Automated Planning for Adaptive Cyber-Physical Systems under Uncertainty in Temporal Availability Constraints

Raquel Sánchez-Salas, Javier Troya, and Javier Cámara
{raquelsalanchez,jtroya,jcamara}@uma.es
ITIS Software – Universidad de Málaga
Málaga, Spain

ABSTRACT

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 deal with different classes of constraints (e.g., ordering, structural) and uncertainties (e.g., in sensing, actuation, latencies). However, planning under temporal availability constraints, i.e., when a given resource or other element of the system required to perform a task is available only during a limited and variable time window, is a challenge that remains largely unexplored. This paper presents an approach to address this challenge, employing genetic algorithms to incorporate temporal uncertainties effectively. Our method demonstrates enhanced robustness and efficiency in task-based sCPS scenarios with variable time windows, such as electric vehicle charging and healthcare robotics. Our evaluation shows that our approach significantly reduces computational cost and maintains solution feasibility under prescribed levels of uncertainty, outperforming MILP-based optimization.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

planning, adaptation, uncertainty, availability constraints, CPS

ACM Reference Format:
Raquel Sánchez-Salas, Javier Troya, and Javier Cámara. 2024. Automated Planning for Adaptive Cyber-Physical Systems under Uncertainty in Temporal Availability Constraints. In *19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '24), April 15–16, 2024, Lisbon, AA, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3643915.3644083

1 INTRODUCTION

Self-adaptive, smart Cyber-Physical Systems (sCPS) are becoming increasingly prevalent in various domains, including healthcare, transportation, and industrial automation, to name a few. These systems are designed to operate in dynamic and uncertain environments, making decisions and adapting their behavior in response to changes in their context and requirements. However, the complexity and uncertainty inherent in these environments pose significant challenges for the effective and efficient operation of sCPS. In consequence, in recent years there has been 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 [3, 5, 9, 15].

Among the existing approaches to automated planning under uncertainty in sCPS, a notable limitation is the inability to consider uncertainty in temporal availability constraints. 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. Current methods often struggle to effectively incorporate this type of uncertainty, and this can potentially lead to suboptimal or even unfeasible plans that can negatively impact the performance and reliability of the system.

To address this gap, in this paper we contribute a novel approach to automated planning that explicitly addresses the uncertainty in temporal availability constraints in task-based sCPS scenarios. Our method is specifically designed to handle in a robust manner uncertainties about e.g., the exact arrival and departure times of an electric vehicle that needs to be charged within a specific time window. To achieve this, our approach makes use of genetic algorithms and is able to efficiently explore 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).

To evaluate our approach, we focus on two research questions that concern the effectiveness and efficiency of our approach:

(**RQ1**): **Effectiveness**. We formulate two different subquestions:
(RQ1.1) How effective is our approach in comparison to an optimal algorithm?
(RQ1.2) How effective is our approach in comparison to an optimal algorithm under prescribed levels of uncertainty in temporal availability constraints?

(**RQ2**): **Efficiency**. How efficient is our approach with respect to an optimal algorithm?

We evaluate our approach on two case studies corresponding to distinct domains: a shared smart infrastructure for the charging of electric vehicles, and healthcare robotics. Our results demonstrate a remarkable reduction in computational cost for producing solutions using our proposed approach, with only a marginal loss in optimality when compared to a baseline method to solve the

SEAMS '24, April 15–16, 2024, Lisbon, AA, Portugal

ACM ISBN 979-8-4007-0585-4/24/04…$15.00
https://doi.org/10.1145/3643915.3644083

planning problem that employs mixed-integer linear programming (MILP), which guarantees finding a globally optimal solution. Notably, while there is a slight compromise in optimality, the solutions generated by our approach exhibit greater robustness under uncertainty, in contrast to those produced by the baseline method, which significantly degrade as the level of uncertainty increases. This inherent robustness of our approach, coupled with its computational efficiency, makes it a promising solution to incorporate e.g., into the planning stage of MAPE-K [10] sCPS where uncertainty in temporal availability constraints is one of the primary factors affecting the feasibility and quality of the adaptation plans generated.

In the remainder of this paper, Section 2 introduces the motivating scenarios for our approach. Section 3 formalizes the problem. Next, Section 4 introduces our approach and Section 5 describes its evaluation, including threats to validity. Section 6 discusses related work. Finally, Section 7 concludes the paper and presents some directions for future work.

