The published version of this paper is available from
https://dl.acm.org/doi/pdf/10.1145/3643915.3644081

Towards Proactive Decentralized Adaptation of Unmanned Aerial Vehicles for Wildfire Tracking

Enrique Vilchez, Javier Troya, and Javier Cámara
{enriquev,jtroya,jcamara}@uma.es
ITIS Software – Universidad de Málaga
Málaga, Spain

ABSTRACT

Smart Cyber-Physical Systems (sCPS) operate in dynamic and uncertain environments, where anticipation to adverse situations is crucial and decentralization is often necessary due to e.g., scalability issues. Addressing the limitations related to the lack of foresight of (decentralized) reactive self-adaptation (e.g., slower response, suboptimal resource usage), this paper introduces a novel method that employs Predictive Coordinate Descent (PCD) to enable decentralized proactive self-adaptation in sCPS. Our study compares PCD with a reactive Deep Q-Network (DQN) strategy on Unmanned Aerial Vehicles (UAV) in wildfire tracking adaptation scenarios. Results show how PCD outperforms DQN when furnished with high-quality predictions of the environment, but progressively degrades in effectiveness with predictions of decreasing quality.

CCS CONCEPTS

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

KEYWORDS

UAV, proactive adaptation, predictive coordinate descent

ACM Reference Format:
Enrique Vilchez, Javier Troya, and Javier Cámara. 2024. Towards Proactive Decentralized Adaptation of Unmanned Aerial Vehicles for Wildfire Tracking. 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, 7 pages. https://doi.org/10.1145/3643915.3644081

1 INTRODUCTION

Motivated by the need of Cyber-Physical Systems (CPS) to operate in complex domains where they must continually adjust to various sources of uncertainty, smart CPS (sCPS) are endowed with adaptive capabilities and are often required to make crucial decisions based on incomplete or imperfect information. For instance, autonomous vehicles navigate in environments where they cannot always foresee the actions of other drivers, smart grids must efficiently distribute energy despite unpredictable fluctuations in demand, and in environmental contexts, Unmanned Aerial Vehicles (UAV) monitoring forest areas for wildfire tracking must deal with limited visibility due to obstructions like smoke.

In these scenarios, sCPS often operate with decentralized decision-making, which offers advantages over centralized approaches in terms of scalability and resilience, allowing systems to manage a growing number of components without a centralized bottleneck and mitigating the overall impact of failure of a single node. Moreover, decentralized adaptation reduces latency in decision-making, as decisions are made locally based on the immediate data available.

Despite these advantages, current decentralized decision-making approaches in sCPS (e.g., [4, 6, 18]) are predominantly reactive and focus on responding to changes as they occur. Reactive adaptation, while simpler to implement, has notable disadvantages compared to proactive adaptation [20, 21]: Reactive systems often lack the foresight needed for anticipatory responses, leading to less efficient resource utilization and potentially slower adaptation to changes, and may also struggle in maintaining performance and reliability, as they are constantly adapting to, rather than preparing for, changes.

While proactive adaptation offers significant advantages, its implementation in decentralized decision-making settings is often challenged by factors like limited global knowledge, or coordination complexity, which is significantly more challenging than in reactive adaptation, as it is based on predictions about the future instead of known current conditions of the environment.

This paper aims at bringing together the best of both worlds by contributing: (i) a problem to assess adaptation in distributed sCPS that incorporates novel features to challenge adaptation strategies (e.g., partial observability of the UAV's environment induced by smoke); (ii) an approach to decentralized decision-making for proactive self-adaptation in sCPS that makes a novel use of Predictive Coordinate Descent (PCD [22] – a computational method for large-scale optimization problems) to produce adaptation decisions for UAV teams; (iii) a customizable simulator (Wildfire-UAVSim) that features interactive simulation and batch experimentation, and (iv) a study that compares our PCD approach with our own extended and adapted version of a state-of-the-art reactive approach based on Deep-Q-Networks (DQN) that solves an analogous adaptation problem [12]. The comparison is made both under standard operational conditions, as well as under high levels of uncertainty induced by the limited observability of the environment.
Our study explores the following research questions:

(**RQ1**): ***Effectiveness.*** How effective is our PCD-based approach in comparison to a reactive (DQN-based) approach?
(RQ1.1) How effective is PCD compared to DQN under optimal predictive conditions?
(RQ1.2) How effective is PCD compared to DQN under increasingly high levels of degradation in predictions?
(**RQ2**): ***Efficiency.*** How do the computational costs of PCD vs DQN compare and affect their overall performance?

Results show how: (i) PCD outperforms DQN when furnished with high-quality predictions both under standard conditions, as well as in challenging scenarios; (ii) PCD progressively degrades in effectiveness when provided with lower-quality predictions; and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SEAMS '24, April 15–16, 2024, Lisbon, AA, Portugal
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0585-4/24/04…$15.00
https://doi.org/10.1145/3643915.3644081

(iii) PCD is less efficient than DQN, although decision times are acceptable for the domain and indicate potential for scalability.

2 MOTIVATING SCENARIO

A team of UAV tracks the evolution of a wildfire as it spreads over a forest area. The spread of the fire must be tracked as accurately as possible by maximizing the amount of up-to-date information monitored of the fire front (i.e., areas that have already burnt or are not close to the fire yet are not relevant). Each UAV is equipped with a downwards-facing camera that captures information about the forest surface. UAV also have to conduct the mission in a safe manner by minimizing the chance of collisions among them, and are equipped with sensors that enable them to estimate the position of other UAV. Moreover, UAV may have to operate under challenging conditions that involve strong wind and smoke, which limits their ability to perceive their surrounding forest area, as well as other UAV. We formalize our scenario in the remainder of this section.

