Long-Term Planning of Preventive Maintenance Using Constraint Programming: A Naval Case Study

Raphaël Boudreault[1][a] and Vanessa Simard[2][b]
[1]Thales, Québec, Canada
[2]Independent Researcher, Canada

Keywords: Preventive Maintenance, Planning, Constraint Programming, Optimization, Ship, Naval, In-Service Support.

Abstract: Maintenance planning is an essential element in the life-cycle management of an asset. Unplanned maintenance work can cause significant productivity and financial loss, while manually assessing compliance is complex and prone to errors. In the naval domain, ensuring mission readiness and operational availability is critical. Thus, periodic preventive maintenance tasks must be carefully allocated over a long-term horizon considering the ship availability, business rules, and workload limitations. This distribution over fixed short work periods can result in tasks being excessively advanced or deferred instead of executed when due. We propose a Constraint Programming approach to produce feasible tactical plans of preventive maintenance for ships minimizing advancements and deferrals of tasks. We validate our methodology on an industrial naval use case and demonstrate its relevance compared to a currently used planning method, greatly reducing over-maintenance with occurrences decreased by up to 25% and advancements by up to 93%. The method is integrated into Maintenance Optimizer™, an interactive planning solution that supports decision-making in this context.

1 INTRODUCTION

Maintenance planning is a critical component in ensuring the efficient life-cycle management of assets across various industries. Unplanned maintenance work can cause important disruptions in operational continuity, while incurring substantial productivity and financial losses. One common strategy to identify and schedule maintenance activities is to follow a *preventive* approach, where tasks are conducted to prevent system failures from happening. These activities may notably be planned according to fixed periodic recommendations prescribed by the Original Equipment Manufacturer (OEM). In the naval domain, this approach is widely used to ensure ship mission readiness and operational availability (Cullum et al., 2018). Given the highly constrained nature of ship availability, periodic maintenance tasks must be carefully allocated in short work periods over extended time frames. For maintenance program planners, this presents important challenges, as decisions made over fixed periods may result in excessive advancement or deferral of task executions, thereby introducing additional costs and risks. Moreover, the manual process of generating viable plans, which includes assessing maintenance compliance based on domain-specific business, certification, and workload limitation rules, is highly complex and susceptible to errors.

An important part of planning challenges can be lifted through the use of Computerized Maintenance Management Systems (CMMS), such as those provided by Oracle, SAP, and IBM Maximo. These tools enable real-time monitoring of enterprise assets and automate the generation of work orders based on pre-established maintenance frequency rules. However, their suitability for optimizing preventive maintenance planning in the naval domain, with its unique specificities, is known to be rather limited. Echoing the insights of (Van Horenbeek et al., 2010), who advocate for tailoring maintenance optimization models to specific applications, Thales Canada has set out to create Maintenance Optimizer™ as an integral component of its planning suite (Boudreault et al., 2022; Lafond et al., 2021). This interactive maintenance planning solution is expressly designed to enhance decision-making in this specialized context. The key motivation for this work comes from identified challenges and innovation opportunities within the con-

[a] https://orcid.org/0000-0002-5602-7515
[b] https://orcid.org/0000-0001-8861-8902

Boudreault, R. and Simard, V.
Long-Term Planning of Preventive Maintenance Using Constraint Programming: A Naval Case Study.
DOI: 10.5220/0013106400003893

In *Proceedings of the 14th International Conference on Operations Research and Enterprise Systems (ICORES 2025)*, pages 32-44
ISBN: 978-989-758-732-0; ISSN: 2184-4372

text of naval platforms in-service support programs. The solution is currently operational and deployed on a secure cloud platform, featuring various capabilities for comparing or modifying optimization decisions.

This paper presents a *Constraint Programming* (CP) approach, currently integrated into Maintenance Optimizer™, that optimizes the long-term planning of preventive maintenance for naval assets. CP is a powerful programming paradigm to solve combinatorial problems, notably to optimally solve many large-scale optimization problems under constraints (Rossi et al., 2006). The optimization problem under consideration in this work is described in Section 2, while Section 3 relates it within existing literature to similar problems and approaches. In Section 4, we introduce the CP model, along with user options and optional constraints available in the solution which may change its definition. Our approach is evaluated in Section 5 across multiple option configurations using a benchmark of five instances derived from a real maintenance program dataset. Finally, Section 6 deliberates on the applicability of the solution in our industrial setting, followed by a conclusion suggesting directions for future research.

