Metamorphic Testing of an Autonomous Delivery Robots Scheduler

Thomas Laurent
National Institute of Informatics
JSPS International Research Fellow
Tokyo, Japan
thomas-laurent@nii.ac.jp

Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
arcaini@nii.ac.jp

Xiao-Yi Zhang
University of Science and Technology Beijing
Beijing, China
xiaoyi@ustb.edu.cn

Fuyuki Ishikawa
National Institute of Informatics
Tokyo, Japan
f-ishikawa@nii.ac.jp

***Abstract*—Delivery systems operated by autonomous robots use schedulers to allocate robots to the different orders. Such schedulers are often optimisation-based algorithms that aim to maximise the number of delivered goods. The oracle problem affects the testing of these schedulers, as it is not always possible to assess whether the schedule produced for a given scenario is the optimal one. In this work, we propose a framework, based on a novel use of metamorphic testing, to assess the optimality of the scheduling algorithm developed by Panasonic for the management of a fleet of autonomous delivery robots in the Fujisawa Sustainable Smart Town, Japan. In the framework, a *metamorphic relation* (MR) transforms a *source test case* in a *followup test case* in a predefined way, and compares the results of the execution of the two tests in a simulated environment: if the comparison violates the expected relation, we can claim that one of the two schedules produced by the scheduler is suboptimal. We propose 19 MRs that target different aspects of the delivery system. Experiments over more than 900,000 test cases show that the different MRs have different abilities in exposing suboptimal behaviour and that most of the MRs do not subsume each other. Moreover, they also show that MR violations can provide useful insights into the scheduler's behaviour to Panasonic's engineers.**

***Index Terms*—autonomous robots, goods delivery, metamorphic testing**

I. Introduction

Services for goods delivery have become very popular in recent years, with a big surge during the COVID pandemic [1]. The critical phase of these services is the *last-mile delivery* in which the goods are carried from the store (e.g., a supermarket) to the final customer; such delivery is affected by different issues like traffic congestion [2], pollution [3], and high operational costs. Furthermore, in some countries like Japan, the system is difficult to sustain due to the lack of workers [4].

Autonomous delivery robots [5] have been proposed as a solution to the above-mentioned issues, as they guarantee a less costly service, with low impact on the traffic and environment. The Ministry of Economy, Trade and Industry (METI) of Japan is supporting joint initiatives between the public and private sectors [6], with the goal of assessing the impact of this technology.

P. Arcaini and F. Ishikawa are supported by Engineerable AI Techniques for Practical Applications of High-Quality Machine Learning-based Systems Project (Grant Number JPMJMI20B8), JST-Mirai. X. Zhang is supported by Youth Fund of the National Natural Science Foundation of China (No. 62302035). We thank our industry partner Panasonic for providing the software and the simulator used in this work, and for the discussing the principles of testing complex real-world autonomous delivery systems.

In this context, our industrial partner Panasonic has implemented an *autonomous delivery system* [7] performed by autonomous robots. The service is under evaluation in the Fujisawa Sustainable Smart Town [8] in Japan. The system allows town customers to order goods from a local store using a phone app. An operation centre collects all the orders in advance, and then during the day allocates the different autonomous robots to deliver the ordered goods. This allocation is done by an optimisation-based scheduler algorithm that tries to maximise the amount of delivered goods. When a robot is assigned to an order, it collects the requested good from the store, and delivers it to the customer. During the robot's journey, different unexpected events can happen, such as a pedestrian suddenly appearing on the robot's path, or other obstacles blocking the route. In this case, the robot stops and a human operator remotely checks the situation and, if needed, operates the robot to solve the issue; once solved, the robot continues operating autonomously.

One of the critical elements of the service is the *scheduler* that, given a set of orders, allocates the robots to deliver the ordered goods. The scheduler solves an optimisation problem whose goal is to maximise the number of deliveries. As most optimisation approaches, the scheduling algorithm developed by Panasonic does not guarantee to always find the optimal solution. Therefore, Panasonic is interested in assessing the *optimality* of the scheduler, i.e., to understand to what extent the schedules produced by the scheduler are optimal. To this end, a simulator is available that allows to simulate the system in different scenarios; a *scenario* defines the allocation of the robots (i.e., when each of them is available), the number of available human operators, and the set of customer's orders to deliver. The issue in testing this type of systems is the *oracle problem* [9], i.e., given the simulation results of a test, it is not always clear whether the obtained solution should be considered optimal (i.e., the test has *passed*) or suboptimal (i.e., the test has *failed*). Indeed, for an optimisation problem, the best solution is usually not known.

Proposed approach. Intuitively, if for a given test input (i.e., a scenario), we knew that a schedule better than the one found by the scheduler existed, we could assess that the test has failed. In this paper, we propose a testing approach for the scheduler, based on the concept of *metamorphic testing* [10], that, in some cases, is able to demonstrate that the solution found by the scheduler is suboptimal. The approach relies on the definition of *metamorphic relations*. A metamorphic relation (MR) works as follows: starting from a *source test case* t_{src}, it produces a *follow up test case* t_{fol} using a *transformation* that modifies t_{src} in a predefined way. For example, it can add some orders to the orders set of t_{src}. Then, both tests t_{src} and t_{fol} are run, and their test results compared. Depending on the applied MR, we expect that an optimal scheduler guarantees some relation between the results of the two tests. For our previous example, we expect that the number of delivered orders in t_{fol} is higher, or at least the same, than in t_{src}. If this is not the case (i.e., a lower number of orders is delivered in t_{fol}), we can claim that the scheduler found a suboptimal schedule in t_{fol}: indeed, a better schedule would have been the one found for t_{src}.

We propose 19 MRs, that modify different elements of the source test case: the set of customer orders (adding, removing, or modifying them), the availability time of the robots, and the total duration of the delivery service. Experiments show that different MRs have different abilities in exposing suboptimal behaviour of the scheduler and that most of the MRs do not subsume each other.

Paper structure. Sect. II introduces the necessary background about Panasonic's autonomous delivery system. Then, Sect. III introduces the proposed metamorphic testing approach, and Sect. IV the MRs we designed. Sect. V describes the experiments we conducted to assess the approach, and Sect. VI presents the experimental results. Finally, Sect. VII discusses some threats that may affect the validity of this work and steps taken to mitigate them, Sect. VIII reviews related work, and Sect. IX concludes the paper.

II. Preliminaries

In this work, we consider an *autonomous delivery system* developed by our industry partner Panasonic, in which goods are delivered by autonomous robots (as the one shown in Fig. 1). The overall system operates as follows:

- customers submit *orders* through an application, i.e., they ask for a particular good to be delivered at their home. All orders for a particular day must be submitted at least the day before;
- the orders are collected by an operation center that uses a *scheduling algorithm* (*scheduler* in the following) to serve the orders with the autonomous robots;
- specifically, during the day, the scheduler allocates the different robots to the different orders, with the goal of maximising the number of delivered orders, but still guaranteeing that all the orders are considered. Due to IP protection, we cannot disclose the implementation, nor the details of the scheduler. For the sake of understanding, it is sufficient to know that the scheduler solves an *optimisation problem* by employing some heuristics that consider:
  - the distance of the orders from the store;
  - the relative position among orders; indeed, robots can serve multiple orders, and orders that are close to each other are candidates to be served by the same robot in the same trip;
  - when the robots are available.

Fig. 1: Goods delivery with autonomous robots (from [7])

Panasonic developed a simulator that allows to experiment with different scenarios. Specifically, the simulator takes in input a *configuration* $conf$ that describes the way the system is deployed, and a set of orders $Ords$ that need to be delivered by the autonomous robots. The configuration $conf$ consists of the following elements:

- $SI = [si_s, si_e]$: the *service interval* during which the system is operational; for example, the system could operate only in the morning or the full day (from morning to evening);
- $\#robs$: the *number of robots* available for the delivery service;
- $AvailTimes$: a set of *available time* intervals $AT_i = [at_s^i, at_e^i]$ (with $|AvailTimes| = \#robs$), describing the time interval when each robot rob_i is deployed; indeed, Panasonic is considering not to deploy all the robots for the whole service interval SI, to avoid congestion and mitigate the costs; moreover, fewer robots operating in the city at the same time could favour the social acceptance of the service;
- $seed_{ords}$: a seed that characterises a given distribution of orders on the map;
- $\#humOps$: the number of human operators monitoring the system; having more operators allows for solving the situations in which the robots are stuck faster, but increases the costs.

$Ords$ is a set of $\#Ords$ orders $\{ord_0, \ldots, ord_{\#Ords-1}\}$, where each of them describes the destination loc of a given delivery. We identify with $Locs$ the set of possible delivery locations. All deliveries are made from the same store.

The output of the simulation consists in several metrics and pieces of data, like the number of delivered orders, the utilisation of the robots, the number of human interventions, etc. In this work, we are interested in the quality of service, and so we will consider the number of delivered orders $\#DelOrds$.