2.0.1 Forest Area and Wildfire Propagation. A forest area is represented as a grid S of $N \times N$ cells, where each cell $s \in S$ incorporates two time-dependent variables that represent: (i) the amount of burnable fuel in the cell $F : S \rightarrow \mathbb{N}$; and (ii) whether the cell is burning ($B : S \rightarrow \{0, 1\}$). We assume a discrete timeline, and for simplicity, we represent the fuel value of a cell s at time instant t as $F_t(s)$ ($B_t(s)$ for B, respectively). A cell can change when one time unit elapses: (a) part of its fuel is consumed (according to a burning rate β – cf. Expression 1) if the cell is burning, (b) when $F_t(s) = 0$, then the fire is extinguished in that cell ($B_{t+1}(s) = 0$), and (c) when there is fuel remaining in a cell that is not burning ($F_t(s) > 0 \wedge B_t(s) = 0$), we define a probability $p_t(s)$ (Expression 2) of the cell getting ignited based on the proximity to other burning cells.

$$F_{t+1}(s) = \begin{cases} max(0, F_t(s) - \beta) & \text{if } B_t(s) \\ F_t(s) & \text{otherwise} \end{cases} \quad (1)$$

$$p_{t+\mho}(s) = \begin{cases} 1 - \prod_{s' \in S_A(s,d)} (1 - dist(s, s')^{-2}\ B_t(s')) & \text{if } F_t(s) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

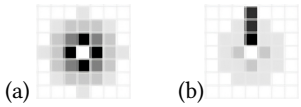
In Expression 2, $p_{t+\mho}(s)$ depends on the state of cells adjacent to s, designated by the set $S_A(s, d) = \{s' \in S \mid dist(s, s') \leq d\}$, where $dist : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is a distance function, and $d \in \mathbb{R}^+$ is a distance threshold. The term $dist(s, s')^{-2}$ captures an inversely proportional relation, which allows to give more importance to the state of cells that are closer to s. Moreover, we define a fire spread rate parameter $\mho$ that controls the speed at which fire propagates from cell to cell (i.e., $\forall t_i,\ \text{with } t \leq t_i < t + \mho \bullet p_{t_i} = 0$).

2.0.2 Wind. It is modeled as a bias on cell igniting probabilities that has direction $w \in \{north, south, east, west\}$ and strength $\mu \in [0, 1]$:

$$p_{t+1}(s') = \begin{cases} p_t(s') + \mu(1 - p_t(s')) & \text{if } s' \in S_\mu(s, w, d) \\ p_t(s') - \mu p_t(s') & \text{otherwise} \end{cases} \quad (3)$$

In Expression 3, $S_\mu(s, w, d) \subseteq S_A(s, d)$ is the set of cells that are in direction w with respect to cell s. Figure 1) illustrates fire spread probabilities in a sample grid.

2.0.3 Smoke. Is characterized as a function mapping a cell to three time-dependent variables $\kappa : S \rightarrow \{0, 1\} \times \mathbb{N} \times \mathbb{N}$. Hence, for $s \in S$, we have a tuple $(\gamma, \gamma_\alpha, \gamma_\omega)$, where $\kappa_t(s).\gamma$ indicates whether the cell contains smoke, $\kappa_t(s).\gamma_\alpha$ is a counter that keeps track of the time between the instant in which a cell ignites and smoke appears, and $\kappa_t(s).\gamma_\omega$ is a counter that keeps track of the time elapsed between the appearance of the smoke in the cell, and its dissipation.



(a) (b)

Figure 1: Fire spread probability for a 7×7 grid: (a) no wind, (b) $w = north$ and $\mu = 0.75$. Darker means higher probability.

Table 1: Mission requirements for UAV wildfire tracking.

Id	Description
R1	**Effective wildfire monitoring.** UAV should maximize the informative value of the monitored area of active wildfire, which is designated by $M_{R1} = \sum_{t=0}^{T} \sum_{s \in \biguplus_{u \in U}(S^O_{\square_t}(u))} v_t(s)$.
R2	**Collision risk avoidance.** The system should minimize the number of events in which UAV are in close proximity (i.e., closer than safety distance threshold δ_s), which is designated by $M_{R2} = \sum_{t=0}^{T} \lvert e_t(U) \rvert / 2$, where $e_t(U) = \{(t, u, u') \in [0, T] \times U \times U \mid u \neq u' \wedge dist(u.s_t, u'.s_t) \leq \delta_s\}$.

2.0.4 Unmanned Aerial Vehicles. We model a UAV as a tuple $u = (s, A, \delta^O)$, where $s \in S$ is its position in the forest area grid, A is the set of possible actions, and $\delta^O \in \mathbb{N}$ is a distance threshold that determines the set of observable cells for u. We designate the set of UAV by U. The set of actions for U consists in moving to adjacent cells in four directions, i.e., $\forall u \in U \bullet u.A = \{north, south, east, west\}$. UAV are equipped with a down-facing camera able to monitor its *influence area*, i.e., the set of cells centered around its position $S^O_\square(u) = \{s' \in S \mid D_{Cheb}(u.s, s') \leq 2 \cdot \delta^O\}$, with D_{Cheb} being the Chebyshev distance function. We characterize burning cells in the influence area of a UAV as $S^O_{burn}(u) = \{s \in S^O_\square(u) \mid B(s) = 1\}$. There is no value in redundant information about the same area monitored by more than one UAV, so we characterize overlapping cells in $S^O_\square(u)$ that are also in the influence area of a different UAV u' as $S^O_{over}(u) = \{s \in S^O_\square(u) \mid \exists u' \in U \bullet u \neq u' \wedge s \in S^O_\square(u')\}$.