2 PROBLEM DESCRIPTION

The problem addressed herein mirrors the challenges faced by ship maintenance program planners in implementing a preventive maintenance strategy within our specific use case. To minimize disruptive failures and costly corrective maintenance work, and thus ensure ship mission readiness and operational availability, preventive maintenance tasks are periodically planned based on recommendations from the equipment manufacturers. However, the windows for performing maintenance tasks are constrained by a predetermined schedule of ship availability, which is defined by the ship deployment periods. The planning horizon, which may span from one to five years, comprises non-overlapping work periods during which the ship is docked for maintenance purposes, typically occurring two to four times per year. The most common type of work period is relatively short, lasting between two to five weeks, although specific work periods with longer duration (e.g., dry-dock periods, where the ship is taken out of the water for extensive maintenance and repair), lasting up to four months, may also be scheduled. From a tactical planning perspective, each work period is limited in capacity by an estimated total available labor time, as well as a maximal task duration. Thus, from this perspective, labor time, duration, and costs are assumed to be approximate estimates of reality, yet are sufficient for long-term allocation of preventive work.

Typical ship maintenance programs contain around 700 distinct preventive maintenance tasks, each with estimated labor time and duration. A task consists of a set of sub-tasks designated for a specific system and location within the ship. The OEM-recommended *periodicity*, ranging from 1 to 180 months, describes how frequently each task should be performed. This periodicity, calculated from a baseline date known as *clock date*, establishes the *due date*, i.e. the date at which the maintenance is considered to be due. For instance, a 12-monthly task would usually be due around the same date each year. However, the ship's schedule may not always align precisely with these due dates. Hence, a *flexibility* rule is applied for most tasks, allowing for a safety range around the due date. This range usually corresponds to 20% of the periodicity, with a maximum of 90 days. For example, a 12-monthly task could be planned up to 72 days before or after the due date without requiring a risk assessment, since 12 months times 30 days times 20% gives 72 days. Tasks performed excessively early are called *advancements*, potentially leading to over-maintenance and increased costs, while those performed excessively late are termed *deferrals*, potentially increasing risks due to under-maintenance. Throughout the plan, the clock date of a maintenance task may require to be updated based on its planned execution. These concepts are summarized in the example of Figure 1.

Among the tasks included in the program, around 50 preventive maintenance tasks are usually identified as being bound to *certification requirements*. These tasks typically relate to systems critical for the operation of the ship and the safety of the crew at sea. Thus, unlike tasks subject to flexibility rules, certified systems are constrained to always be up to date during deployment periods, as illustrated in Figure 2.

Maintenance program planners may consider additional requirements in certain situations to formulate a feasible plan. Maintenance tasks related to the same ship system and with coinciding periodicity may need to be considered *nested*. For instance, the 12-monthly task for a ship's cooling system can include the 3-monthly fluid level check and the 6-monthly filter replacement, ensuring synchronized execution. This approach optimizes resource allocation by consolidating maintenance efforts. Furthermore, ship deployment readiness may be taken into account for each work period. This would typically involve ensuring a set of maintenance tasks are up to date before a specific deployment period according to a given level (e.g., *Minimal*, *Intermediate*, *Critical*).

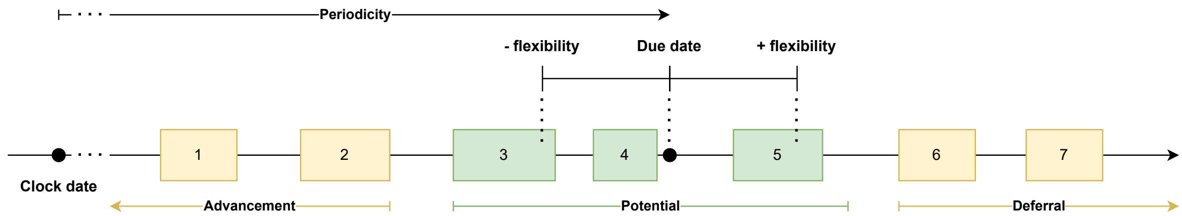

Figure 1: Example of planning a periodic preventive maintenance task with the flexibility rule.