III. Proposed metamorphic testing approach

As explained in Sect. II, the whole system uses a scheduler to allocate robots to the delivery of the different orders. The scheduler implements an optimisation algorithm to try to maximise the number of orders that are actually delivered.[1]

As any optimisation algorithm, the scheduler does not guarantee to generate the optimal solution. Panasonic is currently in the process of assessing the quality of the scheduler, to understand to what extent the solutions computed by the scheduler are optimal. Such assessment is very important for the company, as a better scheduler allows to provide a better service for the customers.

To make this assessment, Panasonic uses the simulator introduced in Sect. II that allows to test the scheduler under different scenarios (*tests* in the following), in terms of system configurations and sets of orders. The issue of this assessment is that there is no oracle available that can tell whether a computed schedule is optimal; therefore, Panasonic engineers currently need to manually inspect simulation results and try to spot possible issues.

In this paper, we propose a metamorphic testing approach that, to some extent, is able to assess the optimality of the scheduler. First, Sect. III-A introduces the approach, and then Sect. IV describes the MRs we propose.

A. Approach overview

As explained in Sect. I, testing the scheduler is affected by the oracle problem [9], as, for a given computed schedule, it is not clear whether the schedule is optimal. Metamorphic testing [10] has been usually applied for systems affected by the oracle problem. In this work, we propose an application of metamorphic testing to assess the optimality of an optimisation algorithm.

Metamorphic testing relies on a set of *metamorphic relations* (MRs). An MR $\mathcal{M}$ is defined by two components:

- a *transformation* $\mathcal{M}.trans(t_{src})$ that, given an original test case t_{src}, produces a set of *followup test cases* $\{t_{fol}^1, \ldots, t_{fol}^k\}$. Sometimes, the MR produces only one followup test case. Note that, in some cases, the MR is not applicable; in this cases, it returns $\emptyset$.
- a *relation* $\mathcal{M}.rel(t_{src}, t_{fol})$ that defines an expected property between the results $Res(t_{src})$ of the source test case t_{src} and the results $Res(t_{fol})$ of a followup test case t_{fol}. If the property is satisfied, it means that the system works as expected, otherwise, a *violation* is found. For the general case of a set of followup tests $\{t_{fol}^1, \ldots, t_{fol}^k\}$, the expected property is satisfied if the comparison is satisfied for all the tests, i.e.,

$$\mathcal{M}.rel(t_{src}, \{t_{fol}^1, \ldots, t_{fol}^k\}) = \bigwedge_{i=1}^{k} \mathcal{M}.rel(t_{src}, t_{fol}^i) \quad (1)$$

Note that the property is satisfied if $\mathcal{M}.trans(t_{src}) = \emptyset$, i.e., if the relation's transformation does not produce any followup test from t_{src}.

[1]Note that Panasonic is experimenting with different system configurations of the robots (in terms of number of robots and their operating times), that provide different trade-offs between cost and customer service. Therefore, for the sets of orders considered in this paper, it is expected that some orders will not be delivered.

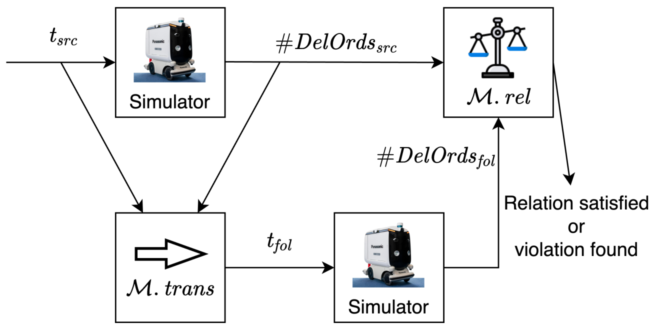


Fig. 2: Proposed metamorphic testing approach – Generation of a followup test case and MR assessment

Fig. 2 illustrates the process of generation and assessment of tests in our proposed metamorphic testing framework. In our system, a test t is given by the pair $\langle conf, Ords\rangle$, where $conf$ is the configuration of the robots and $Ords$ is a set of orders. Executing a test t means simulating t with the simulator. Among the different outputs returned by the simulator, in this work we are interested in the *number of delivered orders* $\#DelOrds$, as this allows to assess the performance of the scheduler; therefore, in the following, we assume that $Res(t) = \#DelOrds$.

So, given a source test case t_{src} and a followup test case t_{fol}, we collect the outputs:

$$(\#DelOrds_{src}, \#DelOrds_{fol})$$

where $Res(t_{src}) = \#DelOrds_{src}$ and $Res(t_{fol}) = \#DelOrds_{fol}$. The pair of outputs is compared using the predicate $\mathcal{M}.rel$ that defines the expected property. The MRs defined in this work (see Sect. IV) have three types of expected properties:

- $\#DelOrds_{src} \leq \#DelOrds_{fol}$: if the property is violated (i.e., $\#DelOrds_{fol}$ is less than $\#DelOrds_{src}$), it means that the scheduler produced a suboptimal schedule for t_{fol}. This is used when the scheduler could have used the schedule produced for t_{src} to achieve at least the same results (i.e., the same number of deliveries) on t_{fol};
- $\#DelOrds_{src} \geq \#DelOrds_{fol}$: if the property is violated (i.e., $\#DelOrds_{fol}$ is greater than $\#DelOrds_{src}$), it means that the scheduler produced a suboptimal schedule for t_{src}; indeed, the scheduler could have generated in t_{src} the exact same schedule produced for t_{fol}, guaranteeing to obtain the same number of deliveries;

- $\#DelOrds_{src} = \#DelOrds_{fol}$: if the property is violated, it means that the scheduler produced a suboptimal schedule for the test having the lower number of deliveries.

For brevity, in the following, we will identify the three properties as $\texttt{expProp}_{\leq}$, $\texttt{expProp}_{\geq}$, and $\texttt{expProp}_{=}$.

Remark 1. Note that, in our case, a violation of an MR does not indicate a functional failure (i.e., there is no functional bug in the implementation), but a suboptimality of the scheduler. Of course, some level of suboptimality is inevitable in optimisation algorithms that rely on heuristics. Our approach is an application of metamorphic testing that allows to better "understand" the behaviour of the system under test; this type of metamorphic testing has been recently advocated by Zhou et al. [11]. The violations found by our approach are useful for Panasonic's engineers to better "understand" the scheduler and to what extent it is optimal. Starting from the obtained results, they can consider whether further improvement is needed, or decide that the shown suboptimal behaviours are acceptable.

IV. Proposed Metamorphic Relations

This section describes the metamorphic relations (MRs) we designed to assess the optimality of the scheduler. We identified two main categories of MRs, that transform different elements of the source test case: those affecting the set of orders (as described in Sect. IV-A) and those affecting the configuration of the system (as described in Sect. IV-B).

A. MRs affecting the set of orders

The MRs of this category modify the set of orders of the source test case. Specifically, given an MR $\mathcal{M}$ and a source test case $t_{src} = \langle conf_{src}, Ords_{src}\rangle$, the followup test case t_{fol} is obtained as follows:

$$\mathcal{M}.trans(t_{src}) = \langle conf_{src}, Ords_{fol}\rangle$$

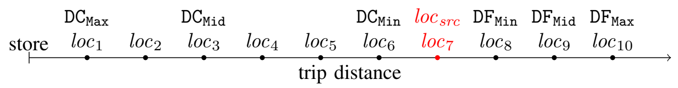
i.e., $conf_{src}$ remains the same and only $Ords_{src}$ is changed.

We identify two general ways to modify orders, as explained in Sects. IV-A1 and IV-A2.

1) Adding/removing orders: The first way to create followup tests by modifying $Ords_{src}$ is to either add or remove orders, as done by the following relations.

Add Order Randomly (AOR): it produces a followup test by adding an order to $Ords_{src}$ with a random destination loc, where loc is sampled from the set $Locs$ of possible delivery locations; it produces k followup tests. The expected property is that the number of delivered orders in the followup tests increases or remains the same, i.e., $\texttt{expProp}_{\leq}$.

Add Order Systematically (AOS): it produces a followup test by adding an order to $Ords_{src}$ in a *systematic* way, by considering the distance to the store. Specifically, it produces k followup tests, where the i^{th} test adds an order to the $\left\lfloor \frac{i \times |Locs|}{k} \right\rfloor^{\text{th}}$ closest location from the store. As AOR, it checks that the number of delivered orders in the followup tests increases or remains the same, i.e., $\texttt{expProp}_{\leq}$.



$$Locs = \{loc_1, \ldots, loc_{10}\}$$

Fig. 3: Illustration of transformations of $\texttt{DC}_{\texttt{Min}}$, $\texttt{DC}_{\texttt{Mid}}$, $\texttt{DC}_{\texttt{Max}}$, $\texttt{DF}_{\texttt{Min}}$, $\texttt{DF}_{\texttt{Mid}}$, and $\texttt{DF}_{\texttt{Max}}$

Remove Order Randomly (ROR): it produces a followup test by randomly removing an order (either delivered or non-delivered) from $Ords_{src}$. It produces k followup tests. The expected property is that the number of deliveries in the followup tests decreases or remains the same, i.e., $\texttt{expProp}_{\geq}$.

