$). Then, for every individual in the population, we generate κ samples that capture variants of p in which the time period for each assignment is generated according to the probability density functions of the start (f_{t^α}) and end (f_{t^ω}) of the availability for a given resource or consumer. Hence, if we let $A \subseteq R \times T \times \mathbb{R}^2$ be the set of possible assignments in a solution, function $generate_\kappa : \mathbb{P}(A) \rightarrow \mathbb{P}(A)$ takes in a set of assignments and returns a different set of assignments, where each one of them is obtained by leaving the resource and task assigned unchanged, and modifying the timeframes by sampling according to the maximum variance of the start and end of the availability of the resource and consumer involved in the assignment:

$$gen_\kappa(r, t, \theta_s, \theta_f) = (r, t, spl(\theta.s, \sigma^2_\alpha), spl(\theta_f, \sigma^2_\omega)), \text{with } \sigma^2_\star = max(f_{t^\star_r}.\sigma^2, f_{t^\star_{i.\Lambda^{-1}(t)}}.\sigma^2), \star \in \{\alpha, \omega\}$$

In the expression above, function gen_κ returns an assignment in which the start and end times (θ_s and θ_f, respectively) are generated based on a sample from a Gaussian distribution (function spl, which takes in a mean and a variance as arguments).

At this stage, the κ samples have been generated without any awareness about potential resource failures. Hence, we further modify each of the generated samples to avoid including consumer-resource assignments that overlap with failure periods of resources as specified in RFS (function *consolidate*, line 22). Each sample is modified according to one of the resource failure schedules contained in RFS, which is randomly drawn by function *select*. We provide further details about the definition of function *consolidate* in Section 4.3.3.

The algorithm has been modified to handle two types of uncertainty in a structured manner. During the evaluation and selection phases, both uncertainty factors are considered when generating and evaluating solutions. Unlike the previous version of our work [42], where uncertainty was handled in isolation, the new approach systematically mitigates both types by incorporating them into the fitness evaluation and adaptation process. This is achieved through the **generateRFS** function (cf. Algorithm 3), which accounts for resource failures, and the **consolidate** function (cf. Algorithm 4), which ensures feasible assignments by adapting to these failures.

While the algorithm handles uncertainty sources in a combined manner, this does not imply independence. The probabilistic evaluation of task assignments (lines 21-22 of the algorithm) ensures that both uncertainty sources influence scheduling decisions, implicitly capturing their interaction within the planning process.

When each of the κ samples is generated for each individual p, its fitness along the k optimization objectives is stored in FP_κ (line 23). Once all the samples are generated and evaluated, we assign as the fitness of the individual p, the mean of the fitness of all samples (individually, for each of the k optimization objectives – function *mean*, line 25).

Once all the individuals in the population have been evaluated, the algorithm selects the r_{sel} top-most individuals in lexicographical order (function *selection*, line 27). Then, the algorithm generates the next population by applying the $crossover : A \times A \rightarrow A \times A$ operator (line 28), which combines individuals by taking the start time of one parent, and the end time of another one:

$$crossover((r_1, t_1, \theta_{s1}, \theta_{f1}), (r_2, t_2, \theta_{s2}, \theta_{f2})) = ((r_1, t_1, \theta_{s2}, \theta_{f1}), (r_2, t_2, \theta_{s1}, \theta_{f2}))$$

For $mutation : A \rightarrow A$ (line 29), we apply a change to the start or end time of the assignment, by incrementing (or decrementing) it by an amount of time randomly generated, so that the result always has to satisfy the constraint that the start and end time of the assignment cannot go beyond the mean of the probability density function of the resource and consumer availability, i.e.:

$$\theta'_s \geq max(f_{t_r^\alpha}.\mu, f_{t^\alpha_{i.\Lambda^{-1}(t)}}.\mu) \wedge \theta'_f \leq min(f_{t_r^\omega}.\mu, f_{t^\omega_{i.\Lambda^{-1}(t)}}.\mu).$$

Finally, the algorithm finishes by returning the set of non-dominated solutions in the population (line 31 – cf. Definition 6).

4.3.2 Generation of Resource Failure Schedules. Algorithm 3 creates a set of failure schedules for resources in the system. It receives as input a problem instance, a number of failure schedule samples to generate κ_X, and a time granularity parameter t_g (cf. Definition 7). The algorithm starts by creating an empty set of schedules *RFS* (line 3), to which it adds each of the failure schedules *RF* generated (lines 5-14). Each failure schedule is generated by iterating over the different time slots starting at each time instant θ (line 7) in the timeline $[0, i.\Theta]$, and determining whether for that slot, there exists a failure probability defined in $i.F_X$ (line 8). If that is indeed the case, then the algorithm generates a boolean value that determines whether the resource r is in a failure state during the current time slot (using function $sfr : [0, 1] \rightarrow \mathbb{B}$, which employs the failure probability of r in $i.F_X$). If the value returned by sfr is true, then the slot is added to *RF* as one in which the resource r has failed.

Algorithm 3: *generateRFS*: Resource Failure Schedule Generation.

1 **Inputs:** Problem instance $i = (R, C, F_{\uparrow E}, F_X, T, \Lambda, \Theta, O)$ (cf. Definition 3); Number of samples for probabilistic evaluation κ_X; Time granularity parameter t_g.
2 **Output:** A list of failure times for resources in $i.R$, where each element is defined over $R \times [\mathbb{R}_{\geq 0}, \mathbb{R}_{\geq 0}]^{\lceil \frac{i.\Theta}{t_g} \rceil}$.
3 $RFS \leftarrow \emptyset$
4 **foreach** $\{1, \ldots, \kappa_X\}$ **do**
5 $\quad RF \leftarrow \{r \mapsto \langle\rangle^{\lceil \frac{i.\Theta}{t_g} \rceil} \mid r \in i.R\}$
6 $\quad$ **foreach** $r \in i.R$ **do**
7 $\quad\quad$ **foreach** $\theta \in \{0, t_g \ldots, i.\Theta - t_g\}$ **do**
8 $\quad\quad\quad$ **if** $\exists f_X \in i.F_X \bullet f_X.\theta_s \leq \theta \leq f_X.\theta_f \wedge f_X.r = r$ **then**
9 $\quad\quad\quad\quad$ **if** $sfr(f_X(r, [\theta, \theta + t_g]))$ **then**
10 $\quad\quad\quad\quad\quad RF[r] \leftarrow RF[r] \cup \{[\theta, \theta + t_g]\}$
11 $\quad\quad\quad\quad$ **end**
12 $\quad\quad\quad$ **end**
13 $\quad\quad$ **end**
14 $\quad$ **end**
15 $\quad RFS \leftarrow RFS \cup \{RF\}$
16 **end**
17 **return** *RFS*

Algorithm 4: *consolidate* : Consolidation of resource-consumer assignments.

1 **Inputs:** Resource failure schedule RFS, set of resource-consumer assignments A of the form $(r, t, \theta_s, \theta_f)$ (cf. Algorithm 2).
2 **Output:** A list of modified resource-consumer assignments A', s.t. $\forall a' \in A', a'.r \notin dom(RFS)$.
3 $A' \leftarrow \emptyset$
4 $R_{AV} = \{r \in R \mid r \in \bigcup_{a \in A} a.r\}$
5 **foreach** $a \in A$ **do**
6 **foreach** $rfs \in RFS$ **do**
7 **foreach** $\theta \in rfs[a.r]$ **do**
8 **if** $a.\theta_s \leq \theta \leq \theta_f$ **then**
9 $a.r \leftarrow select(R_{AV})$
10 $R_{AV} \leftarrow R_{AV} \backslash \{a.r\}$
11 **end**
12 **end**
13 **end**
14 $A' \leftarrow A' \cup \{a\}$
15 **end**
16 **return** A'

4.3.3 Consolidation of Resource-consumer Assignments. Algorithm 4 transforms a set of resource-consumer assignments A that may contain assignments including resources prone to fail (as specified in a resource failure schedule RFS), into a new set of assignments A' that does not contain any resources in RFS. To that end, it starts by determining the set of available resources R_{AV} (those not included in any of the assignments of A, line 4). Next, for each of the assignments in A, it checks whether any of the entries in the RFS contains a time slot in which the resource is supposed to fail that overlaps with the time of the assignment (lines 7-8). If that is indeed the case, the algorithm randomly reassigns (function *select*, line 9) the consumer in the assignment to a new resource among the available ones (also removing it from the available resource set, lines 9-10).

5 EVALUATION

We evaluate our approach with the objective of: **Assessing its *effectiveness* (RQ1)** under four different types of scenario that include: (RQ1.1) no uncertainty (well-known temporal availability constraints and no resource failures), (RQ1.2) uncertainty in temporal availability constraints, (RQ1.3) resource failures, and (RQ1.4) a combination of prescribed levels of uncertainty in temporal availability constraints and resource failures; **Determining *impact* of the different sources of uncertainty on the satisfaction of system goals (RQ2)**; and **Assessing its *efficiency* (RQ3)**.

In our evaluation, we define effectiveness as the system's ability to generate plans that successfully achieve the desired goals under uncertainty. The specific metrics for effectiveness vary by case study. In the EV charging infrastructure problem, the two objectives are minimizing both the timespan required to charge all EVs, as well as the overall economic cost of charging, which may fluctuate, depending on the time of the day. In the RoboMax food logistics problem, the two objectives involve minimizing the timespan required to deliver all food trays, as well as the number of disruptive events in which the robot attempts to deliver the food to an inpatient room while a companion is not present to receive it.