2 MOTIVATING SCENARIOS

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), 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.

Figure 1: Electric vehicle charging infrastructure at the Ada Byron research building, University of Málaga.

To motivate our approach, we describe a scenario involving a parking lot at the Ada Byron research building in the University of Málaga, 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. 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.

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. 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.

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 3.1 (Availability window). We define the availability window of a CPS element $e \in E = \{R \cup C\}$ as $W : E \rightarrow \mathbb{R}_0^+ \times \mathbb{R}^+$,

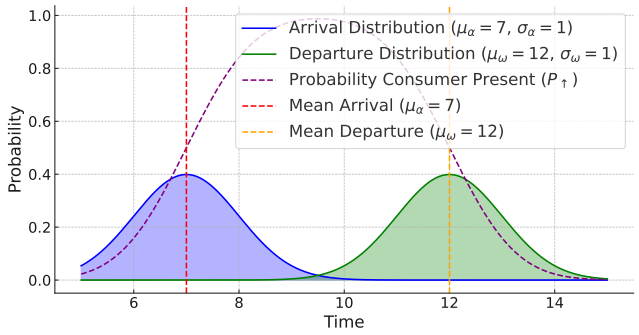

Figure 2: Example of probability availability function for an electric vehicle.

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 3.2 (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:

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

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 3.3. 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 3.4 (Problem instance). A problem instance is characterized by a tuple $(R, C, F_{\uparrow E}, T, \Lambda, \Theta, O)$, 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, T is a set of tasks, $\Lambda : T \to C$ 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 $O : S_v \to \mathbb{R}^k$ is a function that maps solutions (cf. Definition 3.6) of a problem instance v to k optimization criteria.

Definition 3.5 (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 3.6 (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 3.7 (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 $S_v^\star$:

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

4 APPROACH

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.

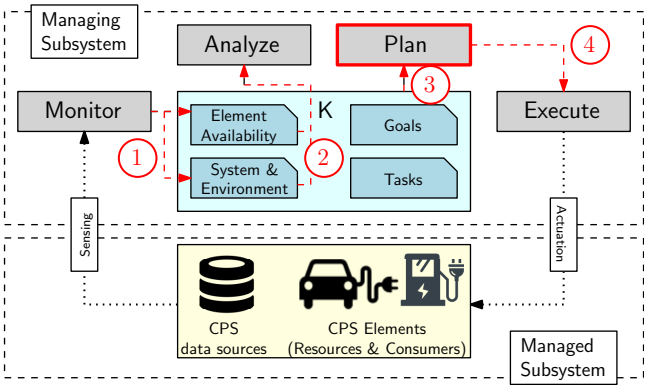

Figure 3: Run-time context overview of our approach.

4.0.1 Knowledge. Incorporates various models that contain information about the tasks to be performed, system goals (e.g., optimization objectives – cf. O in Definition 3.7), resource and consumer availability – cf. Definition 3.1), 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.

4.0.2 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.

4.0.3 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.

4.0.4 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 3.4), 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.

4.0.5 Execution. Once a plan is generated ④ (cf. Definition 3.6), 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 4.1), and follow with a description of our genetic algorithm to solve the problem under uncertainty in temporal availability constraints (Section 4.2).

4.1 Planning Subject to Well-Known Temporal Availability Constraints

We present in this section an algorithm that solves the planning problem with well-defined temporal availability constraints (cf. Definition 3.1) in an optimal fashion.

Algorithm 1 provides as output a problem solution as a set of resource assignments according to Definition 3.6, and receives as input a deterministic problem instance, which is equivalent to the one described in Definition 3.4, but including deterministic availability windows of resources and consumers, instead of their availability probability functions:

Definition 4.1 (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 4.2 (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.