Remove Delivered Order Systematically (RDOS): it removes a delivered order from $Ords_{src}$ in a systematic way, by considering the distance from the store. Specifically, it produces up to k followup tests, where the i^{th} test removes the $\lfloor \frac{i \times \#DelOrds_{src}}{k} \rfloor^{\text{th}}$ closest delivered order from the store. The expected property is that the number of delivered orders in the followup tests decreases or remains the same, i.e., $\texttt{expProp}_{\geq}$.

Remove Non-Delivered Order Systematically (RNDOS): The definition is exactly as RDOS, except for the fact that non-delivered orders are removed systematically, and that the expected property is $\texttt{expProp}_{=}$; indeed, if the order was not considered for delivery in t_{src}, removing it should not affect the overall schedule.

2) Modifying orders: The second way to create followup tests by modifying $Ords_{src}$ is to change the delivery location of the orders, as in the following relations.

Delivered Closer: This MR is defined in a parametric way as follows. Starting from a source test case t_{src}, it changes the location loc_{src} of a delivered order to a position p closer to the store, with p a parameter of the MR. The MR produces up to k followup tests, where the i^{th} followup test changes the $\lfloor \frac{i \times \#DelOrds_{src}}{k} \rfloor^{\text{th}}$ closest delivered order from the store. The expected property is that the number of delivered orders in the followup tests increases or remains the same, i.e., $\texttt{expProp}_{\leq}$. The parametric MR is instantiated in three concrete MRs, that differ in the position p where the order is moved, as visualised in Fig. 3 and described in the following:

- **Delivered Closer Max** ($\texttt{DC}_{\texttt{Max}}$): the order is moved to the closest location to the store (e.g., loc_1 in Fig. 3).
- **Delivered Closer Mid** ($\texttt{DC}_{\texttt{Mid}}$): the order is moved to the median location between the store and the original delivery location loc_{src} (e.g., loc_3 in Fig. 3).
- **Delivered Closer Min** ($\texttt{DC}_{\texttt{Min}}$): the order is moved to the next location closest to the store compared to the original delivery location loc_{src} (e.g., loc_6 in Fig. 3).

Delivered Further: The MR is defined in a parametric way as follows. Starting from a source test case t_{src}, it changes the location loc_{src} of a delivered order to a position p further from the store w.r.t. loc_{src}; p is a parameter of the MR. The MR produces up to k followup tests, where the i^{th} followup test changes the $\lfloor\frac{i\times\#DelOrds_{src}}{k}\rfloor^{\text{th}}$ closest delivered order from the store. The expected property is that the number of delivered orders in the followup tests decreases or remains the same, i.e., $\texttt{expProp}_{\geq}$. The parametric MR is instantiated in three concrete MRs, that differ in the position p where the order is moved, as visualised in Fig. 3 and described in the following:

- **Delivered Further Max** ($\texttt{DF}_{\texttt{Max}}$): the order is moved to the furthest location from the store (e.g., loc_{10} in Fig. 3).
- **Delivered Further Mid** ($\texttt{DF}_{\texttt{Mid}}$): the order is moved to the median location between the original delivery location loc_{src} and the furthest location from the store (e.g., loc_9 in Fig. 3).
- **Delivered Further Min** ($\texttt{DF}_{\texttt{Min}}$): the order is moved to the next furthest location to the store compared to the original delivery location loc_{src} (e.g., loc_8 in Fig. 3).

Non-Delivered Closer: This MR has a similar definition to "Delivered Closer", except that it moves orders non-delivered in the source test case t_{src}. Also in this case, the orders are moved to a position p closer to the store, being p a parameter of the MR. Three concrete MRs are instantiated by defining p in a different way:

- **Non-Delivered Closer Max** ($\texttt{NDC}_{\texttt{Max}}$): same as $\texttt{DC}_{\texttt{Max}}$, but for non-delivered orders.
- **Non-Delivered Closer Mid** ($\texttt{NDC}_{\texttt{Mid}}$): same as $\texttt{DC}_{\texttt{Mid}}$, but for non-delivered orders.
- **Non-Delivered Closer Min** ($\texttt{NDC}_{\texttt{Min}}$): same as $\texttt{DC}_{\texttt{Min}}$, but for non-delivered orders.

Non-Delivered Further: This MR has a similar definition to "Delivered Further", except that it moves orders non-delivered in the source test case t_{src}, and that its expected property is $\texttt{expProp}_{=}$; indeed, making orders that were not considered in the schedule of t_{src} harder to deliver should not affect the new schedule. Also in this case, the orders are moved to a position p further from the store w.r.t. loc_{src}, being p a parameter of the MR. Three concrete MRs are instantiated by defining p in a different way:

- **Non-Delivered Further Max** ($\texttt{NDF}_{\texttt{Max}}$): same as $\texttt{DF}_{\texttt{Max}}$, but for non-delivered orders.
- **Non-Delivered Further Mid** ($\texttt{NDF}_{\texttt{Mid}}$): same as $\texttt{DF}_{\texttt{Mid}}$, but for non-delivered orders.
- **Non-Delivered Further Min** ($\texttt{NDF}_{\texttt{Min}}$): same as $\texttt{DF}_{\texttt{Min}}$, but for non-delivered orders.

B. MRs affecting the configuration

The MRs of this category modify the configuration of the system. Specifically, given an MR $\mathcal{M}$ and a source test case $t_{src} = \langle conf_{src}, Ords_{src}\rangle$, the followup test case t_{fol} is obtained as follows:

$$\mathcal{M}.trans(t_{src}) = \langle conf_{fol}, Ords_{src}\rangle$$

i.e., $Ords_{src}$ remains the same and only $conf_{src}$ is changed.

The configuration can be modified by either modifying the system's service time, or the time of availability of the different robots.

Change Service Interval ($\texttt{CSI}_{\Delta}$): it produces a followup test by changing the service interval SI of the source test case by Δ minutes. The expected property depends on the value of Δ:

- If Δ is positive (i.e., the service interval is increased), we expect that the number of delivered orders increases or remains the same, i.e., $\texttt{expProp}_{\leq}$;
- If Δ is negative (i.e., the service interval is decreased), we expect that the number of delivered orders decreases or remains the same, i.e., $\texttt{expProp}_{\geq}$.

Change Available Time ($\texttt{CAT}_{\Delta}$): it produces a followup test by modifying the availability time AT of a robot; specifically, it changes it by Δ minutes, either at the beginning or the end of the period AT. The expected property depends on the value of Δ:

- If Δ is positive (i.e., the robot is available for longer time), we expect that the number of delivered orders increases or remains the same, i.e., $\texttt{expProp}_{\leq}$;
- If Δ is negative (i.e., the robot is available for a shorter period of time), we expect that the number of delivered orders decreases or remains the same, i.e., $\texttt{expProp}_{\geq}$.

V. Experiment design

This section describes the experiments we designed to assess the effectiveness of the approach. All experimental results and the code used to obtain them are available in the online repository [12]; we cannot share the simulator of system for IP protection.

A. Research questions

We will assess the proposed metamorphic testing framework using the following research questions (RQs):

- **RQ1**: What is the effectiveness of the proposed metamorphic testing approach?
  This RQ assesses how often the different MRs are violated, i.e., find suboptimal schedules that should be inspected by Panasonic's engineers.
- **RQ2** How large are the violations when the relations are not respected?
  This RQ assesses how large the violations uncovered by each of the MRs are. The larger the violation, the further away a schedule is from optimality.
- **RQ3** Are there subsumption relationships amongst the proposed MRs? If not, to what extent detecting one MR guarantees to detect another MR?
  This RQ investigates whether some of the proposed MRs subsume others, i.e., if they find the non-optimal cases that other relations find. If such subsumption relationships were present, the subsumed relations would be redundant. Even if an MR $\mathcal{M}_1$ does not fully subsume another MR $\mathcal{M}_2$, if detecting $\mathcal{M}_1$ guarantees most of the times to also detect $\mathcal{M}_2$, we could avoid checking $\mathcal{M}_2$, thus saving testing time.

In addition to answering the previous RQs, we will make a more *qualitative* analysis to check which type of failures are discovered by the approach and which insights they can provide to Panasonic's engineers.

B. Benchmarks

The optimality of the scheduler can only be assessed in cases where the resources (in terms of number of robots and their available times) are scarce, i.e., the system cannot deliver all the orders or it can barely deliver all of them. Indeed, if the system is over-equipped, delivering all the orders is not difficult, even if the scheduler is not optimal. Instead, in a case of scarce resources, the difference between an optimal and a suboptimal scheduler can be demonstrated.