We define efficiency as the computational cost required to compute a valid plan. This is measured as the total execution time of the planning algorithm for a given problem instance.

5.1 Experimental Setup

To obtain the data for our evaluation, we performed experiments that compare our GA-based solution (cf. Section 4.3) with an implementation of the MILP-based solution described in Section 4.2, which we employ as baseline. The experiments were carried out on an ASUS ExpertBook laptop equipped with an Intel Core i7-1165G7 processor, 16GB of RAM, and a 512GB SSD, running Windows 11. The implementation was developed and executed in Visual Studio Code, where the necessary code was written to simulate the experimental conditions. While the full code and dataset are available [44], we summarize the key aspects here to ensure clarity and completeness. The implementation of our GA-based solution was carried out using Python with numpy v2.26.2, whereas our MILP-based implementation was developed also with Python and the linear programming modeler Pulp v2.7.0, using CBC as a solver. All the code and data required to run the experiments is accessible from [44].

We conducted different sets of experiments for each of the case studies described in Section 2, instantiating the general problem formalized in Section 3 for each of the domains according to the mapping provided in Table 1.

Table 1. Mapping of our CPS problem to different domains.

Concept	EV Charging Infrastructure	Robomax: Food Logistics
Resource	Charger	Robot
Consumer	Electric Vehicle	Patient
Task	Charging EV	Serving inpatient food
Constraint	EV on location and available for charging	Companion at inpatient room
Data source	DB updated via app	Inpatient record
Objectives	Timespan (min.)	Timespan (min.)
	Cost of charge (min.)	#Disruptions (min.)

A set of tests is conducted for each problem size and case study, considering different variances in consumer availability times and varying probabilities of resource failures. For each problem, we considered increasing instance sizes in the set $\{(2, 20), (4, 40), (8, 80), (16, 160)\}$, where every couple $(|R|, |C|)$ designates the size of the set of reources and consumers of the problem instance, respectively.

For every set of experiments, and problem instance, we run our MILP baseline once (because it always yields the same result), and our GA-based algorithm 30 times for statistical significance.

5.2 Results

(RQ1) ***Effectiveness****.* To determine the effectiveness of our approach, we performed two sets of experiments comparing how effective MILP and our GA were in problem instances of increasing size (cf. Section 5.1) both in the EV charging infrastructure and in the RoboMax instantiations of the problem.

(RQ1.1) How effective is our approach in comparison to an optimal algorithm with no uncertainties considered?

To answer this question, we performed a set of experiments in which the results of the algorithms were evaluated under a deterministic environment in which the availability of resources and consumers coincided exactly with their availability windows provided as input to the planning, and resources were failure-free. Figure 4 illustrates the results that compare the effectiveness of our GA-based approach with the MILP baseline .

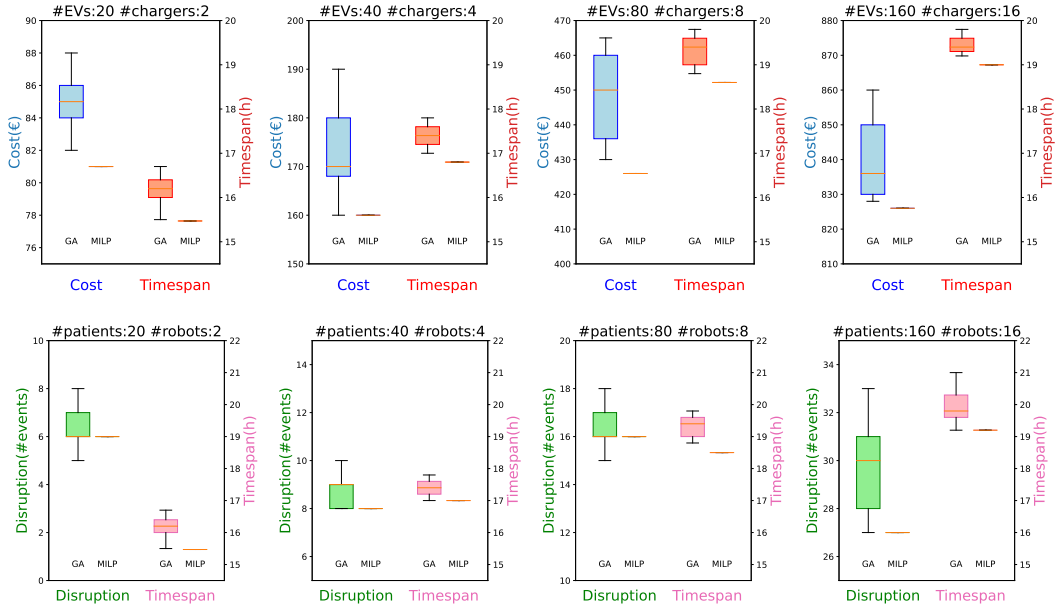

Fig. 4. Effectiveness of our approach vs MILP baseline (no uncertainty): EV charging (top), RoboMax (bottom).

One of the first things that can be observed in the top of the figure (EV charging infrastructure), is that the MILP baseline almost always outperforms the GA both in terms of timespan and cost minimization, because it is providing an optimal solution to the problem (that also explains that the MILP-based results are always the same and not spread out like the ones yielded by the GA). One notable exception to the dominant trend of the MILP is the disruption metric in two of the four problem instances of RoboMax (Figure 4, bottom), which is due to the fact that the algorithms are not explicitly optimizing to minimize this metric (rather, the minimization of disruption in this case is derived from the fact that the algorithm attempts to generate a solution that respects all the constraints).

Concluding remarks on RQ1.1: In the absence of uncertainty , an optimal mixed-integer linear programming algorithm will almost invariably outperform our GA-based solution.

(RQ1.2) How effective is our approach in comparison to an optimal algorithm under prescribed levels of uncertainty in temporal availability constraints? To answer this question, we performed a set of experiments in which the algorithms were evaluated against multiple versions of the environment in which the availability start and end of an element (i.e., consumer or resource) were generated according to probability distributions with means fixed to coincide with the deterministic version of the availability window (i.e., $\forall e \in E \bullet f_{t_e^\alpha}.\mu = W(e).t^\alpha \wedge f_{t_e^\omega}.\mu = W(e).t^\omega$), and different variance values ($f_{t_e^\star}.\sigma^2 \in \{0.1, 0.25, 0.5, 0.75, 1\}$). In this set of experiments, the GA planning solution was configured to consider distributions with variance $\sigma^2 = 0.5$ for the sampling of the κ variants (function *spl*, Section 4.3).

Figure 5 illustrates the results for the two largest problem instances of the EV charging infrastructure problem. Results show how the GA-based approach outperforms the MILP baseline both

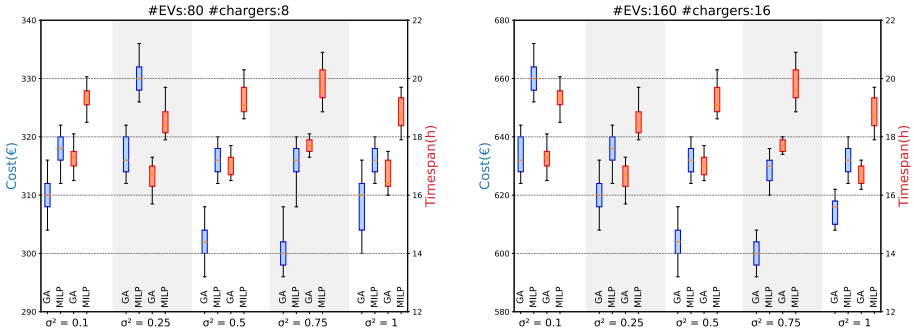

Fig. 5. Effectiveness of our approach vs MILP baseline with uncertainty in temporal availability (EV charging).

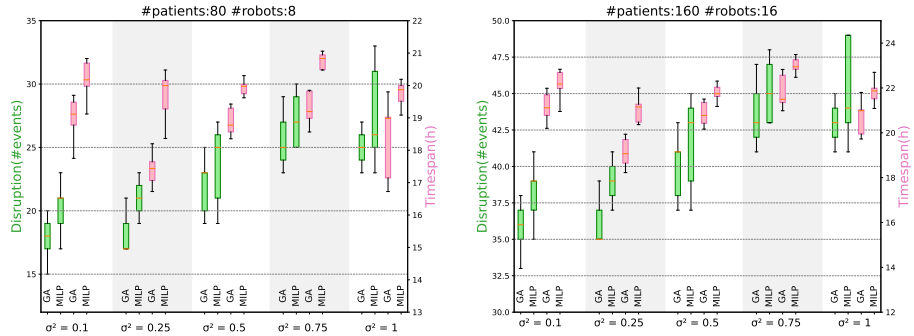

Fig. 6. Effectiveness of our approach vs MILP baseline with uncertainty in temporal availability (RoboMax).

in cost and timespan, across all instances and variance values. It is worth noting, that even for small variance values (e.g., $\sigma^2 = 0.1$), the GA shows a non-marginal improvement over the MILP baseline (ranging between 3.12% and 5.92% for cost, and 8.64% and 11.76% for timespan). Moreover, for variance values that coincide or are close to the one employed to configure the GA algorithm (i.e., $\sigma^2 = \{0.5, 0.75\}$), the difference between GA and MILP is even more remarkable, going up to 6.78% for cost and 11.48% for timespan. Figure 6 shows the results for the RoboMax food logistics problem, which confirm the general trend identified in the EV charging infrastructure problem. Again, the GA always performs clearly better than the MILP baseline both in terms of the disruption and timespan metrics. For small variance values (e.g., $\sigma^2 = 0.1$), the genetic algorithm (GA) demonstrates a significant improvement over the mixed-integer linear programming (MILP) baseline, with timespan reductions ranging from 4.68% to y 11.23% and disruption reductions ranging from 4.87% to 9.31%. For variance values that match or are close to those used to configure the GA algorithm (i.e., $\sigma^2 = \{0.5, 0.75\}$), the difference between GA and MILP becomes even more pronounced, with timespan reductions reaching up to 13.14% and disruption reductions up to 10.52%. In this case, the difference between the GA and MILP is not as strong in the disruption metric, which also presents values more spread out than other metrics, most likely because the algorithms are not explicitly optimizing for it, as discussed in RQ1.1.