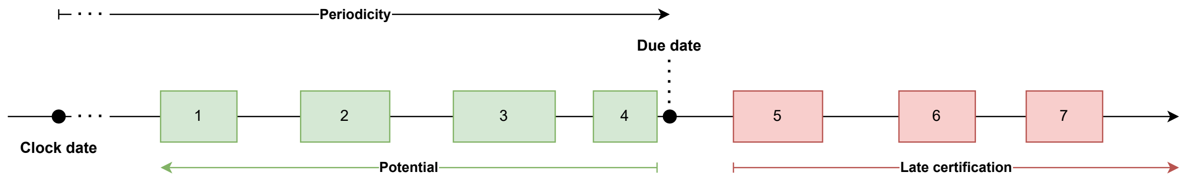

Figure 2: Example of planning a periodic preventive maintenance task with a certification requirement.

3 RELATED WORK

There has been a lot of research on maintenance optimization and planning (Van Horenbeek et al., 2010). According to (Bousdekis et al., 2019), the objective of most optimization algorithms is to reduce costs related to maintenance, risks of failure or the impact of maintenance on production and supply chain. In addition, decision-making should ideally be *predictive*, i.e. triggered by sensor-driven predictions to generate proactive maintenance plans. However, this vision of maintenance planning can hardly be applied to the ship repair and in-service support industries, where maintenance scheduling is highly constrained by defined work periods. Notably, (Ahluwalia and Pinha, 2014) point out other particularities of the ship repair industry such as large distances between ship location and ship repair facilities, high capital investment in specialized equipment, such as cranes and dry docks, and relatively short lead times. This further complicates maintenance planning and renders traditional time-driven approaches unfit for the use case considered herein. In this context, (Cullum et al., 2018) study a risk-based maintenance scheduling framework as a promising way to improve on traditional management of periodic preventive maintenance, but also note that its application is mainly limited by organizational resources. (Boudreault et al., 2022) propose a CP approach for ship refit project scheduling that produces optimized operational schedules under various time, precedence, and resource constraints. In the present work, the preventive maintenance tasks are instead scheduled at a *tactical* decision level. Indeed, tasks must be allocated to individual work periods without being “exactly” scheduled, thus each work period corresponds to an independent project remaining to be scheduled. At both the tactical and operational levels, (Deris et al., 1999) propose a constraint-based approach and a genetic algorithm to solve a Navy maintenance scheduling problem, which aims to maximize fleet availability under temporal and dockyard availability constraints. In their work, the maintenance work periods are to be determined by the results of the optimization.

Similar challenges are faced in other domains for establishing a long-term maintenance planning strategy. As stated by (Diallo et al., 2019), most military systems are designed to operate given alternating sequences of missions and scheduled breaks. The authors focused their research on the Selective Maintenance Problem (SMP), where the decision is to select which task should be performed during a certain work period to maximize the reliability of the system. They propose an integer programming approach to generate maintenance plans, offering an interesting cost and reliability trade-off. The same way military systems have to take breaks for maintenance, trains and other transportation equipment have to be removed from service to perform the most important preventive work. These tasks usually have long and uncertain periodic duration. (Gu et al., 2019) propose a genetic algorithm for deciding the trains' arrival dates at the maintenance center, tackling a problem closely related to the Single Machine Scheduling Problem (SMSP). In comparison to our use case, the imposed ship schedule prevents from choosing the arrival dates, thus we may only decide the sequence of operations. In the drilling domain, (Javanmard and al Wahhab Koraeizadeh, 2016) propose a genetic optimization approach for preventive maintenance scheduling based on reliability and cost estimation models for each of the asset components. Their approach allows to decide a maintenance periodicity minimizing costs with better overall reliability.

In light of the particularities and challenges highlighted by previous studies in preventive maintenance, see e.g. (Basri et al., 2017; Wu, 2011), we aim herein to reduce the risk of under-maintenance and the underlying cost of over-maintenance by proposing an optimization-based planning strategy. To the best of our knowledge, developing this strategy for our specific use case requires an original approach.

4 METHODOLOGY

The following outlines our proposed CP approach to solve the problem described in Section 2.