Therefore, in order to assess our framework, we selected source test cases in which not all the orders can be delivered or they can be delivered by using all the robots all the time. Specifically, we selected a test suite TS_{src} of 8100 source test cases $t_{src} = \langle conf_{src}, Ords_{src} \rangle$ by changing the following elements (see Sect. II):

- $seed_{ords}$: we use 100 seeds that describe different distributions of orders in the town;
- $\#Ords$: set $Ords$ can contain 20, 25, and 30 orders;
- $\#humOps$: in the system, there can be 1, 2, or 3 human operators to overview the operations of the robots;
- SI: we consider one service interval [09:00-12:00];
- nine different settings of the robots (number and their available times) that Panasonic can evaluate to consider aspects like cost of the service, occupancy of the road, and customer satisfaction:
  - *full allocation settings*: four settings in which $\#robs \in \{2, 3, 4\}$. In these settings, the robots are available for the whole service interval, i.e., $at_s^i = si_s \wedge at_e^i = si_e$ for each robot i ($i \in \{1, \ldots, \#robs\}$);
  - *partial allocation settings*: six settings in which not all the robots are executed during the whole service interval:
    $AvailTimes_1$ = {[09:00-12:00], [10:00-11:00]}
    $AvailTimes_2$ = {[09:00-12:00], [09:00-10:30], [10:30-12:00]}
    $AvailTimes_3$ = {[09:00-10:30], [09:30-11:00], [10:00-11:30], [10:30-12:00]}
    $AvailTimes_4$ = {[9:00-10:30], [10:30-12:00]}
    $AvailTimes_5$ = {[9:00-10:30], [9:30-11:00], [10:30-12:00]}
    $AvailTimes_6$ = {[09:00-12:00], [09:00-10:30], [10:30-12:00], [10:00-11:00]}

Some MRs produce k tests, with k being a parameter of the MR. For the experiments, we set $k = 5$ for all these MRs.

Table I reports statistics related to the transformations applied by each MR $\mathcal{M}$. Specifically, *#$\mathcal{M}$ applications* reports how many times $\mathcal{M}$ has been applied, while *#followup tests* reports the total number of produced followup tests (considering that some MRs produce $k = 5$ followup tests for each application). The number of MR violations will be counted at the level of *# MR applications*; *#followup tests* is reported to assess the cost of applying the testing framework.

We notice that, for some MRs like `AOR`, *#$\mathcal{M}$ applications* is exactly 8100, as these MRs can be applied to each source test case. For other MRs like `CAT`$_{\text{-}60}$, instead, *#$\mathcal{M}$ applications* is lower than 8100, as these MRs are applicable only to some source test cases (e.g., `CAT`$_{\text{-}60}$ is applicable only in the *full allocation settings* when the robots are utilised for the whole service interval).

In terms of *#followup tests*, they are the same as *#$\mathcal{M}$ applications* for `CSI`$_{\triangle}$ that produces only one followup test case, and higher for MRs producing multiple followup test cases. For some MRs like `AOR`, exactly $k = 5$ followup test cases are produced for each source test case; however, for some MRs like `NDC`$_{\texttt{Max}}$, for some source test cases it is not possible to generate all 5 followup test cases (e.g., for `NDC`$_{\texttt{Max}}$, there are less than 5 non-delivered orders that can be moved).

C. Evaluation metrics

In order to answer RQ1, we will report, for each MR $\mathcal{M}$, the *violation rate* $VR_{\mathcal{M}}$, i.e., the percentage of times that $\mathcal{M}$ has been violated (as defined in Eq. 1) out of the total number of times that it is applicable (i.e., the transformation produces some followup test), formally:

$$VR_{\mathcal{M}} = \frac{|\{t_{src} \in TS_{src} \mid \neg\mathcal{M}.rel(t_{src}, \mathcal{M}.trans(t_{src}))\}|}{|\{t_{src} \in TS_{src} \mid \mathcal{M}.trans(t_{src}) \neq \emptyset\}|} \tag{2}$$

In order to answer RQ2, for each MR $\mathcal{M}$, we will compute the *delivery difference* $DD_{\mathcal{M}}$, i.e., the difference between the result $\#DelOrds_{fol}$ of a followup test case and the results of the source test case $\#DelOrds_{src}$ in case of violation:

$$DD_{\mathcal{M}} = \#DelOrds_{fol} - \#DelOrds_{src} \tag{3}$$

We will report the distribution of $DD_{\mathcal{M}}$ across all the source and followup test cases for which a violation has been found.

In order to answer RQ3, given two MRs $\mathcal{M}_1$ and $\mathcal{M}_2$, we will report the *subsumption relation rate* $SR_{\mathcal{M}_1,\mathcal{M}_2}$ that identifies the number of source test cases for which $\mathcal{M}_1$ is violated and also $\mathcal{M}_2$ is violated, i.e.,

$$SR_{\mathcal{M}_1,\mathcal{M}_2} = \frac{\left|\left\{ t_{src}\in TS_{src} \;\middle|\; \begin{array}{l} \neg\mathcal{M}_1.rel(t_{src}, \mathcal{M}_1.trans(t_{src}))\wedge \\ \neg\mathcal{M}_2.rel(t_{src}, \mathcal{M}_2.trans(t_{src})) \end{array} \right\}\right|}{\left|\left\{ t_{src}\in TS_{src} \;\middle|\; \begin{array}{l} \neg\mathcal{M}_1.rel(t_{src}, \mathcal{M}_1.trans(t_{src}))\wedge \\ \mathcal{M}_2.trans(t_{src}) \neq \emptyset \end{array} \right\}\right|} \tag{4}$$

Note that the denominator selects the source test cases for which $\mathcal{M}_1$ finds a violation (and therefore it is applicable) and $\mathcal{M}_2$ is applicable, i.e., it produces followup tests.

VI. Experiment results

This section discusses the results of the experiments described above, following the RQs defined in Sect. V-A.

A. Answer to RQ1

This RQ assesses whether the proposed MRs can expose suboptimal behaviour of the scheduler. Fig. 4 reports, for each MR $\mathcal{M}$, the violation rate $VR_{\mathcal{M}}$ as defined in Eq. 1.

First of all, we observe that the framework is able to find some violations, showing its ability to expose suboptimality. However, different MRs have different violation rates.

TABLE I: Benchmarks – Number of applications of each MR $\mathcal{M}$, and number of generated followup tests

	AOR	AOS	ROR	RDOS	RNDOS	$\mathtt{DC}_{\mathtt{Max}}$	$\mathtt{DC}_{\mathtt{Mid}}$	$\mathtt{DC}_{\mathtt{Min}}$	$\mathtt{DF}_{\mathtt{Max}}$	$\mathtt{DF}_{\mathtt{Mid}}$	$\mathtt{DF}_{\mathtt{Min}}$	$\mathtt{NDC}_{\mathtt{Max}}$	$\mathtt{NDC}_{\mathtt{Mid}}$	$\mathtt{NDC}_{\mathtt{Min}}$	$\mathtt{NDF}_{\mathtt{Max}}$	$\mathtt{NDF}_{\mathtt{Mid}}$	$\mathtt{NDF}_{\mathtt{Min}}$	$\mathtt{CSI}_{-60}$
#$\mathcal{M}$ applications	8100	8100	8100	8100	5238	8100	8100	8100	8100	8100	8100	5236	5236	5236	5232	5232	5232	2700
#followup tests	40500	40500	40500	40500	24193	38094	38094	38094	37950	37950	37950	23521	23521	23521	23449	23449	23449	2700

	$\mathtt{CSI}_{-55}$	$\mathtt{CSI}_{-50}$	$\mathtt{CSI}_{-45}$	$\mathtt{CSI}_{-40}$	$\mathtt{CSI}_{-35}$	$\mathtt{CSI}_{-30}$	$\mathtt{CSI}_{-25}$	$\mathtt{CSI}_{-20}$	$\mathtt{CSI}_{-15}$	$\mathtt{CSI}_{-10}$	$\mathtt{CSI}_{-5}$	$\mathtt{CSI}_{5}$	$\mathtt{CSI}_{10}$	$\mathtt{CSI}_{15}$	$\mathtt{CSI}_{20}$	$\mathtt{CSI}_{25}$	$\mathtt{CSI}_{30}$	$\mathtt{CSI}_{35}$
#$\mathcal{M}$ applications	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700
#followup tests	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700	2700

	$\mathtt{CSI}_{40}$	$\mathtt{CSI}_{45}$	$\mathtt{CSI}_{50}$	$\mathtt{CSI}_{55}$	$\mathtt{CSI}_{60}$	$\mathtt{CAT}_{-60}$	$\mathtt{CAT}_{-45}$	$\mathtt{CAT}_{-30}$	$\mathtt{CAT}_{-15}$	$\mathtt{CAT}_{-10}$	$\mathtt{CAT}_{-5}$	$\mathtt{CAT}_{5}$	$\mathtt{CAT}_{10}$	$\mathtt{CAT}_{15}$	$\mathtt{CAT}_{30}$	$\mathtt{CAT}_{45}$	$\mathtt{CAT}_{60}$
#$\mathcal{M}$ applications	2700	2700	2700	2700	2700	2700	5400	5400	5400	5400	5400	5400	5400	5400	5400	5400	5400
#followup tests	2700	2700	2700	2700	2700	16200	32400	32400	32400	32400	32400	18000	18000	18000	18000	15300	15300