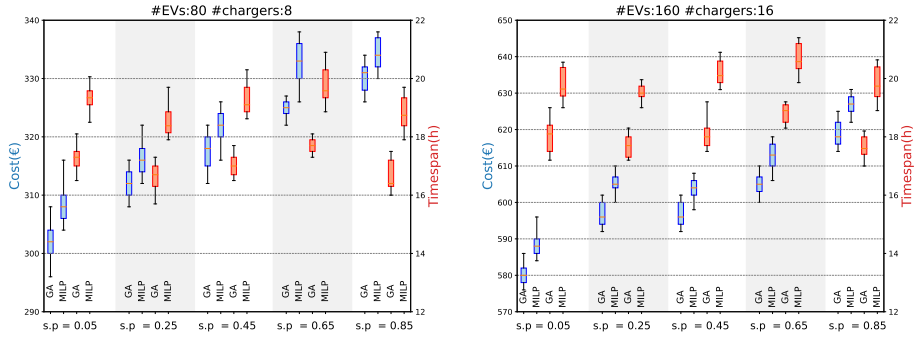

Fig. 7. Effectiveness of our approach vs MILP baseline with uncertainty in elements reliability (EV charging).

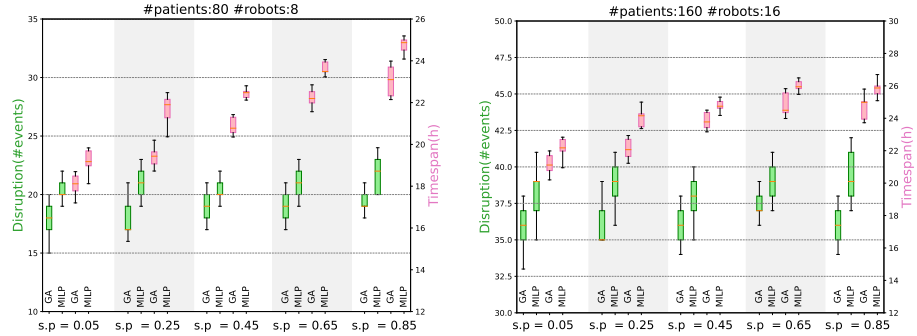

Fig. 8. Effectiveness of our approach vs MILP baseline with uncertainty in elements reliability (RoboMax).

Concluding remarks on RQ1.2: Under prescribed levels of uncertainty in temporal availability constraints, our GA-based approach yields robust solutions that always outperform those obtained by the MILP baseline, which exhibit a notable degradation even under low levels of uncertainty.

(RQ1.3) How effective is our approach in comparison to an optimal algorithm under prescribed levels of uncertainty in elements reliability?

To answer this question, we performed a set of experiments in which the algorithms were evaluated against multiple versions of the environment where the resources are prone to fail with probabilities that fluctuate over time. In this case, there is no uncertainty associated with temporal availability constraints, so the availability of consumers coincides exactly with their availability windows provided as input to the planning. Various probabilities are considered for different time slots during the day. For instance, the cumulated probability of charger failure between 8a.m. and 12p.m. might be 0.1, whereas that same probability might change between 1p.m. and 6p.m. to 0.35 due to weather conditions, increased usage, or longer operating hours among other factors.

To carry out the experiments, different values of the failure probability are established for different time slots. The total cumulated probability for that day is calculated using CDF (cf. Section 3). We

start by taking low values for failure probability, starting with 0.05 and increasing by 0.20 up to 0.85. For each value, a set of tests is conducted, each one consisting of multiple algorithm executions.

Figure 7 presents the results for the two largest problem instances of the EV charging infrastructure problem. The findings demonstrate that the GA-based approach outperforms the MILP baseline in terms of both cost and timespan across all instances and variance values. It should be noted that even for small probability of failure values (e.g., sp=0.05), the GA shows an improvement over the MILP baseline (ranging between 1.55% and 2.66% for cost, and 7.74% and 12.5% for timespan). In this case, the improvements in cost are less pronounced compared to the experiments conducted for RQ1.2. This is due to the fact that the cost is more influenced by variations in vehicle availability windows than by the number of chargers available. In the RoboMax case, for small failure probability values our approach also shows an improvement over the MILP baseline (ranging between 5.55% and 8.1% for disruption , and 6.01% and 11.53% for timespan). For higher probability of failure values (i.e., s.p. = 0.85), the difference between GA and MILP in the EV case is even more remarkable, especially for timespan, which goes up to 16.67%, with cost also going up to 2.77%.

Figure 8 illustrates the results for the RoboMax food logistics problem, reinforcing the general trend observed in the EV charging infrastructure problem. Once again, our approach consistently outperforms the baseline in both the disruption and timespan metrics. However, the difference between the GA and MILP is less pronounced in the disruption metric, which also shows a wider spread of values compared to other metrics. This variability is likely due to the fact that the algorithms are not explicitly optimizing for disruption, as discussed in RQ1.1.

Concluding remarks on RQ1.3: Under prescribed levels of uncertainty on elements reliability, our approach consistently generates robust solutions that outperform those yielded by the baseline. Notably, even when the probability of failure is minimal, the solutions from the MILP baseline exhibit significant degradation. However, it is worth noting that the improvement is more limited than the one observed in RQ1.2.

(RQ1.4) How effective is our approach in comparison to an optimal algorithm under prescribed levels of uncertainty in temporal availability constraints and elements reliability?

To answer this question, we performed a set of experiments in which the algorithms were evaluated against multiple versions of the environment that combine the uncertainty in temporal availability constraints as set up for RQ1.2, and the element failure probabilities as employed in RQ1.3.

Tables 2 and 3 illustrate the results for one of the largest problem instances of the EV charging infrastructure problem. Table 2 sumarizes the results for cost and Table 3 for timespan. Results show how our approach outperforms the MILP baseline both in cost and timespan, across all instances, variance and probability values. In particular, for small values of both the failure probability ($s.p = 0.05$) and variance ($\sigma^2 = 0.1$), it is observed that the cost obtained by the genetic algorithm decreases on average by 5.96% compared to that obtained by the MILP algorithm, while the timespan decreases on average by 16.16%. For high values of failure probability ($s.p = 0.85$) and variance ($\sigma^2 = 1$), the cost obtained by the genetic algorithm decreases on average by 22.77% compared to that obtained by the MILP algorithm, while the timespan decreases on average by 23.89%. Overall, it is evident that the combination of both uncertainties amplifies the differences between the results obtained by the genetic algorithm and the MILP algorithm, with the results obtained by the genetic algorithm being remarkably more robust. Figure 9 reinforces these results. The plot on the left-hand side considers a charger failure probability of 25%, whereas the plot in the right-hand side considers

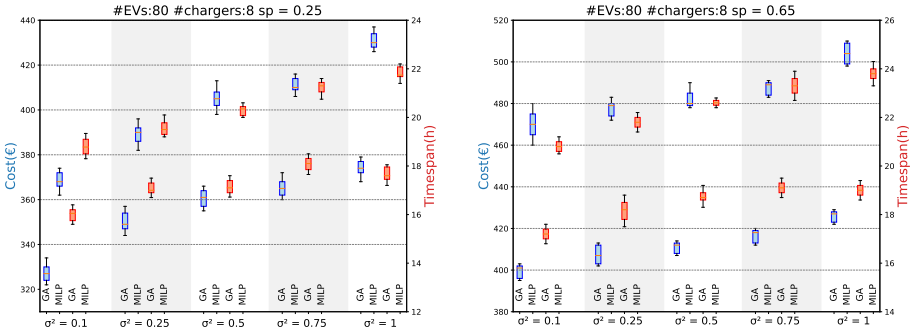

Fig. 9. Effectiveness of our approach vs MILP baseline with uncertainty in both, temporal availability and elements reliability (EV charging).