2.0.5 Mission Requirements. During operation, the team of UAV should satisfy the set of requirements captured in Table 1. Requirement R1 captures the effective tracking of the wildfire and consists in maximizing the informative value of the set of monitored cells during system operation (metric M_{R1}). In the expression, $v : S \rightarrow \mathbb{R}_{\geq 0}$ is a function that assigns a constant informative (i.e., utility) value $C \in \mathbb{R}^+$ to a cell s in the grid if it is burning and not covered by smoke (i.e., $v_t(s) = C$ if $B_t(s) = 1 \wedge \kappa_t(s).\gamma = 0$), and zero, otherwise. Note that the use of $\biguplus$ in the expression avoids accruing value for the same cell more than once when it is monitored by more than one UAV. Requirement R2 targets safety during operation by defining metric M_{R2}, which counts the number of instances in which two UAVs are closer than a safety distance threshold δ_s.

3 APPROACH

Our decision-making approach is designed to work at run-time in the planning stage of the adaptation loop of MAPE-K [14] systems. Figure 2 shows the run-time operational context of our approach.

Each UAV is running an instance of the MAPE-K adaptation loop illustrated in the figure. In line with other proactive adaptation proposals, our approach follows an adaptation paradigm inspired by the main ideas behind Model Predictive Control (MPC) [15],

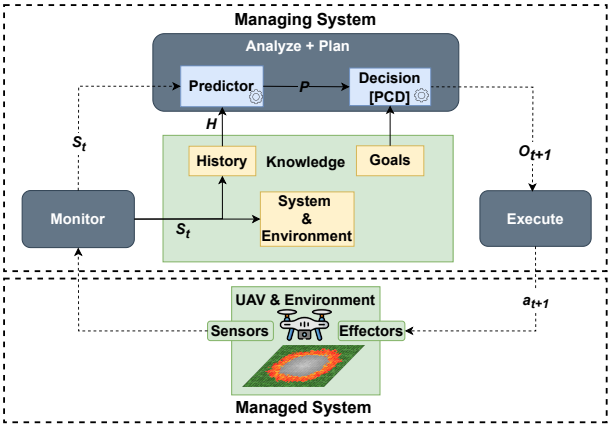


Figure 2: Run-time operational context of our approach.

which include: (i) using predictive models to forecast future system behaviour, (ii) the computation of a sequence of control (i.e., adaptation) actions, and (iii) using a *receding horizon*, i.e., repeating the selection of the sequence of control actions from the *current system state* after the execution of the first control action. Using a receding horizon allows mitigating the impact of potential disturbances (i.e., deviations from predictions in the actual behavior of the operational environment), resulting in more robust adaptation.

3.0.1 Monitor. During the monitoring, information from the sensors of the UAV at time instant t, including the observed area from the downwards-facing camera S_t^O, is aggregated (into the current state of the system and environment, designated by s_t) and incorporated into the System & Environment model, as well as into the History, which stores the historical sequence of observations of s_t.

3.0.2 Analyze and Plan. Analysis and planning are tightly integrated. **Analyze:** Analysis employs a Predictor component that receives information in every decision period from the system and environment model (s_t), and predicts the next elements of a sequence of observations of the system (i.e., from a sequence historical observations $H = \langle s_{t-k}, \ldots, s_{t-1}, s_t \rangle$ obtained from the History model, it predicts the next h time steps $P_t = \langle s_{t+1}, \ldots, s_{t+h} \rangle$, where h is the length of the receding horizon). Our approach is agnostic to the Predictor used, which can be implemented using multiple technologies, some of which have been tried for proactive adaptation in centralized decision-making, e.g., Recurrent Neural Networks (RNN) [9], Long Short-Term Memory (LSTM) networks [5], autoregressive (AR) time series predictors [20]. **Plan:** With as input P_t, planning uses a Decision component that runs our PCD algorithm (cf. Section 3.1) to produce a plan for the UAV O_{t+1}, guided by a Goals model that contains a goal specification based on the reward function described in Expression 4, and the requirements in Table 1.

3.0.3 Execution. The output of the planner O_{t+1} is then mapped as a set of actions a_{t+1} to effectors in the UAV.

3.1 Predictive Coordinate Descent

Our algorithm is based on Predictive Coordinate Descent (PCD) [22], which is used primarily for large-scale optimization in machine learning. In its basic form, coordinate descent successively minimizes along coordinate directions to find the minimum of a function. In each step, the algorithm optimizes the objective function with respect to a single coordinate or a subset of coordinates while keeping others fixed. This process is repeated, cycling through all coordinates or a subset of them until a convergence criterion is met. PCD improves over the traditional coordinate descent method by incorporating predictive modeling. In our context, this involves exploiting a prediction of the evolution of the environment to guide the direction of optimization. By predicting which coordinates are most likely to lead to a rapid decrease in the objective function, the algorithm can converge faster than standard coordinate descent.