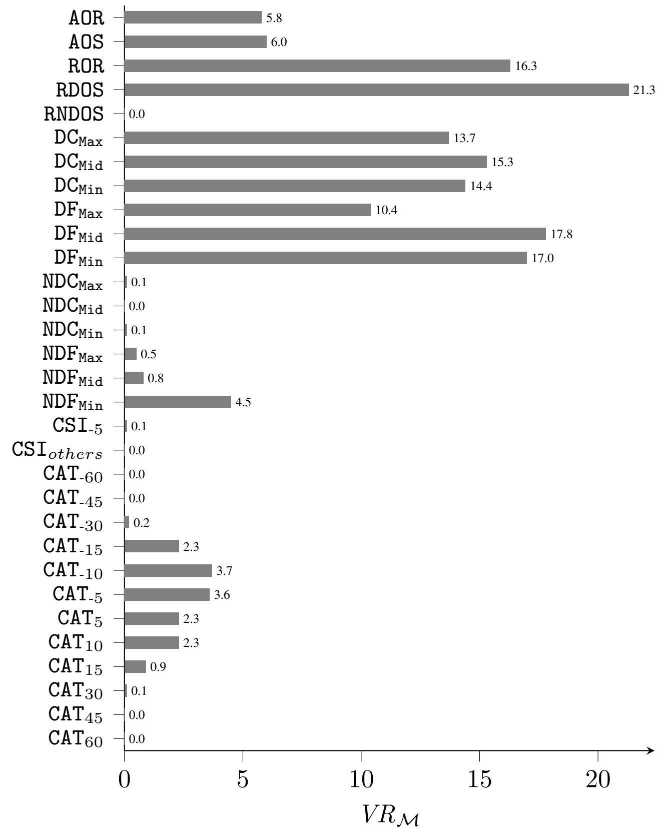


Fig. 4: RQ1 – *Violation rate* $VR_{\mathcal{M}}$

We notice that, overall, MRs that modify the set of orders (see Sect. IV-A) are violated more often than those that modify the configuration of the system (see Sect. IV-B). This is reasonable, as the performance of the scheduler is assessed in terms of number of delivered orders; therefore, modifying the orders set $Ords_{src}$ has a more direct impact on the task of the scheduler. Modifying the system configuration $conf_{src}$, instead, has a less direct impact on the task of the scheduler, although it can still influence it.

By looking at the specific MRs, we notice that RDOS that removes delivered orders in a systematic way has the highest violation rate. This can be explained by the fact that removing orders that were delivered leads to a different scheduling problem for the system; by removing an order that the scheduler considered in its schedule for t_{src}, the new problem can become simpler, leading the scheduler to find a more optimal schedule (i.e., leading to more delivered orders) that it could not find for t_{src}. In a similar way, ROR that removes random orders also has a high violation rate; however, it is smaller than RDOS as it also removes non-delivered orders, which does not affect the scheduling problem as much.

We also observe that all the MRs of type DC and DF –that modify the delivery location of orders that are delivered in the source test case– have high violation rates. Also in these cases, moving the location of orders leads the scheduler to plan different schedules that highlight suboptimal choices.

As expected, MRs of type NDC and NDF – that modify the delivery location of non-delivered orders – have low violation rates; indeed, if the scheduler did not consider them for delivery in the source test case t_{src}, it is unlikely that it will consider them when moving them in the followup test case t_{fol}. Still, $\mathtt{NDF}_{\mathtt{Min}}$ that moves a non-served order to the next furthest position, has a violation rate of 4.5%; for $\mathtt{NDF}_{\mathtt{Min}}$, we expect that nothing changes in terms of delivery (i.e., the expected property is $\mathtt{expProp}_{=}$), but we notice that, in the cases of violation, more orders are delivered. We inspected these cases and we observed that the moved orders are delivered together with other orders[2]; we will further analyse this case in Sect. VI-D.

Regarding the MRs related to the configuration, only $\mathtt{CAT}_{\Delta}$ that modifies the available time of a robot by Δ minutes exhibits non-negligible violation rates. In particular changing the available time between -15 and +10 minutes (i.e., $\mathtt{CAT}_{-15}$, $\mathtt{CAT}_{-10}$, $\mathtt{CAT}_{-5}$, $\mathtt{CAT}_{5}$, $\mathtt{CAT}_{10}$) can expose some violations; larger changes, instead, almost never lead to a violation. This is reasonable, as changing the available time of a robot of around 10 minutes corresponds to delivering or not an order: on these small variations, the scheduler is very sensitive and finding suboptimal behaviour is more likely.

[2]Recall that an order is moved furthest w.r.t. the store, but possibly in another part of the town.

TABLE II: RQ2 – *Delivery difference* $DD_{\mathcal{M}}$

	$DD_{\mathcal{M}}$					
	-3	-2	-1	1	2	3
AOR	0.0	1.35	98.65	-	-	-
AOS	0.0	1.49	98.51	-	-	-
ROR	-	-	-	95.03	4.66	0.31
RDOS	-	-	-	95.28	4.63	0.09
$\mathtt{DC_{Max}}$	0.0	1.94	98.06	-	-	-
$\mathtt{DC_{Mid}}$	0.0	2.98	97.02	-	-	-
$\mathtt{DC_{Min}}$	0.11	1.93	97.96	-	-	-
$\mathtt{DF_{Max}}$	-	-	-	97.3	2.7	0.0
$\mathtt{DF_{Mid}}$	-	-	-	96.22	3.78	0.0
$\mathtt{DF_{Min}}$	-	-	-	96.97	3.03	0.0
$\mathtt{NDC_{Max}}$	0.0	0.0	100.0	-	-	-
$\mathtt{NDC_{Min}}$	0.0	0.0	100.0	-	-	-
$\mathtt{NDF_{Max}}$	0.0	0.0	0.0	100.0	0.0	0.0
$\mathtt{NDF_{Mid}}$	0.0	0.0	0.0	100.0	0.0	0.0
$\mathtt{NDF_{Min}}$	0.0	0.0	0.0	99.22	0.78	0.0
$\mathtt{CSI}_{-5}$	-	-	-	100.0	0.0	0.0
$\mathtt{CAT}_{-30}$	-	-	-	100.0	0.0	0.0
$\mathtt{CAT}_{-15}$	-	-	-	100.0	0.0	0.0
$\mathtt{CAT}_{-10}$	-	-	-	98.67	1.33	0.0
$\mathtt{CAT}_{-5}$	-	-	-	100.0	0.0	0.0
$\mathtt{CAT}_{5}$	0.0	0.0	100.0	-	-	-
$\mathtt{CAT}_{10}$	0.0	0.0	100.0	-	-	-
$\mathtt{CAT}_{15}$	0.0	0.0	100.0	-	-	-
$\mathtt{CAT}_{30}$	0.0	0.0	100.0	-	-	-

TABLE III: RQ3 – *Subsumption relation rate* $SR_{\mathcal{M}_1,\mathcal{M}_2}$ ($\mathcal{M}_1$ is on the rows, $\mathcal{M}_2$ is on the columns)

	AOR	AOS	ROR	RDOS	$\mathtt{DC_{Max}}$	$\mathtt{DC_{Mid}}$	$\mathtt{DC_{Min}}$	$\mathtt{DF_{Max}}$	$\mathtt{DF_{Mid}}$	$\mathtt{DF_{Min}}$	$\mathtt{NDC_{Max}}$	$\mathtt{NDC_{Min}}$	$\mathtt{NDF_{Max}}$	$\mathtt{NDF_{Mid}}$	$\mathtt{NDF_{Min}}$	CSI	CAT
AOR	-	52.8	8.8	12.6	56.4	63.0	60.9	5.6	8.6	10.9	0.8	1.3	0.0	0.8	1.5	0.0	22.3
AOS	50.8	-	10.5	14.2	56.8	61.5	59.9	5.8	7.8	10.1	0.5	0.2	0.0	0.2	1.2	0.0	17.8
ROR	3.1	3.9	-	82.4	6.2	9.8	8.8	31.7	48.4	44.9	0.0	0.0	0.5	0.8	9.4	1.0	17.8
RDOS	3.4	4.0	63.2	-	6.8	9.3	8.9	31.6	50.3	47.2	0.0	0.0	0.3	0.6	8.2	0.8	15.9
$\mathtt{DC_{Max}}$	23.8	24.8	7.4	10.5	-	81.1	70.9	5.5	10.7	8.6	0.3	0.5	0.3	0.6	2.5	0.0	18.0
$\mathtt{DC_{Mid}}$	23.8	24.1	10.4	13.0	72.7	-	69.7	6.9	12.7	10.7	0.3	0.5	0.3	0.5	2.1	0.0	17.0
$\mathtt{DC_{Min}}$	24.4	24.9	9.9	13.1	67.5	74.0	-	6.1	8.7	8.7	0.3	0.5	0.3	0.6	2.2	0.0	16.1
$\mathtt{DF_{Max}}$	3.1	3.3	49.7	64.5	7.2	10.2	8.4	-	72.2	64.8	0.0	0.0	1.9	3.2	13.0	1.2	25.6
$\mathtt{DF_{Mid}}$	2.8	2.6	44.4	60.0	8.2	10.9	7.1	42.2	-	63.0	0.0	0.0	0.6	1.8	10.4	0.8	19.5
$\mathtt{DF_{Min}}$	3.7	3.5	43.1	59.0	6.9	9.6	7.4	39.6	65.9	-	0.0	0.0	1.2	2.5	12.6	0.9	21.7
$\mathtt{NDC_{Max}}$	100	66.7	0.0	0.0	100	100	100	0.0	0.0	0.0	-	0.0	0.0	0.0	0.0	NA	0.0
$\mathtt{NDC_{Min}}$	100	20.0	0.0	0.0	100	100	100	0.0	0.0	0.0	0.0	-	0.0	0.0	0.0	NA	40.0
$\mathtt{NDF_{Max}}$	0.0	0.0	25.0	25.0	12.5	12.5	12.5	66.7	37.5	66.7	0.0	0.0	-	66.7	58.3	0.0	5.0
$\mathtt{NDF_{Mid}}$	6.8	2.3	22.7	25.0	13.6	11.4	13.6	61.4	59.1	77.3	0.0	0.0	36.4	-	52.3	0.0	14.8
$\mathtt{NDF_{Min}}$	2.5	2.1	52.1	59.2	10.5	9.7	9.7	45.8	62.6	72.7	0.0	0.0	5.9	9.7	-	4.9	38.6
CSI	0.0	0.0	100	100	0.0	0.0	0.0	100	100	100	0.0	0.0	0.0	0.0	100	-	NA
CAT	16.2	12.7	39.2	45.8	33.5	34.5	30.2	33.7	45.6	48.9	0.0	0.4	0.2	0.8	15.1	NA	-