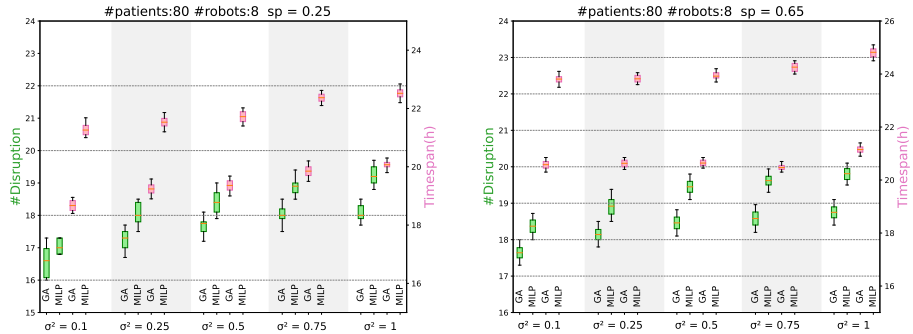

Fig. 10. Effectiveness of our approach vs MILP baseline with uncertainty in both, temporal availability and elements reliability (RoboMax).

a failure probability of 65%. When both types of uncertainties are combined, it can be observed that the difference both in cost and timespan obtained by the two algorithms increases substantially.

Tables 4 and 5 illustrate the results for RoboMax case. Table 4 summarizes the results for disruption and Table 5 for timespan. Again, results show that our approach outperforms the MILP baseline in both metrics. For small values the failure probability ($s.p = 0.05$) and variance ($\sigma^2 = 0.1$), it is observed that the number of disruptions obtained by the genetic algorithm decreases on average by 1.88% compared to that obtained by the MILP algorithm, while the timespan decreases on average by 13.81%. For higher values of failure probability ($s.p = 0.85$) and variance ($\sigma^2 = 1$), the number of disruptions obtained by the genetic algorithm decreases on average by 6.16% compared to that obtained by the MILP algorithm, while the timespan decreases on average by 20.02%.

In RoboMax, it is also clear that the combination of both uncertainties add up to increase the difference between the outcomes produced by the genetic algorithm and those generated by the MILP algorithm, with the genetic algorithm's results proving to be significantly more robust. Figure 10 reinforces these results. Again, the difference between the GA and MILP is less pronounced in the disruption metric, which also shows a wider spread of values compared to other metrics. This

variability is likely due to the fact that the algorithms are not explicitly optimizing for disruption, as discussed previously.

In both cases, the evaluation results further demonstrate that the interplay of these uncertainties significantly impacts system performance, reinforcing that they do not simply co-exist but interact within the planning process. This confirms that our approach effectively captures their combined effect.

Algorithmic robustness, as defined by Jensen et al. [24], refers to the sustained performance of a system despite changes in its operating environment. In our case, these changes correspond to increasing levels of uncertainty in temporal availability constraints and element reliability. Our evaluation systematically assesses robustness by analyzing how the algorithm adapts under different uncertainty levels (RQ1.2 - RQ1.4). The results indicate that our approach maintains feasible solutions and mitigates performance degradation, demonstrating robustness in uncertain conditions.

Concluding remarks on RQ1.4:Under prescribed levels of uncertainty both in temporal availability constraints and elements reliability, our approach produces robust solutions that consistently outperform those obtained by the MILP baseline. Notably, even with marginal uncertainty, solutions derived from the MILP baseline exhibit significant degradation. Differences in performance between GA and MILP become much more pronounced when both uncertainties are combined, compared to RQ1.2 and RQ1.3.

Table 2. Cost: GA vs MILP with uncertainty in temporal availability and elements reliability. EV charging.

	σ^2 = 0.1			σ^2 = 0.25			σ^2 = 0.5			σ^2 = 0.75			σ^2 = 1		
	MILP	**GA**	**ΔCost**	**MILP**	**GA**	**ΔCost**	**MILP**	**GA**	**ΔCost**	**MILP**	**GA**	**ΔCost**	**MILP**	**GA**	**ΔCost**
s.p = 0.05	326.85	308.46	-18.39 (-5.96%)	345.65	318.62	-27.03 (-8.48%)	366.74	330.54	-36.2 (-10.95%)	374.25	338.43	-35.82 (-10.58%)	386.52	345.37	-41.15 (-11.91%)
s.p = 0.25	368.76	327.51	-41.25 (-12.59%)	392.74	349.37	-43.37 (-12.41%)	406.87	361.41	-45.46 (-12.57%)	411.72	364.54	-47.18 (-12.94%)	432.41	376.75	-55.66 (-14.77%)
s.p = 0.45	410.48	357.38	-53.1 (-14.85%)	435.97	374.52	-61.45 (-16.40%)	443.21	377.74	-65.47 (17.33%)	438.51	381.96	-56.55 (-14.80%)	470.24	392.41	-77.83 (-19.83%)
s.p = 0.65	469.96	401.75	-68.21 (-16.97%)	478.48	407.65	-70.83 (-17.37%)	485.42	412.18	-73.24 (-17.76%)	489.38	418.43	-70.95 (-16.95%)	504.38	427.54	-76.84 (-17.97%)
s.p = 0.85	500.18	424.65	-75.53 (-17.78%)	518.53	438.46	-80.07 (-18.26%)	532.54	441.76	-90.78 (-20.54%)	542.63	448.52	-94.11 (-20.98%)	575.65	468.86	-106.79 (-22.77%)

Table 3. Timespan: GA vs MILP with uncertainty in temporal availability and elements reliability. EV charging.

	σ^2 = 0.1			σ^2 = 0.25			σ^2 = 0.5			σ^2 = 0.75			σ^2 = 1		
	MILP	**GA**	**ΔTmsp**	**MILP**	**GA**	**ΔTmsp**	**MILP**	**GA**	**ΔTmsp**	**MILP**	**GA**	**ΔTmsp**	**MILP**	**GA**	**ΔTmsp**
s,p = 0.05	18.76	16.15	-2.61 (-16.16%)	18.58	16.63	-1.95 (-11.72%)	19.53	16.76	-2.77 (-16.52%)	20.57	17.54	-3.03 (-17.27%)	19.94	17.26	-2.68 (-15.52%)
s,p = 0.25	18.97	16.08	-2.89 (-17.97%)	19.42	17.12	-2.30 (-13.43%)	20.65	17.24	-3.41 (-19.78%)	21.28	18.27	-3.01 (-16.47%)	21.43	17.68	-3.75 (-21.21%)
s,p = 0.45	20.12	16.82	-3.30 (-19.62%)	20.47	17.58	-2.89 (-16.43%)	21.35	17.78	-3.57 (-20.08%)	22.65	18.78	-3.87 (-20.61%)	22.42	18.21	-4.21 (-23.12%)
s,p = 0.65	20.76	17.18	-3.58 (-20.83%)	21.72	18.15	-3.57 (-19.67%)	22.64	18.82	-3.82 (-20.29%)	23.17	19.37	-3.80 (-19.62%)	23.78	19.04	-4.74 (-24.89%)
s,p = 0.85	22.74	18.68	-4.06 (-21.73%)	22.54	18.46	-4.08 (-22.10%)	23.26	19.15	-4.11 (-21.46%)	24.16	19.65	-4.51 (-22.95%)	25.56	20.63	-4.93 (-23.89%)

(RQ2) ***Impact****. How much do the different sources of uncertainty impact system goals?*

To answer this question, we studied the correlation between significant variables (uncertainty-related, such as variance of uncertainty in temporal availability constraints σ^2 and failure probabilities $s.p$) and the different measures of performance (e.g., cost, timespan) using Pearson's method [17]. Pearson's correlation coefficient is a suitable measure for assessing the strength of the relationship

Table 4. Disruption: GA vs MILP with uncertainty in temporal availability and elements reliability. RoboMax.

	σ^2 = 0.1			σ^2 = 0.25			σ^2 = 0.5			σ^2 = 0.75			σ^2 = 1		
	MILP	**GA**	**ΔDisr.**	**MILP**	**GA**	**ΔDisr.**	**MILP**	**GA**	**ΔDisr.**	**MILP**	**GA**	**ΔDisr.**	**MILP**	**GA**	**ΔDisr.**
s.p = 0.05	16.82	16.51	-0.31 (-1.88%)	17.51	16.86	-0.65 (-3.86%)	17.86	17.16	-0.7 (-4.08%)	18.24	17.34	-0.9 (-5.19%)	18.76	17.52	-1.24 (-7.08%)
s.p = 0.25	17.35	16.95	-0.4 (-2.36%)	18.15	17.32	-0.83 (-4.79%)	18.43	17.72	-0.71 (-4%)	18.72	17.95	-0.77 (-4.11%)	19.14	18.03	-1.11 (-6.16%)
s.p = 0.45	17.68	17.23	-0.45 (-2.61%)	18.57	17.61	-0.96 (-5.45%)	18.87	17.93	-0.94 (5.24%)	19.12	18.13	-0.99 (-5.46%)	19.36	18.28	-1.08 (-5.9%)
s.p = 0.65	18.37	17.64	-0.73 (-4.14%)	18.92	18.14	-0.78 (-4.3%)	19.45	18.46	-0.99 (-5.36%)	19.62	18.58	-1.04 (-5.59%)	19.81	18.75	-1.06 (-5.65%)
s.p = 0.85	18.94	18.26	-0.68 (-3.72%)	19.52	18.57	-0.95 (-5.12%)	19.79	18.71	-1.08 (-5.77%)	19.98	18.82	-1.16 (-6.16%)	20.13	19.04	-1.09 (-5.72%)

Table 5. Timespan: GA vs MILP with uncertainty in temporal availability and elements reliability. RoboMax.