Algorithm 1: Predictive Coordinate Descent (PCD)

Inputs: Set of UAVs U; Prediction P and horizon length h; Maximum number of iterations z_{max}; **Outputs:** Set of actions selected for UAVs O.
PCD(U, P, h, z_{max}) :
  $\sigma^\star \leftarrow \emptyset$
  $\mu_\sigma^\star \leftarrow -\infty$
  for $z = 0$ **to** z_{max} **do**
    $\sigma \leftarrow \{(u, \langle\rangle) \mid u \in U\}$
    for $step = 0$ **to** h **do**
      foreach $u \in U$ **do**
        if $step = 0$ **then**
          $\sigma(u) \leftarrow \sigma(u) \frown \langle random(u.A) \rangle$
        else
          $\sigma(u) \leftarrow \sigma(u) \frown \langle \underset{a \in u.A}{argmax}\ P(\sigma(u) \frown \langle a \rangle) \rangle$
        end
      end
    end
    $\mu_\sigma \leftarrow \frac{\sum_{u \in U} P(\sigma(u))}{|U|}$
    if $\mu_\sigma > \mu_\sigma^\star$ **then**
      $\sigma^\star \leftarrow \sigma$
      $\mu_\sigma^\star \leftarrow \mu_\sigma$
    end
  end
  $O = \{(u, a) \mid u \in U \wedge a = head(\sigma^\star(u))\}$
  return O

3.1.1 Reward. Based on the characterization of burning and overlapping cells in the influence area of a UAV (cf. Section 2), we define a reward function $r : U \rightarrow [-1, 1]$, which is calculated as:

$$r(u) = \frac{|S^O_{burn}(u)|}{|S^O_\square(u)|} - \frac{|S^O_{over}(u)|}{|S^O_\square(u)|} \tag{4}$$

The first term of Expression 4 adds a positive reward in the range $[0, 1]$ that gets closer to one as the number of observed burning cells is maximized. The second term penalizes the reward obtained by subtracting a value in the range $[0, 1]$ that is directly proportional to the number of cells observed by the UAV that are also in the area of influence of other UAVs. This reward function is crafted to balance requirements R1 and R2 (cf. Table 1): while the first term contributes to the maximization of M_{R1}, the second term contributes both to avoiding the maximization of M_{R1} due to the redundant observation of burning cells, as well as to the minimization of M_{R2}, because the minimization of overlapping areas also penalizes actions that make a UAV invade another UAV's influence area.

3.1.2 Algorithm. Algorithm 1 computes the best set of actions (O) for a set of UAVs (U), given a prediction of the system's behavior $P : A^* \rightarrow \mathbb{R}$ with a prediction horizon of length h. P receives as input sequences of actions of the form $\langle a_1, a2, \ldots, a_k \rangle$ with $k \leq h$, and provides as output a the accrued reward measure predicted over the sequence of states traversed as the actions in the sequence get executed. The algorithm is designed to run in the decision loop of UAV continually, where P is updated after each decision period, following the sliding window paradigm of MPC.

The algorithm starts by initializing $\sigma^\star$, which is the best set of UAV action sequences (i.e., those that maximize P over the horizon) for the set of UAVs. Then, for a maximum number of iterations z_{max}, it builds a set of paths for the UAVs (σ), where each element maps a UAV u with an action sequence (initially empty, cf. line 6). If at the end of the construction of σ, its mean reward across all UAVs (μ_σ, line 16) is better than the mean reward for the best set of paths $\sigma^\star$ found so far ($\mu^\star_\sigma$), we update the best set of paths and its average accrued reward with the new, better values (lines 17-20).

As for the construction of the different σ during each of the z iterations (lines 6-15), the process starts by selecting the first action of each UAV randomly (line 10). This random selection is intended to guarantee the diversity in the exploration of different combinations of action sequences.[1] Once that first action is selected, the process continues by extending the existing prefix of the action sequence with the action that maximizes the predicted accrued reward (line 12). The algorithm ends by returning the initial actions for each of the best action sequences identified in $\sigma^\star$ (line 22). Function $head : A* \rightarrow A$ returns the first action in an action sequence.

4 EVALUATION

We evaluate our approach with the objective of assessing its *effectiveness* (RQ1) and *efficiency* (RQ2) by performing experiments on Wildfire-UAVSim, a simulator of our own development[2].

Experimental Setup. All experiments considered scenarios of increasing team sizes (1-3 UAV), navigating a grid of 60×60 cells, with vegetation density of 1.0 (meaning that all cells in the grid have vegetation), and the amount of fuel in cells is set to $\forall s \in S, 11 \leq F_0(s) \leq 14$. Fire is seeded at the center of the grid. Burning rate was set to $\beta = 1$, and the fire spread rate to $\mho = 2$.

The variants compared were: (PCD) an implementation of Algorithm 1 in a MPC controller, with a horizon length $h = 3$, and maximum number of iterations $z_{max} = 15$; and (DQN) a reactive controller based on a Deep-Q-Network trained on data obtained from 30 simulations of the scenario described above. Both PCD and DQN use the reward function in Expression 4 for decision making. Experiments were performed on an Intel Core i9 CPU with 32GB of RAM. Our software setup included a Python v3.10 run-time running on Ubuntu 22.04.2 LTS 64 bits. Major dependencies include the Mesa v1.2.1 framework [13] for the implementation of multi-agent systems, and Pytorch v2.0.0 [24], used by the DQN baseline.