Algorithm 1: Planning with well known temporal availability constraints (Mixed-Integer Linear Programming).

```
1  Input: Deterministic problem instance
     i = (R, C, W, T, Λ, Θ, O) (cf. Definition 4.1).
2  Output: A problem solution assigning task to resources and
     consumers (cf. Definition3.6).
3  γ ← (X = ∅, O = i.O, ρ = ∅)
4  γ.X ← {(c, r, 0) ↦ 0 | c ∈ i.C, r ∈ i.R}
5  γ.ρ ← {∀x, x′ ∈ γ.X • x.r = x′.r ⇔ (x = x′ ∨ x.s ≠ x′.s)}
6  foreach r ∈ i.R do
7      foreach c ∈ i.C do
8          foreach s ∈ {1, ..., ⌈Θ / t_g⌉} do
9              γ.ρ ← γ.ρ ∪ {γ.X(c, r, s) = 1 ⇒ s ∈
                 slots(i.W(r)) ∩ slots(i.W(c))}
10         end
11     end
12 end
13 solve(γ)
14 A ← ∅
15 foreach x ∈ γ.X do
16     if ∄ a ∈ A • a.r = x.r ∧ a.t = Λ^{-1}(x.c) then
17         A ← A ∪ {(x.r, Λ^{-1}(x.c), [x.s · t_g, (x.s + 1) · t_g])}
18     end
19     else
20         a.θ_s ← min(a.θ_s, x.s · t_g)
21         a.θ_f ← max(a.θ_f, x.s · t_g)
22     end
23 end
24 return A
```

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 4.2), 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 3.5 and 3.6). To achieve this, an initially empty set of assignments A is created (line 14), and then progressively extended by iterating over all the 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.

4.2 Planning under Uncertainty in Temporal Availability Constraints

Once we have presented how to solve a simplified version of our problem in which temporal availability constraints are well known, in this section we describe a genetic algorithm capable of providing solutions under uncertainty in temporal availability constraints.

Algorithm 2 provides as output a problem solution as a set of resource assignments according to Definition 3.6, and receives as input a problem instance (cf. Definition 3.4), as well as a set of parameters to control the behavior of the genetic algorithm (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), 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 $[min(f_{t_r^{\alpha}}.\mu, f_{t_c^{\alpha}}.\mu), max(f_{t_r^{\omega}}.\mu, f_{t_c^{\omega}}.\mu)]$. 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).

Once the initial population has been built, the algorithm iterates over a maximum number of generations g_{max} (lines 15-27) by evaluating the current population of assignments (lines 16-24), and then selecting the parents (line 25) to generate the offspring for the next generation (line 26 crossover, and line 27 mutation). We provide details about these stages of the algorithm in the following.

Algorithm 2: Planning under uncertainty in temporal availability constraints (Genetic Algorithm).

1 **Inputs:** Problem instance $i = (R, C, F_{\uparrow E}, T, \Lambda, \Theta, O)$ (cf. Definition 3.4); 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 κ.
2 **Output:** A problem solution assigning task to resources and consumers (cf. Definition3.6).
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 **foreach** $gen \in \{1, \ldots, g_{max}\}$ **do**
16 $FP \leftarrow \{p \mapsto \langle 0 \rangle^k \mid p \in P\}$
17 **foreach** $p \in P$ **do**
18 $FP_{\kappa} \leftarrow \emptyset$
19 **for** $samp \in \{1, \ldots, \kappa\}$ **do**
20 $p_{\kappa} \leftarrow generate_{\kappa}(p)$
21 $FP_{\kappa} \leftarrow FP_{\kappa} \cup \{p_{\kappa} \mapsto i.O(p_{\kappa})\}$
22 **end**
23 $FP[p] \leftarrow mean(FP_{\kappa})$
24 **end**
25 $P_{sel} \leftarrow selection(FP)$
26 $P \leftarrow crossover(P_{sel})$
27 $P \leftarrow mutation(P)$
28 **end**
29 **return** $\{p \in P : i.O(p) \in Pareto(FP)\}$

Concerning evaluation of individuals in the population (lines 15-27), 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 ($f_{t^{\alpha}}$) and end ($f_{t^{\omega}}$) 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).

When each of the κ samples are generated for each individual p, their fitness along the k optimization objectives is stored in FP_κ (line 21). 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 23).

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 25). Then, the algorithm generates the next population by applying the $crossover : A \times A \rightarrow A \times A$ operator (line 26), 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 27), 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.:

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

Finally, the algorithm finishes by returning the set of non-dominated solutions in the population (line 29 – cf. Definition 3.7).

5 EVALUATION

We evaluate our approach with the objective of assessing its *effectiveness* both under well-known temporal availability constraints (no uncertainty, **RQ1.1**) and prescribed levels of uncertainty (**RQ1.2**), as well as its *efficiency* (**RQ2**).