	σ^2 = 0.1			σ^2 = 0.25			σ^2 = 0.5			σ^2 = 0.75			σ^2 = 1		
	MILP	**GA**	**ΔTmsp**	**MILP**	**GA**	**ΔTmsp**	**MILP**	**GA**	**ΔTmsp**	**MILP**	**GA**	**ΔTmsp**	**MILP**	**GA**	**ΔTmsp**
s,p = 0.05	20.85	18.32	-2.53 (-13.81%)	20.69	18.75	-1.94 (-10.35%)	21.13	18.81	-2.32 (-12.33%)	21.25	19.64	-1.61 (-8.18%)	21.46	19.78	-1.68 (-8.49%)
s,p = 0.25	21.26	18.67	-2.59 (-13.87%)	21.53	19.24	-2.29 (-11.9%)	21.72	19.36	-2.36 (-12.19%)	22.37	19.85	-2.52 (-12.69%)	22.52	20.07	-2.45 (-12.21%)
s,p = 0.45	22.31	19.35	-2.96 (-15.3%)	22.57	19.65	-2.92 (-16.54%)	22.78	19.86	-2.92 (-14.7%)	23.25	20.04	-3.21 (-16.01%)	23.51	20.34	-3.17 (-15.59%)
s,p = 0.65	23.81	20.58	-3.23 (-15.69%)	23.82	20.63	-3.19 (-15.46%)	23.92	20.64	-3.28 (-15.89%)	24.26	20.47	-3.79 (-18.51%)	24.82	21.15	-3.67 (-17.35%)
s,p = 0.85	24.78	21.37	-3.41 (-15.96%)	24.61	20.84	-3.77 (-18.09%)	24.64	21.15	-3.49 (-16.5%)	25.28	21.22	-4.06 (-19.13%)	26.14	21.78	-4.36 (-20.02%)

Table 6. Improvement of our approach under different uncertainty variants: EV charging.

	Improvement over MILP (%)									
Uncertainty source(s)	**Temporal availability (UT)**		**Resource reliability (UR)**		**Combined (UTR)**		**Δ Cost**		**Δ Timespan**	
	Cost	**Timespan**	**Cost**	**Timespan**	**Cost**	**Timespan**	**UT-UTR**	**UR-UTR**	**UT-UTR**	**UR-UTR**
σ^2 = 0.1 s.p = 0.05	3.12	8.64	2.66	7.74	6.45	14.58	3.33	3.79	5.94	6.84
σ^2 = 0.25 s.p = 0.65	3.86	9.15	4.34	9.64	7.76	16.72	3.9	3.42	7.57	7.08
σ^2 = 0.75 s.p = 0.25	4.97	10.08	3.85	9.21	8.73	18.42	3.76	4.88	8.34	9.21
σ^2 = 1 s.p = 0.85	6.78	11.48	5.74	13.1	12.47	19.75	5.69	6.73	8.27	6.65

Table 7. Improvement of our approach under different uncertainty variants: RoboMax.

	Improvement over MILP (%)									
Uncertainty sources	**Temporal availability (UT)**		**Resource reliability (UR)**		**Combined UTR**		**Δ Disruption**		**Δ Timespan**	
	Disruption	**Timespan**	**Disruption**	**Timespan**	**Disruption**	**Timespan**	**UT-UTR**	**UR-UTR**	**UT-UTR**	**UR-UTR**
σ^2 = 0.1 s.p = 0.05	4.87	4.68	5.55	6.01	8.45	12.31	3.58	2.9	7.63	6.3
σ^2 = 0.25 s.p = 0.65	7.18	8.56	7.64	9.64	10.32	15.22	3.14	2.68	6.67	5.58
σ^2 = 0.75 s.p = 0.25	9.31	11.23	8.12	11.56	13.68	17.67	4.37	5.56	6.44	6.11
σ^2 = 1 s.p = 0.85	10.52	12.14	9.73	12.55	15.24	19.78	4.72	5.51	7.64	7.23

between two quantitative, linearly correlated variables. To determine the statistical significance of the correlation coefficients we obtained, we performed two-tailed t-tests on each pair of variables. The following pairs of variables were included in this part of the analysis:

- EV Charging infrastructure: (i) σ^2-Cost, (ii) σ^2-Timespan, (iii)$s.p$-Cost, (iv) $s.p$-Timespan.
- RoboMax: (i) σ^2-Disruption, (ii) σ^2-Timespan, (iii)$s.p$-Disruption, (iv) $s.p$-Timespan.

As a preliminary step, we created a scatter plot for each pair of variables to visualize the relationship between them. In all cases the function that best approximates the relationship is linear, as it can be observed in Figures 11 and 12 (EV charging infrastructure), as well as Figures 13 and 14 (RoboMax). This step confirmed that Pearson's method is adequate to analyze the correlation of

Table 8. Correlation matrix (EV charging).

	Cost	**Timespan**	σ^2	$s.p$
Cost	1	0.976	0.988	0.992
Timespan	0.976	1	0.981	0.990
σ^2	0.988	0.981	1	–
$s.p$	0.992	0.990	–	1

Table 9. Correlation matrix (RoboMax).

	#Disruptions	**Timespan**	σ^2	$s.p$
#Disruptions	1	0.874	0.911	0.887
Timespan	0.874	1	0.994	0.990
σ^2	0.9117	0.994	1	–
$s.p$	0.8878	0.990	–	1

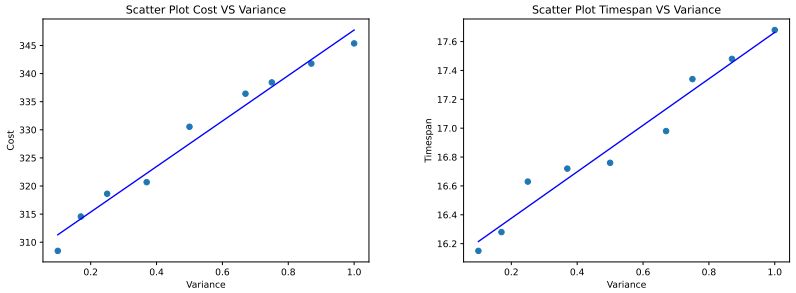

Fig. 11. Correlation between Cost-Variance and Timespan-Variance (EV Charging).

these variables. Pearson coefficients for the EV charging infrastructure and RoboMax case studies are included in Tables 8 and 9, respectively. Note that for our analysis, all p-values obtained in both case studies are below 0.01, indicating the statistical significance of the correlation coefficients.

Table 8 (EV charging infrastructure), shows high correlation coefficients across the board, which indicate strong positive relationships between the variables. The cost is strongly correlated with both variance σ^2 (0.988) and failure probability $s.p$ (0.992), suggesting that both increased variance in EV availability and higher charger failure probabilities entail higher costs, as expected. Similarly, the timespan shows strong correlations with variance (0.981) and failure probability (0.99).

In RoboMax (Table 9) the timespan is strongly correlated with both variance (0.994) and failure probability (0.99). In this case, the number of disruptions also shows a fairly strong correlation with variance (0.911) and failure probability (0.887). However, the correlation coefficients are lower compared to the others. This is consistent with the fact that the algorithm does not explicitly aim at minimizing the number of robot interruptions.

The results obtained for both case studies highlight significant interdependencies among cost /disruption, timespan, variance in temporal availability constraints, and failure probability. However, they do not provide an indication of the magnitude of the impact of uncertainty sources upon system goals. To further inform our answer to this question, we studied the improvement of our approach over MILP under different uncertainty scenarios that include: uncertainty in temporal availability (UT), uncertainty in resource reliability (UR), and both sources of uncertainty combined (UTR). The results in Tables 6 and 7 show that in general, the impact of individual sources of uncertainty applied independently are similar between them, whereas their combined effect tends to approximately double the increase in system metrics almost invariably. If we piece this information together with that of the correlation study, the generalized positive correlation of the effects of sources of uncertainty combined with the additive increases in the metrics observed indicates that there is an *augmentation behavior* in which the effects of the sources of uncertainty are strongly correlated, but none of them dominates the other. This contrasts with other possibilities that might include uncertainty *dominance* (strong correlation, but reducing one source of uncertainty reduces the other), or *conflict* in which sources of uncertainty cancel out each other's effects [6].

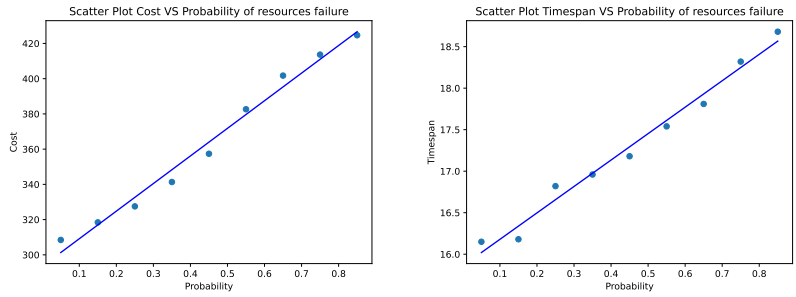

Fig. 12. Correlation between Cost-Probability and Timespan-Probability (EV Charging).

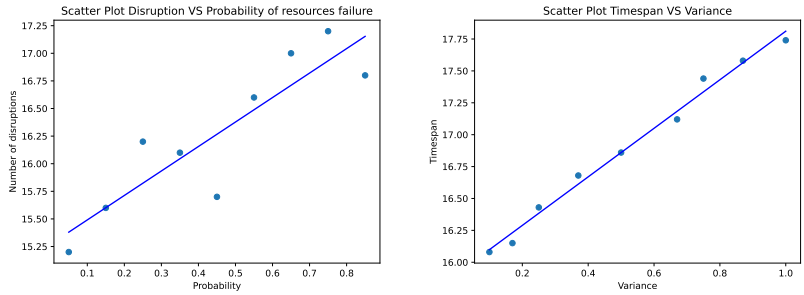

Fig. 13. Correlation between Disruption-Variance and Timespan-Variance (RoboMax).