4.1 Results

(RQ1) ***Effectiveness.*** We performed experiments comparing the effectivess of DQN vs. PCD with an increasing number of UAV.

(RQ1.1) How effective is PCD compared to DQN under optimal predictive conditions? We compared PCD and DQN under four scenarios: (i) normal operating conditions (no wind, no smoke), (ii) harsh operating conditions (wind, no smoke), (iii) partial observability conditions (smoke, no wind), and (iv) harsh and partial observability conditions (wind and smoke). In all scenarios, PCD had a perfect prediction of the evolution of the scenario until the planning horizon (ground truth obtained by looking ahead in the simulation).

Figure 3 shows the efficiency of DQN and PCD on M_{R1} (effective monitoring), and M_{R2} (collision avoidance). Figure 3a shows

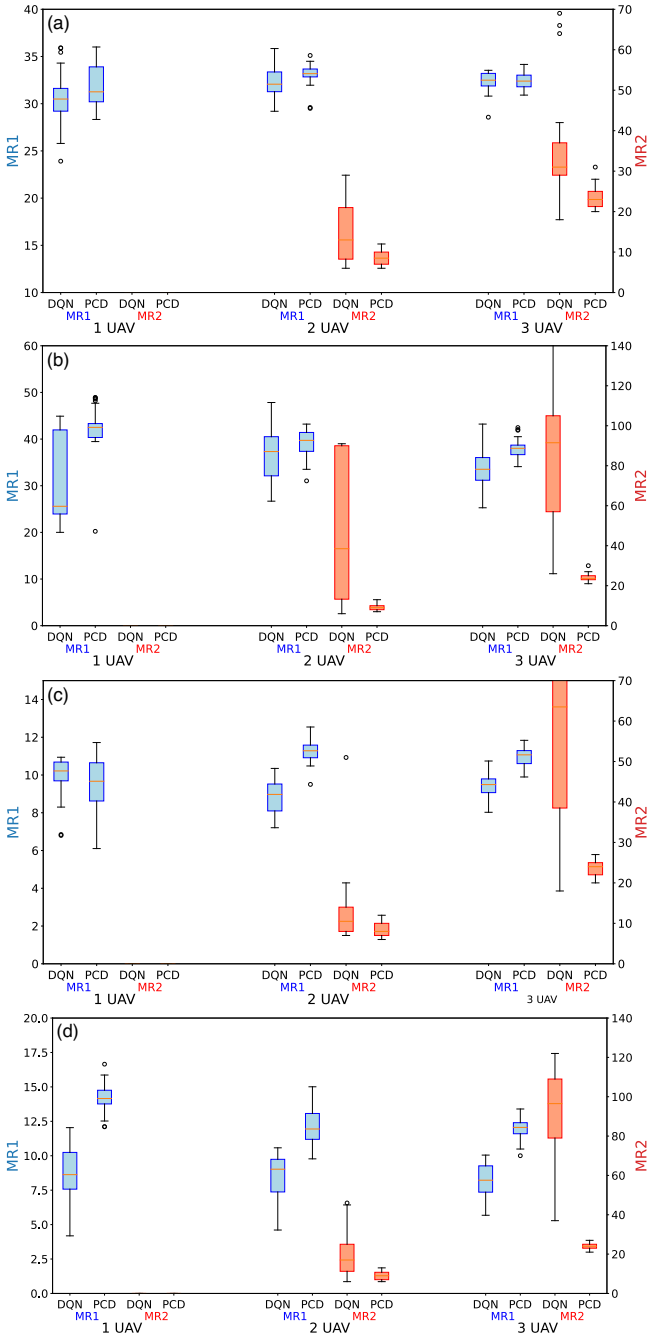


Figure 3: Effectiveness with ground truth predictions: (a) normal conditions, (b) wind, (c) smoke, and (d) smoke and wind.

that, for standard conditions (no wind, no smoke), PCD performs generally better than DQN both along M_{R1} (higher is better) and M_{R2} (lower is better). While performance on M_{R1} is comparable across all UAV team sizes, for M_{R2} PCD performs better and is more consistent than DQN, which shows much more dispersion across executions. Figure 3b illustrates results under wind conditions, which show that the improvement of PCD over DQN along the two metrics is even more remarkable, especially in safety, with much more dispersion in M_{R2} for DQN. With smoke only (Figure 3c)

[1] In fact, this selection is pseudo-random and requires that all UAVs are synchronized on the seed for the generated pseudo-random sequence to guarantee that they all obtain the same estimation of rewards.

[2] A replication package with all the code and data required to run our experiments can be downloaded from https://github.com/atenearesearchgroup/Wildfire-Simulator-DQN-PCD.

we find the same trend, with a slight improvement of PCD over DQN on M_{R1} (saving for the 1-UAV case, which shows a slight degradation). For smoke and wind (Figure 3d) we see again better results for PCD across the board, with even further improvement on M_{R1}, compared to previous cases.

> **Summary:** PCD outperforms DQN in terms of effective monitoring and collision risk avoidance with ground truth predictions. The difference is more remarkable for collision risk avoidance under wind and/or smoke conditions.