5.1 Experimental Setup

To obtain the data for our evaluation, we performed experiments that compare our GA-based solution (cf. Section 4.2) with an implementation of the MILP-based solution described in Section 4.1, which we employ as baseline. 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 can be downloaded from https://github.com/atenearesearchgroup/Automated-Planning-for-Adaptive-CPS.

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. 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.

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.)

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 10 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? To answer this question, we performed a set of experiments in which the results of the algorithm 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. Figure 4 illustrates the results that compare the effectiveness of our GA-based approach with the MILP baseline with no uncertainty in temporal availability constraints. One of the first things that can be observed 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 that belong to the GA). One notable exception to the dominant trend of the MILP is the disruption metric both in two of the four problem instances of RoboMax, 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 in temporal availability constraints, an optimal 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 4.2).

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 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. 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.

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.

(RQ2) ***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. Table 2 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.

Table 2: Planning execution times: GA vs MILP.

EV Charging Infrastructure				
$\|R\|$	$\|C\|$	**GA Time (ms)**	**MILP Time (ms)**	**ΔTime (%)**
2	5	32.5 ± 134.2	28.42 ± 157.46	-12.55
2	20	278.52 ± 238.76	289.36 ± 445.6	+3.88
4	40	543.57 ± 428.8	1925.59 ± 356.82	+254.77
8	80	1668.64 ± 465.76	6678.31 ± 458.42	+300.67
16	160	3564.76 ± 396.75	15248.97 ± 508.81	+327.60
Robomax: Food Logistics				
$\|R\|$	$\|C\|$	**GA Time (ms)**	**MILP Time (ms)**	**ΔTime (%)**
2	5	46.7 ± 128.2	50.31 ± 148.35	-7.75
2	20	325.86 ± 456.85	456.82 ± 518.32	+40.17
4	40	642.48 ± 455.74	1895.43 ± 509.82	+195.83
8	80	1789.54 ± 406.57	6234.75 ± 465.58	+248.39
16	160	3668.58 ± 519.92	14824.53 ± 564.39	+311.1

Concluding remarks on RQ2: 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

We discuss four types of threats to validity in the work presented in this paper. **Construct validity:** We designed our approach with uncertainty in temporal availability constraints in mind. However, we are aware that there are other sources of uncertainty that might affect the effectiveness of our approach [13, 18, 19] (e.g., uncertainties due to model abstraction, imperfect sensing and actuation). Hence, we will perform further experiments in future work, in which we will incorporate additional sources of uncertainty and explore their interaction [6] with uncertainty in temporal availability constraints. **Internal validity:** We compared our genetic algorithm with a MILP baseline. We acknowledge that there are other alternatives, such as well-known Multi-Objective Genetic Algorithms (MOGAs) like NSGA-II [21], or probabilistic and statistical model checking [11, 12], 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. **External validity:** 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