Answer to RQ1: Overall, the proposed MRs constitute an effective testing approach to assess the optimality of the scheduler. MRs affecting the set of orders have the higher violation rates, as they are more related to the scheduler's task. Still, MRs that apply small variations to the available time of a robot exhibit non-negligible violation rates.

B. Answer to RQ2

In this RQ, we want to assess "how much" the expected properties are violated. To do this, for each MR $\mathcal{M}$, we compute the *delivery difference* $DD_{\mathcal{M}}$ between each source test case and followup test case for which the expected property is violated (see Eq. 3). We observe that, at most, the difference in terms of delivered orders is ± 3. Table II reports, for each MR $\mathcal{M}$ that is sometimes violated (i.e., $VR_{\mathcal{M}} > 0$), the percentage of violations for which $DD_{\mathcal{M}} \in \{-3, -2, -1, 1, 2, 3\}$ (we report - if a value cannot be obtained in case of violation, e.g., a positive difference when the expected property is of type $\mathtt{expProp}_{\geq}$). We observe that most of the violations have a delivery difference of ± 1, demonstrating, on one hand, that the scheduler provided by Panasonic is quite good, as we cannot find too large violations; on the other hand, this also shows the power of our metamorphic framework that is able to unveil these subtle suboptimal behaviours.

Still, we see that for most MRs affecting the orders (AOR, AOS, ROR, RDOS, all DC, and all DF), the MRs violations have $DD_{\mathcal{M}} = 2$ in a non-negligible number of cases. As we have seen in RQ1, these MRs are the most effective ones, as they are those more related to the task of the scheduler. Note that improving the performance in these scenarios is very interesting for Panasonic, as being able to deliver two (even three in some cases) more orders is important in a setting in which the total number of orders to deliver is between 20 and 30 (see Sect. V-B).

Answer to RQ2: Overall, the Panasonic scheduler is quite robust and, therefore, the magnitude of the optimality violations are not too large. Still, the proposed metamorphic framework is able to reveal suboptimal behaviours that lead to deliver two or three less orders, which is non-negligible in the experimented settings.

C. Answer to RQ3

In this RQ, we want to discover whether there exists some subsumption relation between pairs of MRs ($\mathcal{M}_1$, $\mathcal{M}_2$), i.e., whether violating $\mathcal{M}_1$ implies violating $\mathcal{M}_2$ for the same source test case. Knowing this is very useful, as, in case of subsumption, one can save testing time by only assessing $\mathcal{M}_2$.

While assessing whether there is a *theoretical subsumption relation* is not possible, we can still check whether the relation holds experimentally. We do this by using the *subsumption relation rate* $SR_{\mathcal{M}_1,\mathcal{M}_2}$ as defined in Eq. 4. Table III reports the value for each pair of MRs (all the variants of CSI and CAT are reported in an aggregated way). An $SR_{\mathcal{M}_1,\mathcal{M}_2}$ of 100% (highlighted in yellow in the table) means that $\mathcal{M}_1$ is (experimentally) completely subsumed by $\mathcal{M}_2$. We observe

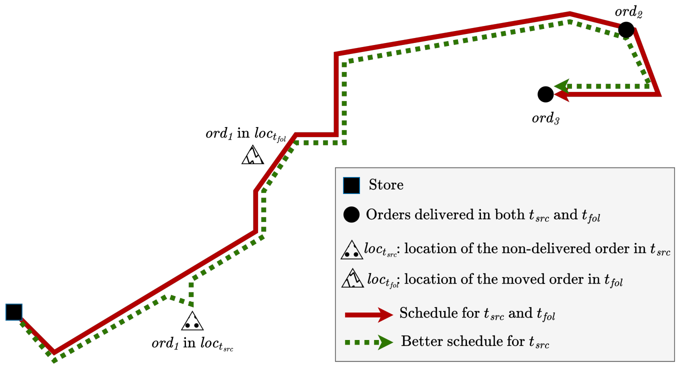

ord_2
ord_3
ord_1 in $loc_{t_{fol}}$
ord_1 in $loc_{t_{src}}$
■ Store
● Orders delivered in both t_{src} and t_{fol}
$loc_{t_{src}}$: location of the non-delivered order in t_{src}
$loc_{t_{fol}}$: location of the moved order in t_{fol}
Schedule for t_{src} and t_{fol}
Better schedule for t_{src}

Fig. 5: Qualitative evaluation – Example of suboptimal schedule in t_{src}

that this is the case for $\texttt{NDC}_{\texttt{Max}}$, $\texttt{NDC}_{\texttt{Min}}$, and $\texttt{CSI}$; assuming that the observed subsumption generalises to any test, we may think to avoid using these MRs in the future.

Other MRs like $\texttt{ROR}$, $\texttt{DC}_{\texttt{Max}}$, $\texttt{DC}_{\texttt{Mid}}$, $\texttt{DC}_{\texttt{Min}}$, $\texttt{DF}_{\texttt{Max}}$, $\texttt{NDF}_{\texttt{Mid}}$, and $\texttt{NDF}_{\texttt{Min}}$, although they are never fully subsumed, have high $SR_{\mathcal{M}_1,\mathcal{M}_2}$ (more than 70%, highlighted in gray in the table) with some other MR; in case of limited testing budget, one may consider to also avoid applying these MRs.

For the other MRs, instead, $SR_{\mathcal{M}_1,\mathcal{M}_2}$ is always less than 70% (often less than 60%), showing that they are able to expose different types of suboptimal behaviours and, therefore, should all be applied.

Answer to RQ3: MRs $\texttt{NDC}_{\texttt{Max}}$, $\texttt{NDC}_{\texttt{Min}}$, and $\texttt{CSI}$ are fully subsumed by other MRs, and so we can avoid applying them. MRs $\texttt{ROR}$, $\texttt{DC}_{\texttt{Max}}$, $\texttt{DC}_{\texttt{Mid}}$, $\texttt{DC}_{\texttt{Min}}$, $\texttt{DF}_{\texttt{Max}}$, $\texttt{NDF}_{\texttt{Mid}}$, and $\texttt{NDF}_{\texttt{Min}}$ are often subsumed by other MRs, and we could avoid applying them in case of limited resources. Other MRs, instead, overall expose different types of suboptimal behaviours and should all be applied.

D. *Qualitative evaluation*

This section analyses some violations discovered by the metamorphic framework to better understand the insights that can be gained by Panasonic's engineers through this technique. In particular, it highlights a case where a violated relation shows a suboptimal schedule in t_{src}, and one where it shows a suboptimal schedule in t_{fol}.

Example of suboptimal schedule in t_{src}: We consider the violation of $\texttt{NDF}_{\texttt{Min}}$ already discussed in Sect. VI-A. Fig. 5 visualises the violation. In the schedule computed for the source test case t_{src}, a robot is allocated to serve the orders ord_2 and ord_3; ord_1 (in position loc_{src}), instead, is not delivered during the whole service interval; the path followed by robot is the full red line. The application of $\texttt{NDF}_{\texttt{Min}}$ to t_{src} produces the followup test case t_{fol} by moving the non-delivered order ord_1 from location loc_{src} to location $loc_{t_{fol}}$ (along the path followed in the schedule for t_{src}). In the schedule for t_{fol}, ord_1 is delivered together with the other two orders ord_2 and ord_3. We inspected the execution of t_{src} and we noticed that the robot could have also delivered ord_1: indeed, the execution of t_{fol} shows that there is enough time for the collection of the ordered good by the user from the robot; moreover, we checked that the time required by the small detour is negligible and would have been allowed by the time constraints. So, we can claim that the path shown with the dashed green arrow would have been a better schedule. We can conclude that the current settings of the heuristics of the scheduler prefer smoother shorter delivery paths and that penalise too much slightly longer paths that can, however, provide a benefit in terms of number of deliveries.