4.1 Model

Input Parameters. First of all, horizon $h \in \mathbb{Z}^{>0}$ determines the planning timeline as the set $\mathcal{T} := \{0, 1, \ldots, h\}$ of time points. Now, let $\mathcal{W} := \{1, 2, \ldots, n\}$ be the set of work periods to include in the plan. Given that each work period $w \in \mathcal{W}$ has an (inclusive) start time $s_w \in \mathcal{T}$ and an (inclusive) end time $e_w \in \mathcal{T}$, we suppose $e_w < s_{w+1}$ for $w \in \mathcal{W} \setminus \{n\}$, i.e. the work periods are chronologically ordered and non-overlapping, with $s_1 = 0$. Each work period $w \in \mathcal{W}$ is also associated with its maximal maintenance task duration $d_w^{\max} \in \mathbb{Z}^{>0}$ and its capacity of maintenance labor time, $c_w \in \mathbb{Z}^{>0}$. In order to correctly plan maintenance tasks at the end of the timeline and ensure continuity with future work periods, we consider a *virtual* work period $w' := n+1$. Indeed, it would be undesirable to plan tasks in the last work period simply because the model is not aware of an upcoming work period. Thus, we suppose this work period starts (and *virtually* ends) on time point $h+1$ and has an unlimited capacity, i.e. $s_{w'}, e_{w'} := h+1$ and $d_{w'}^{\max}, c_{w'} := \infty$. We use $\mathcal{T}' := \mathcal{T} \cup \{h+1\}$ and $\mathcal{W}' := \mathcal{W} \cup \{w'\}$ in the following.

Let $\mathcal{M}$ be the set of preventive maintenance tasks to plan. Each task $m \in \mathcal{M}$ has an initial due date $t_m^{\text{init}} \in \mathcal{T}$, which defines the first time in the planning timeline where the task needs to be executed, while next due dates are computed using its periodicity, $p_m \in \mathbb{Z}^{>0}$. Tasks that are bound to a certification, i.e. required to always be up to date, are contained in the set $\mathcal{M}^* \subseteq \mathcal{M}$. For all other tasks $m \in \mathcal{M} \setminus \mathcal{M}^*$, the acceptable execution range around the due dates is encoded with its flexibility $f_m \in \mathbb{Z}^{\geq 0}$. Finally, each maintenance task $m \in \mathcal{M}$ has a duration $d_m \in \mathbb{Z}^{>0}$ that is assumed to be equal to its required labor time.

The main decisions of the model relate to where each occurrence of a maintenance task is executed in the planning timeline. The model supposes a maintenance occurrence can be *skipped*, i.e. that a maintenance task can be assigned more than once to the same work period. However, since the task will only be executed once during that period, at least one of its periodic occurrences will appear to be "ignored". For this reason, we limit the model decisions to $k^{\max} \in \mathbb{Z}^{>0}$ occurrences for each maintenance task and define $\mathcal{K} := \{1, 2, \ldots, k^{\max}\}$ as the set of occurrences.

Variables. The main decision variables of the model are defined as $E_m^k \in \mathcal{W}'$ for the work period where occurrence $k \in \mathcal{K}$ of maintenance task $m \in \mathcal{M}$ is planned to be executed. Intermediate variables used by the model objective and constraints are defined as follows:

- $T_m^k \in \mathbb{Z}^{\geq 0}$, the computed due date for occurrence $k \in \mathcal{K}$ of maintenance task $m \in \mathcal{M}$, as a time point in $\mathcal{T}$ or later than horizon;
- $A_m^k, D_m^k, LC_m^k \in \{0, 1\}$, which are 1 if and only if occurrence $k \in \mathcal{K}$ of maintenance task $m \in \mathcal{M}$ is respectively by definition an advancement, a deferral, or an unsatisfied (*late*) certification requirement;
- $B_m^w \in \{0, 1\}$, which is 1 if and only if maintenance task $m \in \mathcal{M}$ is planned at least once in work period $w \in \mathcal{W}'$.

Objective. The global objective of the model is to minimize advancements and deferrals of maintenance occurrences at the tactical planning level, while satisfying as much as possible the certification requirements. To encode this objective, we define **target**(m, k) as the preferred work period in $\mathcal{W}'$ for planning occurrence $k \in \mathcal{K}$ of maintenance task $m \in \mathcal{M}$. This preference is to be set as a user option (see Section 4.2.1). The target is then used to compute its absolute "difference" from the chosen execution E_m^k, as a number of work periods (see Figure 3). The difference (increased by one) is multiplied by penalty weights, $w_A, w_D, w_{LC} \in \mathbb{Z}^{>0}$ if occurrence $k \in \mathcal{K}$ of maintenance task $m \in \mathcal{M}$ is respectively an advancement, a deferral, or a late certification requirement. Here, the "$+1$" ensures the penalties are applied even in the case where the chosen work period is right on target. This is relevant when the target is actually an advancement or a deferral due to the task's limited flexibility, as the best choice can be to increase the difference in order to avoid the greatest penalty. Summing over all the maintenance occurrences, the