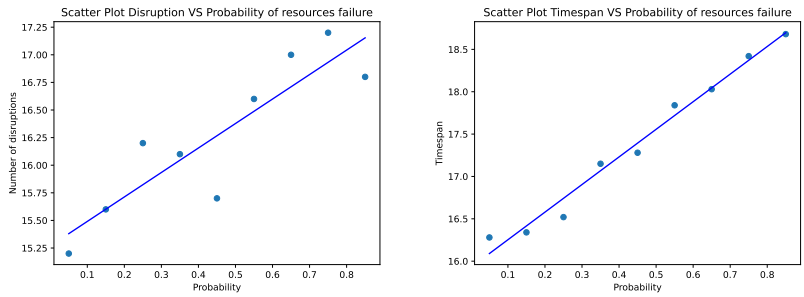

Fig. 14. Correlation between Cost-Variance and Cost-Timespan (RoboMax).

Concluding remarks on RQ2: There are significant dependencies indicated by positive correlations between the effects of the sources of uncertainty (variance and failure probability) and system goal metrics (cost/disruption, timespan). This, along with the additive increases on metrics observed in scenarios that combine both sources of uncertainty (UTR), indicates a compound impact on the metrics that suggests an *augmentation behavior* in which the effects of the sources of uncertainty are strongly correlated, but none of them dominates the other.

Table 10. **Planning Execution Times: GA vs MILP.**

EV Charging Infrastructure

$\|R\|$	$\|C\|$	GA Time ms	MILP Time ms	ΔTime %
2	5	31.82 ± 125.65	27.87 ± 132.98	-12.41
2	20	269.25 ± 156.87	281.36 ± 247.5	+4.49
4	40	537.84 ± 322.8	1897.56 ± 361.82	+252.81
8	80	1584.92 ± 475.14	6728.17 ± 463.81	+324.51
16	160	3601.23 ± 345.26	15149.28 ± 498.88	+320.67

Robomax: Food Logistics

$\|R\|$	$\|C\|$	GA Time ms	MILP Time ms	ΔTime %
2	5	42.54 ± 140.21	45.21 ± 145.68	+6.28
2	20	328.23 ± 436.83	462.81 ± 420.38	+41.00
4	40	651.15 ± 452.64	1901.92 ± 498.29	+192.09
8	80	1757.84 ± 390.58	6260.25 ± 484.72	+256.13
16	160	3681.21 ± 520.04	14928.53 ± 421.32	+305.53

(RQ3) ***Efficiency****. How efficient is our approach with respect to an optimal algorithm?*

To answer this question, we measured the execution times both for the MILP-based and the GA-based approaches from the set of experiments carried out for RQ1.1-RQ1.4. Table 10 summarizes the results obtained for the different problem instance sizes, which show how the efficiency of the GA-based solution is clearly better than that of the MILP baseline. As expected, the difference in computation time for small problem instances is marginal, but as the problem instance size increases, we observe remarkable efficiency differences that go up to a 327.67% increment of computation time of MILP with respect to the GA, for problem instances of 16 resources and 160 consumers.

Concluding remarks on RQ3: The GA-based approach is clearly more efficient than MILP, yielding execution times that are comparable for small problem instances, but differ remarkably as the problem instance size increases.

5.3 Threats to Validity

There are four types of threats that can affect the validity of our study according to Wohlin et al. [53]. They are summarized in the following.

5.3.1 Construct validity threats. These threats are concerned with the relationship between theory and what is observed. We designed our approach with two interacting sources of uncertainty, namely uncertainty in temporal availability constraints and element reliability. However, we are aware that there are other sources of uncertainty that might affect the effectiveness of our approach [31, 38, 39] (e.g., uncertainties due to model abstraction, imperfect sensing and actuation). Hence, we will perform further experiments in future work, in which we will explore the interaction of different sources of uncertainty [11].

5.3.2 Internal validity threats. They are related to those factors that might affect the results of our evaluation. We compared our genetic algorithm with a MILP baseline. The choice of Mixed-Integer Linear Programming (MILP) as the baseline is motivated by its ability to provide optimal solutions in the absence of uncertainty. MILP serves as a fundamental reference point since any other heuristic or approximate algorithm, in the best case, would converge to the same solution as MILP. Additionally, MILP is widely used in optimization problems similar to ours (e.g.[33]), making it a reliable benchmark for performance evaluation.

We acknowledge that there are other alternatives for addressing this problem, such as well-known Multi-Objective Genetic Algorithms (MOGAs) like NSGA-II [48], constraint programming, dynamic programming, and heuristic approaches like Simulated Annealing and Tabu Search. These methods do not guarantee optimality and often rely on problem-specific heuristics. Similarly, reinforcement learning-based approaches introduce additional complexity and may require extensive training data, making them less suited for direct comparison in our setting.

Other methods are probabilistic and statistical model checking [25, 28], that can be adapted to our problem and potentially outperform MILP. We plan to explore the tradeoffs in terms of efficiency and effectiveness of such alternatives in the future.

5.3.3 External validity threats. These threats have to do with the extent to which it is possible to generalize the findings of the experiments. Our experiments have been performed on two case studies, which are simulations of two different CPS. Without conducting more experiments in the real CPS and other types of CPS, we cannot generalize the effectiveness and efficiency of our approach. However, we want to point out that: (i) our problem formalization has been general enough to be easily applicable to two CPS that differ in terms of constituent components, tasks, roles of humans-in-the-loop, and mission objectives, and (ii) in our context, conducting experiments with a real CPS requires that we connect the implementation of our approach to it during operation for long periods of time. Despite the fact that setting up this type of experiment is complex and expensive, we plan to expand our evaluation with experiments in real testbeds, including the EV charging infrastructure in our research facility (cf. Figure 1).

5.3.4 Conclusion validity threats. These threats are concerned with the issues that affect the ability to draw correct conclusions from the data obtained from the experiments. We evaluate our method with metrics that are specific to the planning problem at hand (such as qualities specific to each CPS system like cost or timespan), and not in the broader context of the adaptation loop. However, other metrics could be adopted to evaluate the overall qualities of the adaptive system, e.g. [7, 49]. We will include these additional metrics in future work while conducting experiments in the real testbeds, where there is a clear opportunity to collect data that enables a broader evaluation of the adaptation loop.

6 RELATED WORK

Uncertainty management in Cyber-Physical Systems (CPS) has been widely studied in different contexts. Zhang et al. [54] propose a conceptual model to classify and understand uncertainty in CPS, emphasizing the need for systematic identification of uncertainty sources. Other works focus on uncertainty-aware task scheduling and planning in self-adaptive CPS, such as [13], which explores techniques to handle environmental and operational uncertainty during task execution. While these studies provide valuable insights, they do not specifically address the combined effect of uncertainty in temporal availability constraints and resource failures in task-based CPS. Our work extends this research by developing a planning approach that mitigates multiple sources of uncertainty, ensuring adaptability in dynamic environments.

Other than the works related to the general field of CPS, the number of approaches for planning under uncertainty has recently experienced substantial growth in multiple other areas. Although we cannot give an exhaustive account of existing approaches, we summarize in this section some that are relevant to our proposal and belong to various (potentially overlapping) areas of research.

Scheduling. There are multiple proposals that deal with the assignment of resources under availability constraints, although uncertainty in resource availability has been largely unexplored until recently [1]. One of the notable exceptions is the algorithm developed by Wang et al. [50], which addresses the *Multi-Skilled Resource-Constrained Project Scheduling Problem* (MSRCPSP) with uncertainty in resource availability to minimize the *makespan* of a set of tasks, although it is not equipped to deal with other optimization objectives and their trade-offs.

Self-adaptive systems. Cheng et al.'s [14] early work proposes resource availability prediction as a means to achieve proactive adaptation, a direction that was later followed by other authors (e.g., [16, 36]), but without support for explicit resource availability constraints. DECIDE [5] is an approach to build distributed self-adaptive software used in mission-critical applications that

uses quantitative verification at runtime to agree individual component contributions to meeting system-level quality-of-service requirements, and then to ensure that components achieve their agreed contributions to the mission in the presence of changes and failures. Despite not explicitly considering temporal availability constraints, this approach is able to provide robust contributions to the mission in the presence of failures. Cámara et al. [10] propose a planning approach for adaptive CPS able to consider the impact of mutual dependencies between software architecture and task planning on mission goals. The approach is able to generate adaptation decisions both on the reconfiguration of the system's software architecture (which can be carried out in any physical location), and (physical) task plans. Solutions provide both structural and quantitative formal guarantees under prescribed levels of uncertainty in the reliability of task executions. Solutions are generated using a combination of Alloy [23] and the PRISM probabilistic model checker [26], and provide both structural and quantitative formal guarantees under prescribed levels of uncertainty in the reliability of task executions. Carwehl et al. [12] propose an architecture comprising an uncertainty reduction controller that drives the adaptive acquisition of new information within the self-adaptation loop, and a method that uses probabilistic model checking to synthesize such controllers that aim at delivering optimal trade-offs between uncertainty reduction benefits and new information acquisition costs. Filippone et al. [20] propose an approach for the realization of adaptable multi-robot systems, which are capable of dealing with uncertainties by adapting their mission at runtime. The approach hinges on the concept of *adaptable task* that is used in the global mission specification of the MRS to identify the mission tasks affected by uncertainties. Adaptation alternatives are modeled as sub-missions and associated with the adaptable task. At runtime, ad hoc written *trigger functions* executed by robots sense and evaluate the environment and select the most suitable adaptation alternative for execution.