Example of suboptimal schedule in t_{fol}: We consider a violation of MR $\texttt{AOS}$ that modifies a source test case t_{src} by adding an order ord_{new} to the order set $Ords$. We observe that, in the followup test t_{fol}, the system delivers the added order ord_{new}, but does not deliver anymore three orders that were delivered in t_{src}; so, in total, two less orders are delivered.[3] We inspected the schedule in t_{fol} and we noticed these differences w.r.t. the schedule in t_{src}:

- robot rob_1 delivers ord_{new}, but this leads to it not delivering order ord_1 that it delivered in t_{src}; moreover, rob_1 does not deliver order ord_a that ends up not being delivered by any robot;
- robot rob_2 now delivers ord_1, but does not deliver ord_2 anymore, that is now delivered by rob_0;
- because of these different allocations, rob_0 cannot deliver orders ord_b and ord_c anymore.

This result can be very interesting for Panasonic's engineers, as it shows how the delivery of one order can affect many others. This also shows that the heuristics applied by the scheduler are somehow locally greedy and cannot capture the whole picture. Although this is inevitable to some extent, it still provides some insight to improve the scheduling algorithm.

VII. Threats to Validity

The validity of the proposed metamorphic testing approach could be affected by different threats. By following classical categories [13], we discuss them in the following.

Construct validity: A construct validity threat could be that the metrics we use for assessing the metamorphic testing framework in the experiments are not suitable. The goal of the framework is to find suboptimal behaviours, so the *violation rate* $VR_{\mathcal{M}}$ (see Eq. 1) used in RQ1 allows to assess the effectiveness of each proposed MR. Moreover, our framework also wants to check how much suboptimal the scheduler is; to this aim, the *delivery difference* $DD_{\mathcal{M}}$ (see Eq. 3) used in RQ2 gives an indication how much suboptimal a schedule is. Finally, the proposed framework can be costly to execute in terms of time; therefore, checking the *subsumption relation rate* $SR_{\mathcal{M}_1,\mathcal{M}_2}$ (see Eq. 4) as done in RQ3 allows to understand which MRs are redundant and so possibly reduce the cost of the application of the approach.

[3]For $\texttt{AOS}$, the expected property is $\texttt{expProp}_{\leq}$, i.e., in t_{fol}, the system should deliver more or the same number of orders that is delivered in t_{src}.

Internal validity: An internal validity threat could be that the relationship between the applied approach and the obtained results is by chance. For example, the obtained satisfaction or violation of the MRs could be due to a faulty implementation. To mitigate this threat, we have carefully checked the implementation.

External validity: The metamorphic testing framework proposed in this work is specific to the autonomous delivery system of Panasonic. Therefore, it could be that it is not directly applicable to other delivery systems or, even more generally, to other systems that require a scheduling. We point out that we agree with Briand et al. [14] in observing that research conducted through an industry collaboration is "context-driven" by definition. Therefore, in this work, we report our contribution for testing the Panasonic system, and we leave as future work the extension to other systems.

VIII. Related work

In this section, we discuss works related to our approach. We first discuss metamorphic testing approaches in Sect. VIII-A, and then, in Sect. VIII-B, previous analysis approaches for the Panasonic autonomous delivery system.

A. Metamorphic testing

Metamorphic testing [10], [15] is a property-based testing approach that, in addition to testing [16], [17], [18], [19], has been applied to facilitate also other software engineering activities like debugging [20], [21], [22], and system explanation [11].

The main advantage of metamorphic testing is to alleviate the oracle problem. Various studies on metamorphic testing focus on the development of theories and techniques to identify high-quality MRs. For example, Chen et al. [23] and Sun et al. [24] proposed an approach based on category and choices to design MRs in an effective and efficient way. Qiu et al. [25] analysed the feasibility and effectiveness of merging multiple MRs, with the goal of reducing the number of followup tests to consider.

Metamorphic testing has been applied for the analysis of different domains, such as Web applications [26], computer vision [27], debuggers [28], deep learning compilers [29], chess engines [30], quantum computing [31], [32], elevators [33], [34], [35], Natural Language Processing (NLP) [36], Natural Language Inference (NLI) [37], textual content moderation software[38], and Conversational AI System [39].

Our framework targets the scheduler of autonomous robots. In the context of autonomous systems, metamorphic testing has been largely investigated [40], [41], [42]. For example, some works apply metamorphic testing to test the perception modules of autonomous driving systems (ADSs), in which the MRs produce transformations of the input images [43], [17], [44], [45]. Other works apply metamorphic testing for testing the ADS decision modules, such as the controllers and motion planners [46], [47].

We claim that our metamorphic testing approach can be used not only for testing, but also for understanding the system. Such use of metamorphic testing has been recently advocated by Zhou et al. [11]. They propose an approach in which metamorphic testing is used to facilitate final users to understand how a system works without knowing how it is implemented; this allows users to use the system better and, possibly, discover problems in the system. In order to derive MRs for this goal, the authors introduce the concepts of "symmetry" metamorphic relation pattern and a "change direction" metamorphic relation input pattern, which can be used as general guidance to derive multiple concrete metamorphic relations. In our context, the metamorphic testing framework is used by Panasonic's engineers and not the final users. Although the scheduler has been implemented by them, completely understanding its functioning is difficult; the metamorphic testing framework, in addition to assess optimality, also allows them to realise which are the consequences of some implemented heuristics in practice.

B. Analysis of the autonomous delivery system

The autonomous delivery system developed by Panasonic has been considered in other research works [48], [49], [50], [51], whose goal was to understand how to configure the robots (in terms of available time) to optimise stakeholder concerns like customer satisfaction, cost, and safety.

Arcaini et al. [48] proposed a multi-objective search approach that is able to show how different robots configurations lead to different trade-offs among the three stakeholder concerns; since each fitness evaluation requires different execution of the simulator, they also proposed two heuristics that allow to speed up the process, avoiding some simulations.

Byrd Victorica et al. [49] tackle the problem of explaining the Pareto fronts returned by the approach in [48]. Indeed, as these Pareto fronts could contain too many solutions, they may be difficult to understand by stakeholders. Therefore, they propose a clustering approach to group similar solutions of the Pareto front.

The approach we propose in this paper is complementary to those in [48], [49]. Given different system configurations found in those approaches, our current work wants to assess the optimality of the scheduler in them.

IX. Conclusion

This work presents a metamorphic testing framework to assess the optimality of Panasonic's autonomous delivery robot scheduling system. We proposed 19 metamorphic relations that target different aspects of the problem, such as the orders and the configuration of the robots.

Experiments over 8100 source test cases show that the approach is indeed able to expose suboptimal behaviour, although different MRs have different effectiveness. Moreover, experiments show that some MRs (experimentally) are subsumed by others and, therefore, we could avoid applying them.

As future work, we plan to extend the framework to other autonomous systems governed by a scheduler.

References

[1] X. C. Wang, W. Kim, J. Holguín-Veras, and J. Schmid, "Adoption of delivery services in light of the COVID pandemic: Who and how long?" *Transportation Research Part A: Policy and Practice*, vol. 154, pp. 270–286, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856421002676

[2] G. Mangano and G. Zenezini, "The value proposition of innovative last-mile delivery services from the perspective of local retailers," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 2590–2595, 2019, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319315848

[3] M. Viu-Roig and E. J. Alvarez-Palau, "The impact of e-commerce-related last-mile logistics on cities: A systematic literature review," *Sustainability*, vol. 12, no. 16, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/16/6492

[4] S. Reiko, "Japan's logistics crisis," https://www3.nhk.or.jp/nhkworld/en/news/backstories/1771/, 2021, last access: October 25, 2023.

[5] M. Figliozzi and D. Jennings, "Autonomous delivery robots and their potential impacts on urban freight energy consumption and emissions," *Transportation Research Procedia*, vol. 46, pp. 21–28, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146520303598

[6] METI, "Make delivery smart with automated delivery robots," https://www.meti.go.jp/english/mobile/2022/20220802001en.html, 2022, last access: October 25, 2023.

[7] Panasonic Holdings Corporation, "Panasonic to conduct field test of home delivery service by compact, low-speed robot in Fujisawa sustainable smart town," https://news.panasonic.com/global/press/en201214-1, December 2020, last access: October 25, 2023.

[8] M. Sakurai and J. Kokuryo, "Fujisawa sustainable smart town: Panasonic's challenge in building a sustainable society," *Communications of the Association for Information Systems*, vol. 42, no. 1, p. 19, 2018.

[9] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, May 2015.

[10] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–27, 2018.

[11] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic relations for enhancing system understanding and use," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1120–1154, 2020.

[12] T. Laurent, P. Arcaini, X. Zhang, and F. Ishikawa, "Supplementary material for the paper "Metamorphic Testing of an Autonomous Delivery Robots Scheduler"," https://github.com/ERATOMMSD/metamorphic_testing_delivery_robots, 2024.