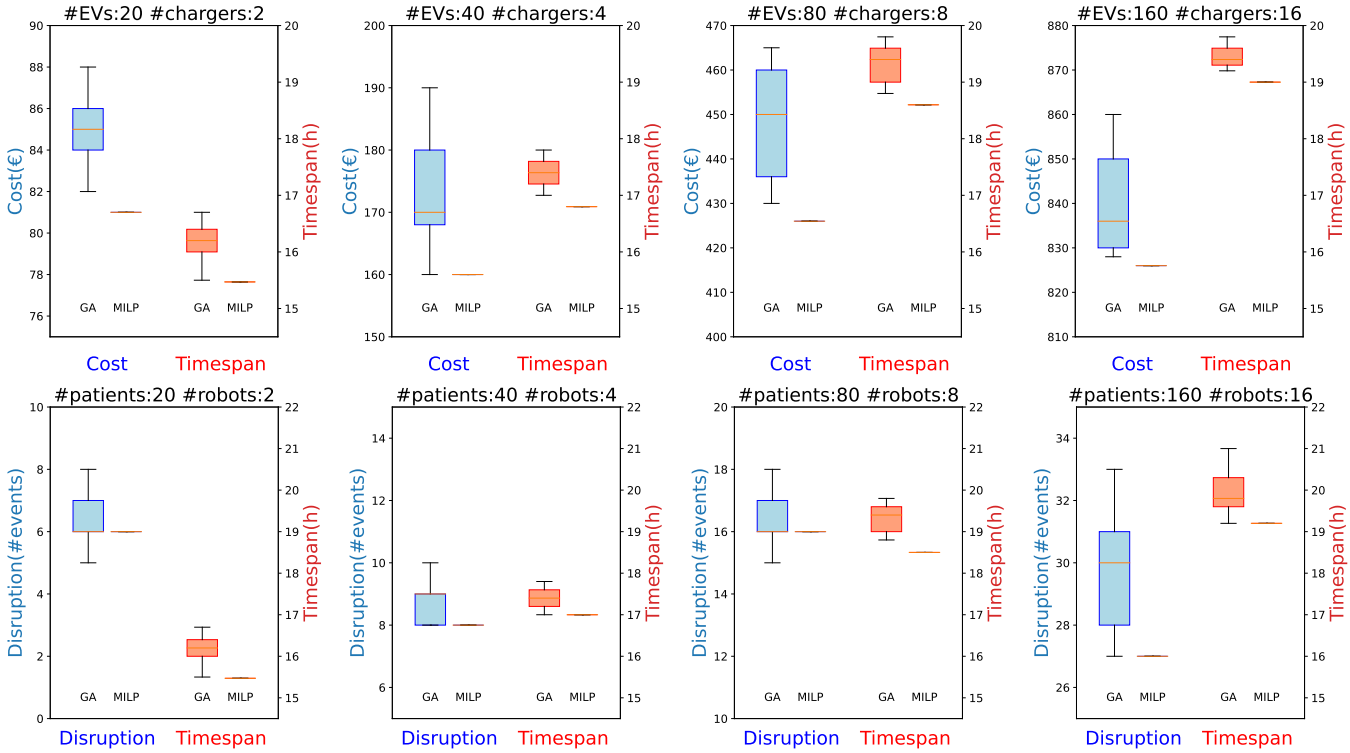

Figure 4: Effectiveness of our genetic algorithm-based approach vs MILP baseline (no uncertainty in temporal availability constraints): EV charging infrastructure (top) and RoboMax food logistics (bottom).

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). **Conclusion validity:** 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. [4, 22]. 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

The number of approaches for planning under uncertainty has recently experienced substantial growth in multiple areas. Although we cannot give an exhaustive account of existing approaches, we summarize in this section some that are relevant to our proposal:
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. [23], 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 [7] early work proposes resource availability prediction as a means to achieve proactive adaptation, a direction that was later followed by other authors (e.g., [9, 17]), but without support for explicit resource availability constraints. DECIDE [3] 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. [5] 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. **Multi-agent and multi-robot systems**. The Genetic Mission Planner (GMP) [15] casts a mission for heterogeneous multi-agent