Multi-agent and multi-robot systems. The Genetic Mission Planner (GMP) [33] casts a mission for heterogeneous multi-agent systems as an extension of the traveling salesperson problem described through a mixed-integer linear programming formulation and solve it using genetic algorithms. Their results show that, in the presence of precedence timing constraints, their genetic planner outperforms another solution implemented via mixed-integer linear programming. In a more recent piece of work [34], the authors explore optimization of tasks via parallelization, making an explicit distinction between so-called *virtual* and *physical* tasks, which are analogous to the reconfiguration/physical tasks described in [10]. Liu et al. [29] propose a multi-objective strength learning particle swarm optimization (SLPSO) algorithm to optimize multiple objectives. In this article, the cooperative multi-robot task planning problem is converted into a two-step problem of task permutation construction and robot subset selection. KANOA [47] combines constraint solving with quantitative verification for multi-robot task allocation and scheduling that is able to deal with task precedence constraints and multiple optimization objectives. MAPmAKER [32] is an approach to decentralized planning for multi robot systems that uses a variant of three-valued LTL semantics to reason under partial knowledge about mission satisfaction. Claes at al. [15] tackle the spatial task allocation problem in warehouse commissioning of robots teams, where robots need to service a set of tasks distributed in the warehouse.

Uncertainty Interaction. The Uncertainty Interaction Problem (UIP) [6, 11] poses various challenges related to the representation, analysis, mitigation, and exploration of interacting sources of uncertainty in software-intensive systems such as sCPS. Many of these challenges are largely unaddressed, although the self-adaptive systems community has recently acknowledged them as crucial to attain more resilient adaptive systems [52], and preliminary efforts that explicitly acknowledge interacting sources of uncertainty have started to emerge [8, 21]. However, none of these efforts explicitly addresses planning under multiple sources of uncertainty.

To the best of our knowledge and despite their ability to mitigate different forms of uncertainty, none of the approaches mentioned in the multi-agent/robot, self-adaptive systems, and uncertainty interaction areas are built to consider uncertainty in temporal availability constraints. Moreover, if we focus on the approaches which (like ours) are centered on planning, we notice that they are not equipped to explicitly mitigate the impact of multiple sources of uncertainty (with uncertainty in temporal availability constraints being one of them) upon system goals.

7 CONCLUSIONS

In this article, we have contributed towards addressing the *Uncertainty Interaction Problem* (UIP) [11] in planning for self-adaptation. In particular, we have considered the interaction of both temporal availability constraints and element reliability in task-based cyber-physical systems. To be able to study how the system behaves under both types of uncertainty, we have developed an implementation that uses genetic algorithms and compare it with a baseline implementation that guarantees a globally optimal solution in the absence of uncertainty.

Our results show that the baseline implementation fares better than our approach under no uncertainty. However, when uncertainty is taken into account, which is the case in the real systems, our proposed approach yields solutions that outperform the baseline implementation. The difference between the baseline and our proposal grows as the degree of uncertainty increases. Besides, when both types of uncertainty are present, the difference with the baseline implementation is even greater, and the effects of the various sources of uncertainty are strongly correlated, exhibiting augmentation behavior [6].

In future work, we plan to continue studying the UIP by considering additional sources of uncertainty and evaluating how they affect the overall system's behavior. Moreover, we will explore the use of uncertainty reduction [9, 12, 35] to selectively mitigate the impact of various sources of uncertainty upon system goals.

VERIFIABILITY

For the sake of verifiability, our prototype as well as all artifacts used for our experiments are available on our project's website [44]. Detailed instructions on how to run our prototype and explore our replication package are also available.

REFERENCES

[1] Behrouz Afshar-Nadjafi. 2021. Multi-skilling in scheduling problems: A review on models, methods and applications. *Computers & Industrial Engineering* 151 (2021), 107004.

[2] Mehrnoosh Askarpour, Christos Tsigkanos, Claudio Menghi, Radu Calinescu, Patrizio Pelliccione, Sergio García, Ricardo Caldas, Tim J. von Oertzen, Manuel Wimmer, Luca Berardinelli, Matteo Rossi, Marcello M. Bersani, and Gabriel S. Rodrigues. 2021. RoboMAX: Robotic Mission Adaptation eXemplars. In *16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2021, Madrid, Spain, May 18-24, 2021*. IEEE, 245–251.

[3] Riadh Ben Halima, Marwa Hachicha, Ahmed Jemal, and Ahmed Hadj Kacem. 2023. MAPE-K patterns for self-adaptation in cyber-physical systems. *The Journal of Supercomputing* 79, 5 (2023), 4917–4943.

[4] Tomas Bures, Danny Weyns, Bradley Schmer, Eduardo Tovar, Eric Boden, Thomas Gabor, Ilias Gerostathopoulos, Pragya Gupta, Eunsuk Kang, Alessia Knauss, et al. 2017. Software engineering for smart cyber-physical systems: Challenges and promising solutions. *ACM SIGSOFT Software Engineering Notes* 42, 2 (2017), 19–24.

[5] Radu Calinescu, Simos Gerasimou, and Alec Banks. 2015. Self-adaptive Software with Decentralised Control Loops. In *Fundamental Approaches to Software Engineering - 18th International Conference, FASE 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings (Lecture Notes in Computer Science, Vol. 9033)*, Alexander Egyed and Ina Schaefer (Eds.). Springer, 235–251.

[6] Javier Cámara, Radu Calinescu, Betty H. C. Cheng, David Garlan, Bradley Schmerl, Javier Troya, and Antonio Vallecillo. 2022. Addressing the uncertainty interaction problem in software-intensive systems: challenges and desiderata

(MODELS '22). Association for Computing Machinery, New York, NY, USA, 24–30. https://doi.org/10.1145/3550355.3552438

[7] Javier Cámara and Rogério de Lemos. 2012. Evaluation of resilience in self-adaptive systems using probabilistic model-checking. In *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2012, Zurich, Switzerland, June 4-5, 2012*, Hausi A. Müller and Luciano Baresi (Eds.). IEEE Computer Society, 53–62.

[8] Javier Camara, Sebastian Hahner, Diego Perez-Palacin, Antonio Vallecillo, Maribel Acosta, Nelly Bencomo, Radu Calinescu, and Simos Gerasimou. 2024. Uncertainty Flow Diagrams: Towards a Systematic Representation of Uncertainty Propagation and Interaction in Adaptive Systems. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 37–43.

[9] Javier Cámara, Wenxin Peng, David Garlan, and Bradley R. Schmerl. 2018. Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation. *Sci. Comput. Program.* 167 (2018), 51–69. https://doi.org/10.1016/J.SCICO.2018.07.002

[10] Javier Cámara, Bradley R. Schmerl, and David Garlan. 2020. Software architecture and task plan co-adaptation for mobile service robots. In *SEAMS '20: IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Seoul, Republic of Korea, 29 June - 3 July, 2020*, Shinichi Honiden, Elisabetta Di Nitto, and Radu Calinescu (Eds.). ACM, 125–136.

[11] Javier Cámara, Javier Troya, Antonio Vallecillo, Nelly Bencomo, Radu Calinescu, Betty H. C. Cheng, David Garlan, and Bradley Schmerl. 2022. The uncertainty interaction problem in self-adaptive systems. *Software and Systems Modeling* 21, 4 (01 Aug 2022), 1277–1294. https://doi.org/10.1007/s10270-022-01037-6

[12] Marc Carwehl, Calum Imrie, Thomas Vogel, Genaína Rodrigues, Radu Calinescu, and Lars Grunske. 2024. Formal Synthesis of Uncertainty Reduction Controllers. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 2–13.

[13] Xi Chen, Peng Li, and Yongle Wu. 2020. Task scheduling for cyber-physical systems under uncertainty: A survey. *IEEE Transactions on Emerging Topics in Computing* 8, 3 (2020), 593–608.

[14] Shang-Wen Cheng, Vahe Poladian, David Garlan, and Bradley R. Schmerl. 2009. Improving Architecture-Based Self-Adaptation through Resource Prediction. In *Software Engineering for Self-Adaptive Systems [outcome of a Dagstuhl Seminar] (Lecture Notes in Computer Science, Vol. 5525)*, Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee (Eds.). Springer, 71–88.

[15] Daniel Claes, Frans Oliehoek, Hendrik Baier, and Karl Tuyls. 2017. Decentralised online planning for multi-robot warehouse commissioning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 492–500.

[16] Deshan Cooray, Ehsan Kouroshfar, Sam Malek, and Roshanak Roshandel. 2013. Proactive self-adaptation for improving the reliability of mission-critical, embedded, and mobile software. *IEEE Transactions on Software Engineering* 39, 12 (2013), 1714–1735.

[17] R. Pisani D. Freedman and R.Purves. 2007. *Statistics* (4th ed ed.). Norton, London,U.K.