(RQ1.2) How effective is PCD compared to DQN under increasingly high levels of degradation in predictions? We evaluated PCD under sub-optimal predictive conditions by furnishing it with different versions of the predictions that were degraded with respect to the ground truth to emulate predictor components of different accuracies. Let $P^{\star} : A^{*} \rightarrow \mathbb{R}$ be a perfect prediction of the environment. We define $P' : A^{*} \rightarrow \mathbb{R}$ as a function that returns a biased value of the estimated accrued reward for an action sequence σ from the current time instant until the horizon. The biased result returned by $P'(\sigma)$ is defined by a Gaussian distribution with mean $P^{\star}(\sigma)$, and a controllable variance parameter (with values in $\{0.1, 0.5, 1\}$).

Figure 4 shows the effectiveness of DQN and PCD with predictions that have different levels of degradation. Figure 4a (standard operational conditions) shows that DQN always performs better than PCD on M_{R1}, with degradations in PCD performance that are, as expected, positively correlated with the variance of the biased prediction. However, for M_{R2} we can see that for the 2-UAV case, PCD performs better than DQN even in the high variance case, and for the 3-UAV case PCD still performs better than DQN with low variance, although with medium and high variance its performance degrades. Figure 4b (wind) show how PCD and DQN's effectiveness on M_{R1} are on par with low variance, although DQN always performs better with medium-high variance. Results for M_{R2} show a similar trend, compared to the results under standard operating conditions, although the improvement of PCD with respect to DQN is further exacerbated in this case, showing much more robust results for PCD with 2 UAV, even under high variance. For the 3-UAV case, we still observe better results of PCD with low-medium variance, and comparable results (slightly degraded) with respect to DQN for high variance. Figure 4c (smoke) shows a general trend similar to the one under wind conditions, with the notable exception of the 2-UAV case, in which DQN yields comparable results to PCD for M_{R2} with medium-high variance (PCD still performs better with low variance). Figure 4d (smoke and wind) shows how the trend is similar to those observed in figures 4b and 4c, but again with worse results for DQN in the 2-UAV case on M_{R2}.

> **Summary:** With moderately degraded predictions, PCD outperforms DQN in monitoring, but performs worse than DQN for medium-degradation. However, in collision risk avoidance PCD performs generally better than DQN and provides comparable results even with high degradation.

(RQ2) ***Efficiency****. How do the computational costs of PCD vs DQN compare?* We measured decision times for both alternatives (Table 2), which show how DQN is much more efficient than PCD. The order of magnitude of DQN decision-making is in the range 0.3-0.4 ms. This efficiency was expected because the controller is based on a pre-trained network that, at run time, only has to look up which action has to be performed for the current state. In contrast, PCD exhibits much higher decision-making times (≈ 2000-4200 ms.). This is explained by the time it takes to explore multiple potential paths until the planning horizon (cf. Algorithm 1). Despite the high decision times, compared to DQN, we have to put that in the context of the application domain. In wildfire tracking, decision-making times of a few seconds are acceptable and the tradeoff with aspects such as safety (cf. results for RQ1) is worthwhile, whereas admittedly, in the context of other real-time applications, these decision-making times might be insufficient to satisfy system requirements. In any case, the decision time for the PCD algorithm shows linear growth with the number of UAV, indicating of a reasonable degree of scalability for the class of application described in this paper.

Table 2: Decision times for PCD and DQN.

#UAV	DQN Time (ms)	PCD Time (ms)
1	0.320	2250.226
2	0.383	3104.799
3	0.324	4152.055

> **Summary:** DQN is much more efficient than PCD. However, decision-making times for PCD are acceptable in the context of the application domain.

4.2 Threats to Validity

Construct validity: Our study explores how PCD performs in comparison with a reactive approach in experiments where the prediction of the environment used by PCD is based on ground truth obtained from a simulation of the adaptation scenario, instead of a specific predictor (e.g., LSTM, RNN, AR). This threat is mitigated by artificially introducing noise into the predictions provided to PCD in our experiments. This allows us fine-grained control over the quality of predictions, enabling us to study how the performance of the adaptation strategy degrades under progressively poorer predictions. **Conclusion validity:** The transitory load of the machine where the performance experiments were executed can influence the performance results. To mitigate this threat, we run all experiments 30 times and averaged results across all runs. **Internal validity:** We compared PCD with a DQN baseline. We are aware of other alternatives, such as Multi-Objective Genetic Algorithms like NSGA-II [27], or probabilistic and statistical model checking [16, 17] used in (centralized) proactive adaptation approaches such as PLA [20] or TVA-e [9]. While these can be eventually adapted to a decentralized setting, this adaptation is not trivial and the direct, state-of-the-art competitor to PCD in wildfire tracking with UAVs is the DQN-based approach [12] that we have adapted to our adaptation scenario (cf. Section 5). We plan to explore the alternatives mentioned in future work. **External validity:** Our experiments are in wildfire tracking scenarios and we need to explore other types of CPS to generalize our results, although this threat is mitigated by shared commonalities between other sCPS (e.g., [1, 8]), which also include distributed components (e.g., robots [1], UUV [8, 25]) that have to operate subject to analogous conditions and uncertainties (e.g., limited sensing, infrequent opportunities for communication).