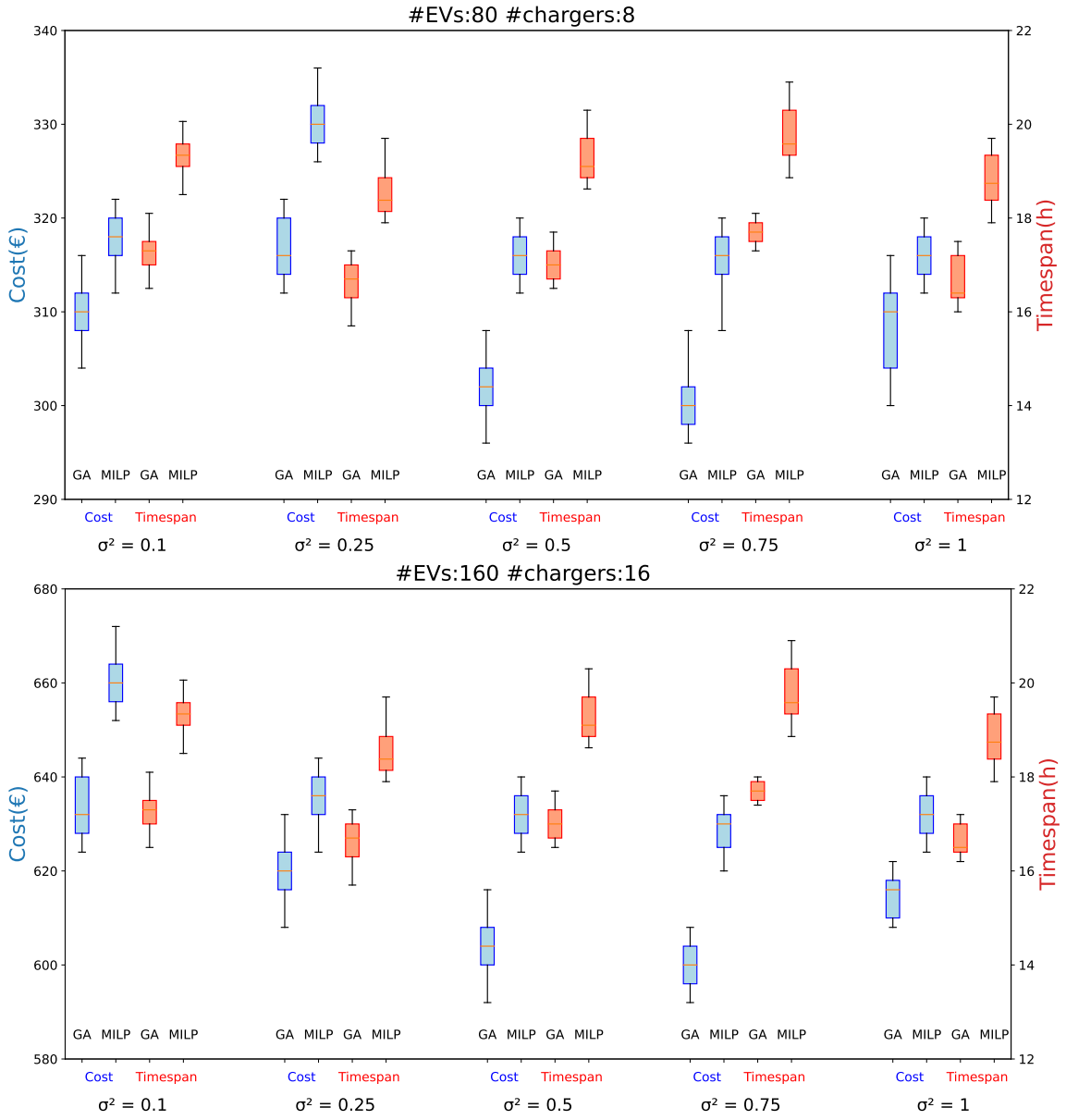

Figure 5: Effectiveness of our genetic algorithm-based approach vs MILP baseline with uncertainty in temporal availability constraints: EV charging infrastructure.

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 [16], 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 [5]. KANOA [20] 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 [14] 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. [8] 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.

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 and self-adaptive systems areas is equipped to consider uncertainty in temporal availability constraints.