[18] Bruno de Finetti. 2017. *Theory of Probability: A critical introductory treatment*. John Wiley & Sons. https://doi.org/10.1002/9781119286387

[19] Naeem Esfahani and Sam Malek. 2013. *Uncertainty in Self-Adaptive Software Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 214–238. https://doi.org/10.1007/978-3-642-35813-5_9

[20] Gianluca Filippone, Juan Antonio Piñera García, Marco Autili, and Patrizio Pelliccione. 2024. Handling uncertainty in the specification of autonomous multi-robot systems through mission adaptation. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 25–36.

[21] Sebastian Hahner, Robert Heinrich, and Ralf Reussner. 2023. Architecture-Based Uncertainty Impact Analysis to Ensure Confidentiality. In *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 126–132. https://doi.org/10.1109/SEAMS59076.2023.00026

[22] Sara M. Hezavehi, Danny Weyns, Paris Avgeriou, Radu Calinescu, Raffaela Mirandola, and Diego Perez-Palacin. 2021. Uncertainty in Self-adaptive Systems: A Research Community Perspective. *ACM Trans. Auton. Adapt. Syst.* 15, 4, Article 10 (dec 2021), 36 pages. https://doi.org/10.1145/3487921

[23] Daniel Jackson. 2019. Alloy: a language and tool for exploring software designs. *Commun. ACM* 62, 9 (2019), 66–76.

[24] Brian LaMacchia Ufuk Topcu Jensen, David and Pamela Wisniewski. 2023. Algorithmic Robustness. *ACM SIGSOFT Software Engineering Notes* (2023).

[25] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2007. Stochastic Model Checking. In *Formal Methods for Performance Evaluation, 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2007, Bertinoro, Italy, May 28-June 2, 2007, Advanced Lectures (Lecture Notes in Computer Science, Vol. 4486)*, Marco Bernardo and Jane Hillston (Eds.). Springer, 220–270. https://doi.org/10.1007/978-3-540-72522-0_6

[26] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6806)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.). Springer, 585–591.

[27] Annu Lambora, Kunal Gupta, and Kriti Chopra. 2019. Genetic algorithm-A literature review. In *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE, 380–384.

[28] Axel Legay, Anna Lukina, Louis Marie Traonouez, Junxing Yang, Scott A Smolka, and Radu Grosu. 2019. Statistical model checking. In *Computing and software science: state of the art and perspectives*. Springer, 478–504.

[29] Xiao-Fang Liu, Yongchun Fang, Zhi-Hui Zhan, and Jun Zhang. 2023. Strength learning particle swarm optimization for multiobjective multirobot task scheduling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 53, 7 (2023), 4052–4063.

[30] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns. 2017. Chapter 3 - A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements. In *Managing Trade-Offs in Adaptable Software Architectures*, Ivan Mistrik, Nour Ali, Rick Kazman, John Grundy, and Bradley Schmerl (Eds.). Morgan Kaufmann, Boston, 45–77. https://doi.org/10.1016/B978-0-12-802855-1.00003-4

[31] Sara Mahdavi-Hezavehi, Paris Avgeriou, and Danny Weyns. 2017. *A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements*. Morgan Kaufmann, Chapter 3, 45 – 77.

[32] Claudio Menghi, Sergio García, Patrizio Pelliccione, and Jana Tumova. 2018. Multi-robot LTL Planning Under Uncertainty. In *Formal Methods - 22nd International Symposium, FM 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 15-17, 2018, Proceedings*. 399–417.

[33] Branko Miloradovic, Baran Çürüklü, Mikael Ekström, and Alessandro Vittorio Papadopoulos. 2022. GMP: A Genetic Mission Planner for Heterogeneous Multirobot System Applications. *IEEE Trans. Cybern.* 52, 10 (2022), 10627–10638.

[34] Branko Miloradovic, Baran Çürüklü, Mikael Ekström, and Alessandro Vittorio Papadopoulos. 2023. Optimizing Parallel Task Execution for Multi-Agent Mission Planning. *IEEE Access* 11 (2023), 24367–24381.

[35] Gabriel A. Moreno, Javier Cámara, David Garlan, and Mark Klein. 2018. Uncertainty reduction in self-adaptive systems. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2018, Gothenburg, Sweden, May 28-29, 2018*, Jesper Andersson and Danny Weyns (Eds.). ACM, 51–57. https://doi.org/10.1145/3194133.3194144

[36] Gabriel A. Moreno, Javier Cámara, David Garlan, and Bradley R. Schmerl. 2015. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, Elisabetta Di Nitto, Mark Harman, and Patrick Heymans (Eds.). ACM, 1–12.

[37] William L. Oberkampf, Sharon M. DeLand, Brian M. Rutherford, Kathleen V. Diegert, and Kenneth F. Alvin. 2002. Error and uncertainty in modeling and simulation. *Reliability Engineering & System Safety* 75, 3 (2002), 333–357. https://doi.org/10.1016/S0951-8320(01)00120-X

[38] Diego Perez-Palacin and Raffaela Mirandola. 2014. Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation. In *Proc. of ICPE'14*. ACM, 3–14.

[39] Andres J. Ramirez, Adam C. Jensen, and Betty H. C. Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *Proc. of SEAMS'12*. IEEE Computer Society, 99–108.

[40] Marvin Rausand. 2013. *Risk Assessment: Theory, Methods, and Applications*. John Wiley & Sons.

[41] Stuart J. Russell and Peter Norvig. 2010. *Artificial Intelligence, A Modern Approach* (3 ed.). Prentice Hall.

[42] Raquel Sanchez, Javier Troya, and Javier Camara. 2024. Automated Planning for Adaptive Cyber-Physical Systems under Uncertainty in Temporal Availability Constraints. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (Lisbon, AA, Portugal) *(SEAMS '24)*. Association for Computing Machinery, New York, NY, USA, 14–24. https://doi.org/10.1145/3643915.3644083

[43] Andrew JE Seely and Peter T Macklem. 2004. Complex systems and the technology of variability analysis. *Critical Care* 8 (2004), 367–384. https://doi.org/10.1186/cc2948

[44] Raquel Sánchez-Salas, Javier Troya, and Javier Cámara. 2024. Automated Planning Under Interacting Sources of Uncertainty. https://github.com/atenearesearchgroup/Automated-Planning-Under-Interacting-Sources-of-Uncertainty..

[45] Daniel P. Thunnissen. 2003. Uncertainty classification for the design and development of complex systems. In *Proc. of the 3rd Annual Predictive Methods Conference, Veros Software*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.128.133

[46] Javier Troya, Nathalie Moreno, Manuel F. Bertoa, and Antonio Vallecillo. 2021. Uncertainty representation in software models: a survey. *Software & Systems Modeling* 20 (2021), 1183–1213. https://doi.org/10.1007/s10270-020-00842-1

[47] Gricel Vázquez, Radu Calinescu, and Javier Cámara. 2022. Scheduling of Missions with Constrained Tasks for Heterogeneous Robot Systems. In *Proceedings Fourth International Workshop on Formal Methods for Autonomous Systems (FMAS) and Fourth International Workshop on Automated and verifiable Software sYstem DEvelopment (ASYDE),*

FMAS/ASYDE@SEFM 2022, and Fourth International Workshop on Automated and verifiable Software sYstem DEvelopment (ASYDE)Berlin, Germany, 26th and 27th of September 2022 (EPTCS, Vol. 371), Matt Luckcuck and Marie Farrell (Eds.). 156–174.

[48] Shanu Verma, Millie Pant, and Vaclav Snasel. 2021. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. *Ieee Access* 9 (2021), 57757–57791.

[49] Norha M. Villegas, Hausi A. Müller, Gabriel Tamura, Laurence Duchien, and Rubby Casallas. 2011. A framework for evaluating quality-driven self-adaptive software systems. In *2011 ICSE Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2011, Waikiki, Honolulu , HI, USA, May 23-24, 2011*, Holger Giese and Betty H. C. Cheng (Eds.). ACM, 80–89.

[50] Min Wang, Guoshan Liu, and Xinyu Lin. 2022. Dynamic Optimization of the Multi-Skilled Resource-Constrained Project Scheduling Problem with Uncertainty in Resource Availability. *Mathematics* 10, 17 (2022).

[51] Danny Weyns. 2020. *An introduction to self-adaptive systems: A contemporary software engineering perspective.* John Wiley & Sons.

[52] Danny Weyns, Radu Calinescu, Raffaela Mirandola, Kenji Tei, Maribel Acosta, Nelly Bencomo, Amel Bennaceur, Nicolas Boltz, Tomas Bures, Javier Camara, et al. 2023. Towards a research agenda for understanding and managing uncertainty in self-adaptive systems. *ACM SIGSOFT Software Engineering Notes* 48, 4 (2023), 20–36.

[53] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, and Björn Regnell. 2012. *Experimentation in Software Engineering.* Springer.

[54] Man Zhang, Shaukat Ali, Tao Yue, and Bran Selic. 2016. Understanding uncertainty in cyber-physical systems: a conceptual model. In *Modelling Foundations and Applications: 12th European Conference, ECMFA 2016, Held as Part of STAF 2016, Vienna, Austria, July 6-7, 2016, Proceedings.* Springer, 247–264.

[55] Hans-Jürgen Zimmermann. 2001. *Fuzzy Set Theory – and Its Applications.* Springer Science+Business Media.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009