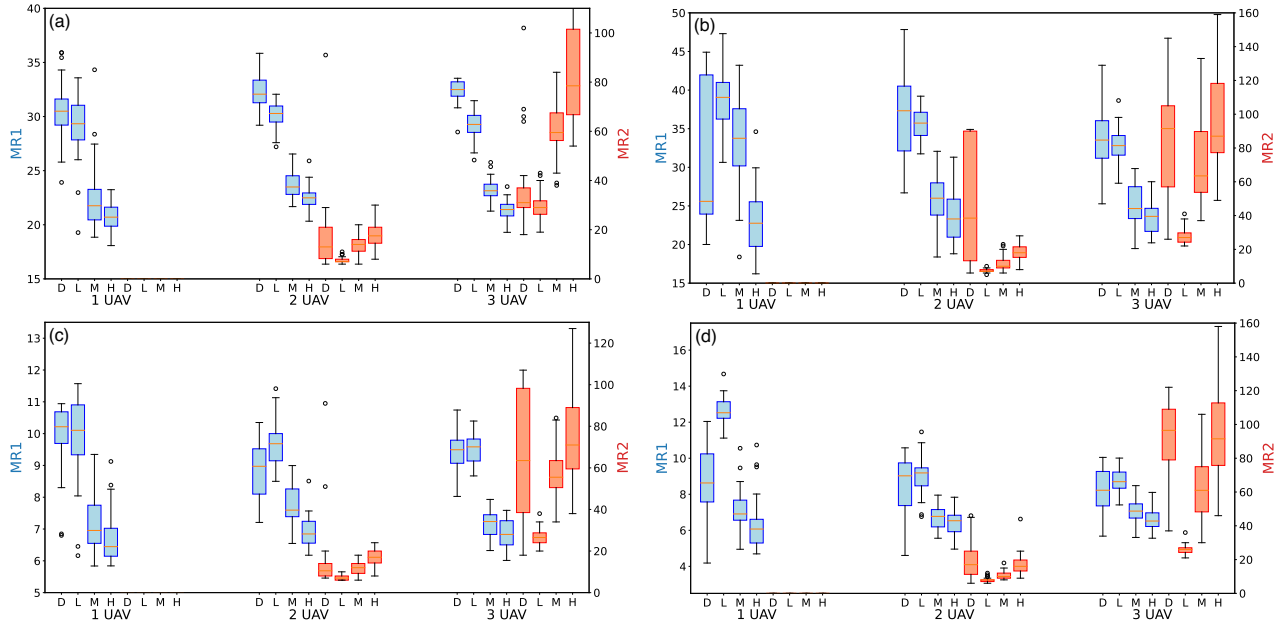


Figure 4: Effectiveness with degraded predictions: (a) normal conditions, (b) wind, (c) smoke, and (d) smoke and wind. D=DQN, L=PCD LOW VARIANCE (0.1), M=PCD MEDIUM VARIANCE (0.5), H=PCD HIGH VARIANCE (1).

5 RELATED WORK

We cannot give an exhaustive account of related approaches, but include a subset in this section strongly connected to our proposal.

AI/ML-based Wildfire Surveillance: Among AI/ML approaches in the intersection of navigation in UAV [3] and their applications to wildfire science [11], we highlight Julian et al.'s [12], which employs *Deep Reinforcement Learning (DRL)* to control a team of UAV that collaborate to maximize the monitored fire area autonomously. Each UAV follows *Dubins' kinematic model*, taking into account environmental factors like angle, speed, and observation capacity. Results show UAV ability to accurately monitor wildfire progression, although the model does not consider smoke. Our reactive DQN evaluation baseline is an extension of this approach adapted to work with quadcopter-type UAV (instead of winged UAV), as well as to consider smoke. PCD is also used as baseline for the evaluation of [12] in a MPC controller that produces decisions based on a fixed belief fire map of the environment (combinations of possible UAV actions over the planning horizon are considered without prediction of the evolution of the environment). Although not directly applied to wildfire surveillance, Baldazo et al. [2] build on the work described by Julian et al. [12] to contribute a DRL algorithm for UAVs to monitor flood areas, also using reactive adaptation.

Self-adaptive systems: Proactive adaptation approaches (e.g., [5, 7, 20]) have explored issues such as latency of adaptation actions (with [9] and without [20] uncertainty in latency) and sCPS reconfiguration [5], always from a centralized perspective. DECIDE [4] is an approach to build distributed self-adaptive software used in mission-critical applications that, despite being reactive, is able to provide robust quality levels in the presence of failures, mitigating to some extent some of the same issues addressed by proactive adaptation (e.g., increased resilience with moderate disruptions). Hnetynka et al. [10] introduce a method for modeling collaboration between autonomous components in smart farming, using dynamic ensembles. Omoniwa et al. [23] propose communication-enabled multi-agent decentralised Q-Network (DACEMAD-DDQN) for optimizing energy efficiency in UAV-assisted networks, enabling UAVs to collaborate and adapt to user density variations.

Multi-agent/robot systems: GMP [19] casts missions for heterogeneous multi-agent systems as an extension of the traveling salesperson problem and solve it using genetic algorithms. KANOA [26] combines constraint solving with quantitative verification for multi-robot task scheduling that is able to deal with task precedence constraints and multiple optimization objectives. MAPmAKER [18] 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. [6] tackle the spatial task allocation problem in warehouse commissioning of robots teams. These approaches are capable of decentralized decision-making under different forms of uncertainty and taking into consideration constraints (e.g., in resources, task precedence).

Despite their sophistication and to the best of our knowledge, existing approaches are either effective at performing proactive adaptation in a centralized fashion, or are capable of decentralized decision making, but reactive in nature and not equipped to perform anticipatory adaptation. Our study is a first step towards exploring this largely unaddressed, yet important gap.