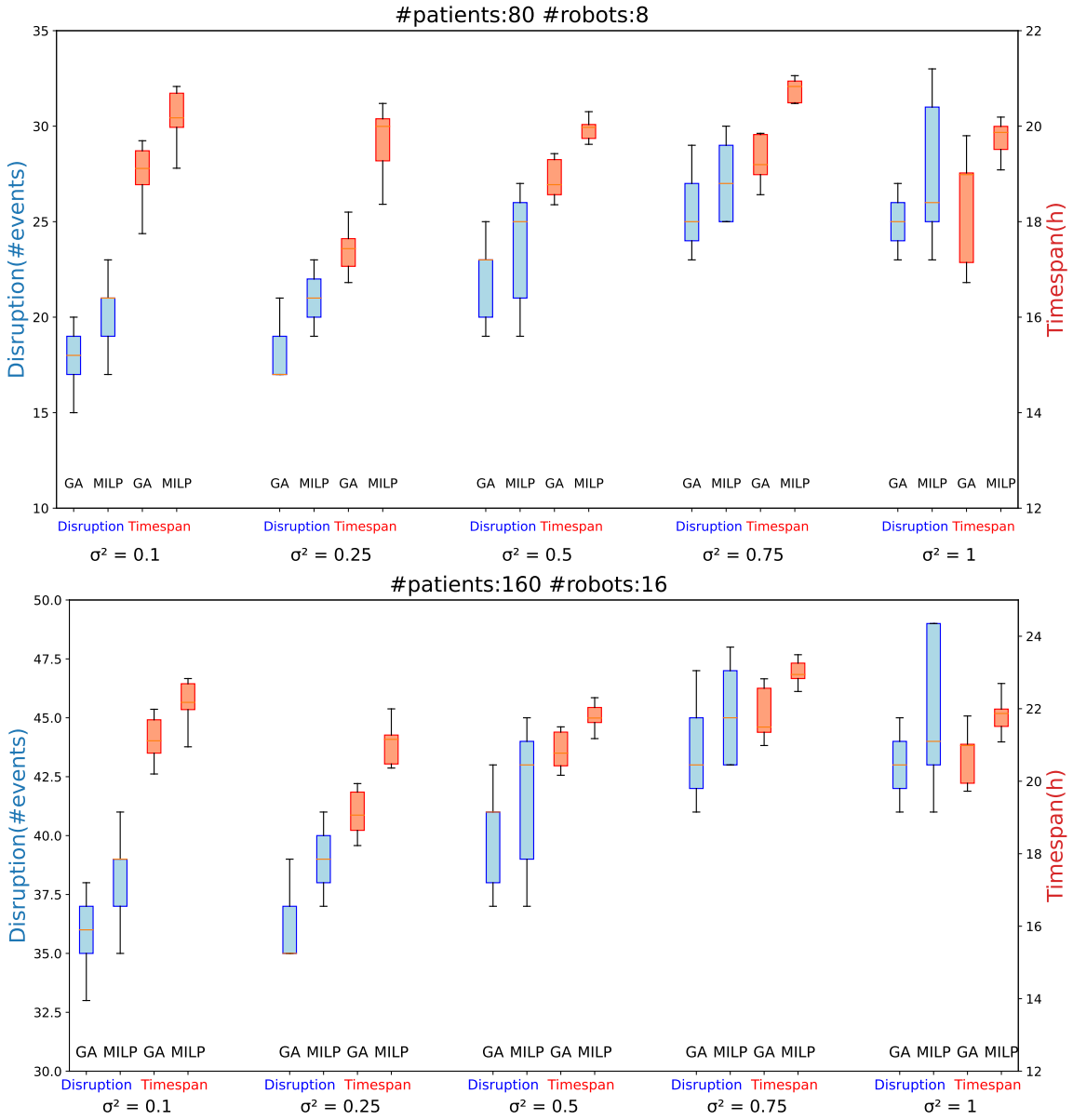

Figure 6: Effectiveness of our genetic algorithm-based approach vs MILP baseline with uncertainty in temporal availability constraints: RoboMax Food Logistics.

7 CONCLUSIONS AND FUTURE WORK

In this article, we have presented our approach to planning under uncertainty in temporal availability constraints for sCPS. Our results have shown the applicability of our approach to two different domains, as well as its efficiency and effectiveness, compared to an alternative solution based on mixed-integer linear programming.

In the current version of our approach, availability windows are restricted to one per CPS element. In future work, we plan on extending our approach to consider multiple availability windows. Moreover, we plan on incorporating additional sources of uncertainty such as reliability in task performance (e.g., robots may be able to deliver food only with some probability of success, chargers may malfunction during charging), and explore additional solution methods, such as probabilistic model checking, to furnish the approach with quantitative guarantees. Finally, we plan on embedding our approach in physical testbeds to evaluate it in a broader context of adaptation with humans-in-the-loop.

ACKNOWLEDGEMENTS

Work partially funded by the Spanish Government (FEDER, Ministerio de Ciencia e Innovación–Agencia Estatal de Investigación) under projects TED2021-130523B-I00 and PID2021-125527NB-I00.

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] 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.

[4] 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.

[5] 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.

[6] Javier Cámara, Javier Troya, Antonio Vallecillo, Nelly Bencomo, Radu Calinescu, Betty H. C. Cheng, David Garlan, and Bradley R. Schmerl. 2022. The uncertainty interaction problem in self-adaptive systems. *Softw. Syst. Model.* 21, 4 (2022), 1277–1294.

[7] 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.

[8] 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.

[9] 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.

[10] Jeffrey O. Kephart and David M. Chess. 2003. The Vision of Autonomic Computing. *Computer* 36, 1 (2003), 41–50. https://doi.org/10.1109/MC.2003.1160055

[11] 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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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).