[13] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[14] L. C. Briand, D. Bianculli, S. Nejati, F. Pastore, and M. Sabetzadeh, "The case for context-driven software engineering research: Generalizability is overrated," *IEEE Software*, vol. 34, no. 5, pp. 72–75, 2017.

[15] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Transactions on software engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[16] H. Liu, F.-C. Kuo, D. Towey, and T. Y. Chen, "How effectively does metamorphic testing alleviate the oracle problem?" *IEEE Transactions on Software Engineering*, vol. 40, no. 1, pp. 4–22, 2013.

[17] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 132–142. [Online]. Available: https://doi.org/10.1145/3238147.3238187

[18] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen, "Metamorphic testing: Testing the untestable," *IEEE Software*, vol. 37, no. 3, pp. 46–53, 2020.

[19] B. Zhang, H. Zhang, J. Chen, D. Hao, and P. Moscato, "Automatic discovery and cleansing of numerical metamorphic relations," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019, pp. 235–245.

[20] X. Xie, W. E. Wong, T. Y. Chen, and B. Xu, "Metamorphic slice: An application in spectrum-based fault localization," *Inf. Softw. Technol.*, vol. 55, no. 5, pp. 866–879, may 2013. [Online]. Available: https://doi.org/10.1016/j.infsof.2012.08.008

[21] M. Jiang, T. Y. Chen, Z. Q. Zhou, and Z. Ding, "Input test suites for program repair: A novel construction method based on metamorphic relations," *IEEE Transactions on Reliability*, vol. 70, no. 1, pp. 285–303, 2020.

[22] T. Y. Chen, T. Tse, and Z. Zhou, "Semi-proving: an integrated method based on global symbolic evaluation and metamorphic testing," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 4, pp. 191–195, 2002.

[23] T. Y. Chen, P.-L. Poon, and X. Xie, "METRIC: METamorphic Relation Identification based on the Category-choice framework," *Journal of Systems and Software*, vol. 116, pp. 177–190, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121215001624

[24] C.-A. Sun, A. Fu, P.-L. Poon, X. Xie, H. Liu, and T. Y. Chen, "METRIC+: a metamorphic relation identification technique based on input plus output domains," *IEEE Transactions on Software Engineering*, vol. 47, no. 9, pp. 1764–1785, 2021.

[25] K. Qiu, Z. Zheng, T. Y. Chen, and P.-L. Poon, "Theoretical and empirical analyses of the effectiveness of metamorphic relation composition," *IEEE Transactions on Software Engineering*, vol. 48, no. 3, pp. 1001–1017, 2022.

[26] J. Ahlgren, M. E. Berezin, K. Bojarczuk, E. Dulskyte, I. Dvortsova, J. George, N. Gucevska, M. Harman, M. Lomeli, E. Meijer, S. Sapora, and J. Spahr-Summers, "Testing web enabled simulation at scale using metamorphic testing," in *Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '21. IEEE Press, 2021, pp. 140–149. [Online]. Available: https://doi.org/10.1109/ICSE-SEIP52600.2021.00023

[27] Y. Yuan, S. Wang, M. Jiang, and T. Y. Chen, "Perception matters: Detecting perception failures of VQA models using metamorphic testing," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 903–16 912.

[28] S. Tolksdorf, D. Lehmann, and M. Pradel, "Interactive metamorphic testing of debuggers," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 273–283.

[29] D. Xiao, Z. LIU, Y. Yuan, Q. Pang, and S. Wang, "Metamorphic testing of deep learning compilers," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 1, feb 2022. [Online]. Available: https://doi.org/10.1145/3508035

[30] M. Méndez, M. Benito-Parejo, A. Ibias, and M. Núñez, "Metamorphic testing of chess engines," *Information and Software Technology*, vol. 162, p. 107263, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584923001179

[31] R. Abreu, J. a. P. Fernandes, L. Llana, and G. Tavares, "Metamorphic testing of oracle quantum programs," in *Proceedings of the 3rd International Workshop on Quantum Software Engineering*, ser. Q-SE '22. New York, NY, USA: Association for Computing Machinery, 2023, pp. 16–23. [Online]. Available: https://doi.org/10.1145/3528230.3529189

[32] M. Paltenghi and M. Pradel, "MorphQ: Metamorphic testing of the Qiskit quantum computing platform," in *Proceedings of the 45th International Conference on Software Engineering*, ser. ICSE '23. IEEE Press, 2023, pp. 2413–2424. [Online]. Available: https://doi.org/10.1109/ICSE48619.2023.00202

[33] J. Ayerdi, S. Segura, A. Arrieta, G. Sagardui, and M. Arratibel, "QoS-aware metamorphic testing: An elevation case study," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, 2020, pp. 104–114.

[34] J. Ayerdi, V. Terragni, A. Arrieta, P. Tonella, G. Sagardui, and M. Arratibel, "Generating metamorphic relations for cyber-physical systems with genetic programming: An industrial case study," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1264–1274. [Online]. Available: https://doi.org/10.1145/3468264.3473920

[35] J. Ayerdi, P. Valle, S. Segura, A. Arrieta, G. Sagardui, and M. Arratibel, "Performance-driven metamorphic testing of cyber-physical systems," *IEEE Transactions on Reliability*, vol. 72, no. 2, pp. 827–845, 2023.

[36] D. T. S. Lee, Z. Q. Zhou, and T. H. Tse, "Metamorphic robustness testing of Google translate," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 388–395. [Online]. Available: https://doi.org/10.1145/3387940.3391484

[37] M. Jiang, H. Bao, K. Tu, X. Zhang, and Z. Ding, "Evaluating natural language inference models: A metamorphic testing approach," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 2021, pp. 220–230.

[38] W. Wang, J.-t. Huang, W. Wu, J. Zhang, Y. Huang, S. Li, P. He, and M. R. Lyu, "MTTM: Metamorphic testing for textual content moderation software," in *Proceedings of the 45th International Conference on Software Engineering*, ser. ICSE '23. IEEE Press, 2023, pp. 2387–2399. [Online]. Available: https://doi.org/10.1109/ICSE48619.2023.00200

[39] Y. Wan, W. Wang, P. He, J. Gu, H. Bai, and M. Lyu, "BiasAsker: Measuring the bias in conversational ai system," *arXiv preprint arXiv:2305.12434*, 2023.

[40] C. B. Kuhn, M. Hofbauer, G. Petrovic, and E. Steinbach, "Introspective black box failure prediction for autonomous driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1907–1913.

[41] M. Iqbal, J. C. Han, Z. Q. Zhou, D. Towey, and T. Y. Chen, "Metamorphic testing of advanced driver-assistance system (ADAS) simulation platforms: Lane keeping assist system (LKAS) case studies," *Inf. Softw. Technol.*, vol. 155, no. C, mar 2023. [Online]. Available: https://doi.org/10.1016/j.infsof.2022.107104

[42] Y. Deng, X. Zheng, T. Zhang, H. Liu, G. Lou, M. Kim, and T. Y. Chen, "A declarative metamorphic testing framework for autonomous driving," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1964–1982, 2023.

[43] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 303–314. [Online]. Available: https://doi.org/10.1145/3180155.3180220

[44] S. Wang and Z. Su, "Metamorphic object insertion for testing object detection systems," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2020, pp. 1053–1065.

[45] P. Naidu, H. Gudaparthi, and N. Niu, "Metamorphic testing for convolutional neural networks: Relations over image classification," in *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE Press, 2021, pp. 99–106. [Online]. Available: https://doi.org/10.1109/IRI51335.2021.00020

[46] T. Y. Chen, F.-C. Kuo, W. K. Tam, and R. Merkel, "Testing a software-based PID controller using metamorphic testing," in *Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems - Volume 1: PECCS,*, INSTICC. Algarve, Portugal: SciTePress, 2011, pp. 387–396.

[47] L. Wu, Z. Xi, Z. Zheng, and X. Li, "Application of metamorphic testing on UAV path planning software," *Journal of Systems and Software*, vol. 204, p. 111769, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121223001644

[48] P. Arcaini, E. Castellano, F. Ishikawa, H. Kawamoto, K. Sawai, and E. Muramoto, "Incremental search-based allocation of autonomous robots for goods delivery," in *2023 IEEE Congress on Evolutionary Computation (CEC)*, 2023, pp. 1–10.

[49] M. Byrd Victorica, P. Arcaini, F. Ishikawa, H. Kawamoto, K. Sawai, and E. Muramoto, "Stability-aware exploration of design space of autonomous robots for goods delivery," in *2023 27th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2023, pp. 177–186.

[50] T. Laurent, P. Arcaini, F. Ishikawa, H. Kawamoto, K. Sawai, and E. Muramoto, "Investigating multi- and many-objective search for stability-aware configuration of an autonomous delivery system," in *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, Dec 2023.

[51] C. Sun, T. Laurent, P. Arcaini, and F. Ishikawa, "Alternating between surrogate model construction and search for configurations of an autonomous delivery system," in *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2024.