6 CONCLUSIONS AND FUTURE WORK

We explored the potential of proactive decentralized decision-making in sCPS. However, results show that its effectiveness is highly dependent on the quality of predictions, and that it can actually be worse than reactive adaptation when predictions are highly biased.

In future work, we will explore techniques to complement predictions with confidence levels, so that the system can determine when to switch between proactive and reactive adaptation strategies. We also plan to build predictive components using RNN, AR, and LSTM networks to evaluate the effectiveness of PCD in their context, and apply our approach to other sCPS.

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] 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.

[2] David Baldazo, Juan Parras, and Santiago Zazo. 2019. Decentralized Multi-Agent Deep Reinforcement Learning in Swarms of Drones for Flood Monitoring. (2019).

[3] Suraj Bijjahalli, Roberto Sabatini, and Alessandro Gardi. 2020. Advances in intelligent and autonomous navigation systems for small UAS. *Progress in Aerospace Sciences* 115 (5 2020), 100617.

[4] 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.

[5] Javier Cámara, Henry Muccini, and Karthik Vaidhyanathan. 2020. Quantitative Verification-Aided Machine Learning: A Tandem Approach for Architecting Self-Adaptive IoT Systems. In *2020 IEEE International Conference on Software Architecture, ICSA 2020, Salvador, Brazil, March 16-20, 2020*. IEEE, 11–22.

[6] 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.

[7] 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.

[8] Simos Gerasimou, Radu Calinescu, Stepan Shevtsov, and Danny Weyns. 2017. UNDERSEA: An Exemplar for Engineering Self-Adaptive Unmanned Underwater Vehicles (Artifact). *Dagstuhl Artifacts Ser.* 3, 1 (2017), 03:1–03:2. https://doi.org/10.4230/DARTS.3.1.3

[9] Aizaz Ul Haq, Niranjana Deshpande, AbdElRahman A. ElSaid, Travis Desell, and Daniel E. Krutz. 2022. Addressing tactic volatility in self-adaptive systems using evolved recurrent neural networks and uncertainty reduction tactics. In *GECCO '22: Genetic and Evolutionary Computation Conference, Boston, Massachusetts, USA, July 9 - 13, 2022*, Jonathan E. Fieldsend and Markus Wagner (Eds.). ACM, 1299–1307.

[10] Petr Hnetynka, Tomas Bures, Ilias Gerostathopoulos, and Jan Pacovsky. 2020. Using Component Ensembles for Modeling Autonomic Component Collaboration in Smart Farming. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (Seoul, Republic of Korea) *(SEAMS '20)*. Association for Computing Machinery, New York, NY, USA, 156–162.

[11] Piyush Jain, Sean C.P. Coogan, Sriram Ganapathi Subramanian, Mark Crowley, Steve Taylor, and Mike D. Flannigan. 2020. A review of machine learning applications in wildfire science and management. *Environmental Reviews* 28 (2020), 478–505. Issue 4.

[12] Kyle D. Julian and Mykel J. Kochenderfer. 2018. Distributed Wildfire Surveillance with Autonomous Aircraft using Deep Reinforcement Learning. *Journal of Guidance, Control, and Dynamics* 42 (10 2018), 1768–1778. Issue 8. https://arxiv.org/abs/1810.04244v1

[13] Jackie Kazil, David Masad, and Andrew Crooks. 2020. Utilizing Python for Agent-Based Modeling: The Mesa Framework. In *Social, Cultural, and Behavioral Modeling*, Robert Thomson, Halil Bisgin, Christopher Dancy, Ayaz Hyder, and Muhammad Hussain (Eds.). Springer International Publishing, Cham, 308–317.

[14] Jeffrey O. Kephart and David M. Chess. 2003. The Vision of Autonomic Computing. *Computer* 36, 1 (2003), 41–50.

[15] Basil Kouvaritakis and Mark Cannon. 2016. Model predictive control. *Switzerland: Springer International Publishing* 38 (2016).

[16] 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.

[17] 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.

[18] 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. https://doi.org/10.1007/978-3-319-95582-7_24

[19] 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. https://doi.org/10.1109/TCYB.2021.3070913

[20] 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. https://doi.org/10.1145/2786805.2786853

[21] Gabriel A. Moreno, Alessandro Vittorio Papadopoulos, Konstantinos Angelopoulos, Javier Cámara, and Bradley R. Schmerl. 2017. Comparing Model-Based Predictive Approaches to Self-Adaptation: CobRA and PLA. In *12th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2017, Buenos Aires, Argentina, May 22-23, 2017*. IEEE Computer Society, 42–53. https://doi.org/10.1109/SEAMS.2017.2

[22] Yu Nesterov. 2012. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization* 22, 2 (2012), 341–362.

[23] Babatunji Omoniwa, Boris Galkin, and Ivana Dusparic. 2023. Density-Aware Reinforcement Learning to Optimise Energy Efficiency in UAV-Assisted Networks. *arXiv preprint arXiv:2306.08785* (2023).

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[25] Gustavo Rezende Silva, Juliane Päßler, Jeroen Zwanepol, Elvin Alberts, S Lizeth Tapia Tarifa, Ilias Gerostathopoulos, Einar Broch Johnsen, and Carlos Hernández Corbato. 2023. SUAVE: An Exemplar for Self-Adaptive Underwater Vehicles. In *2023 IEEE/ACM 18th Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 181–187.

[26] 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. https://doi.org/10.4204/EPTCS.371.11

[27] 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.