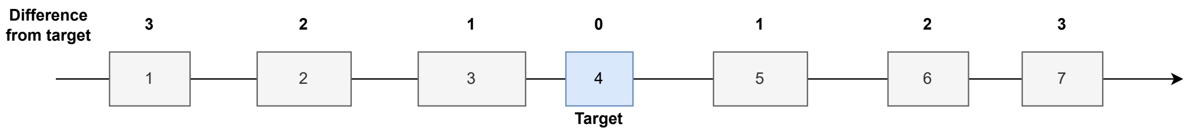

Figure 3: Illustration of how the target is used in the objective function.

objective is thus

$$\min \sum_{m\in\mathcal{M}} \sum_{k\in\mathcal{K}} w_m^k \left(\left| \mathbf{target}(m,k) - E_m^k \right| + 1 \right), \quad (1)$$

$$\text{where } w_m^k := \begin{cases} w_A & \text{if } A_m^k = 1; \\ w_D & \text{if } D_m^k = 1; \\ w_{LC} & \text{if } LC_m^k = 1; \\ 1 & \text{else.} \end{cases}$$

Constraints. The constraints of the model are presented below. Constraints (2) enforce that the initial due date of each maintenance task is as given by its parameter.

$$T_m^1 = t_m^{\text{init}}, \quad \forall m \in \mathcal{M} \quad (2)$$

Then, constraints (3) define the subsequent due dates by using the periodicity. Following the description of Section 2, we define $\mathbf{clockDate}(m,k)$ as the time point in $\mathcal{T}'$ from which to deduce the due date for occurrence $k \in \mathcal{K}$ of maintenance task $m \in \mathcal{M}$. The computation rule is to be set as user options (see Section 4.2.2).

$$T_m^k = \mathbf{clockDate}(m,k) + p_m, \quad \forall m \in \mathcal{M}, \forall k \in \mathcal{K} \setminus \{1\} \quad (3)$$

Constraints (4) make sure the **clockDate** setting creates an increasing sequence of due dates for each maintenance task, thus avoiding the decisions to be "blocked" in time, except when the current execution is at the virtual work period. Similarly, constraints (5) make sure the occurrences of each maintenance task are assigned in a chronological order of work periods. Here, the equal case would correspond to a skipped execution.

$$E_m^k < w' \implies T_m^k < T_m^{k+1}, \quad \forall m \in \mathcal{M}, \forall k \in \mathcal{K} \setminus \{k^{\max}\} \quad (4)$$

$$E_m^k \leq E_m^{k+1}, \quad \forall m \in \mathcal{M}, \forall k \in \mathcal{K} \setminus \{k^{\max}\} \quad (5)$$

For maintenance tasks that are not related to a certification ($m \in \mathcal{M} \setminus \mathcal{M}^*$), constraints (6) encode when the occurrence of a task is an advancement, i.e. when the chosen work period (E_m^k) is neither virtual nor ending earlier than its acceptable flexibility before the due date. Similarly, constraints (7) encode deferrals, i.e. when the chosen work period starts later than the acceptable flexibility after the due date. In the case of certifications, constraints (8) simply set the associated variables to 0.

$$A_m^k = 1 \iff E_m^k < w' \land e_{E_m^k} < T_m^k - f_m, \quad \forall m \in \mathcal{M} \setminus \mathcal{M}^*, \forall k \in \mathcal{K} \quad (6)$$

$$D_m^k = 1 \iff s_{E_m^k} > T_m^k + f_m, \quad \forall m \in \mathcal{M} \setminus \mathcal{M}^*, \forall k \in \mathcal{K} \quad (7)$$

$$A_m^k = 0 \land D_m^k = 0, \quad \forall m \in \mathcal{M}^*, \forall k \in \mathcal{K} \quad (8)$$

Constraints (9) and (10) encode when occurrences do not satisfy certification requirements. This happens when a maintenance task bound to a certification ($m \in \mathcal{M}^*$) is planned in a work period where its start time exceeds the current due date.

$$LC_m^k = 1 \iff s_{E_m^k} > T_m^k, \quad \forall m \in \mathcal{M}^*, \forall k \in \mathcal{K} \quad (9)$$

$$LC_m^k = 0, \quad \forall m \in \mathcal{M} \setminus \mathcal{M}^*, \forall k \in \mathcal{K} \quad (10)$$

Constraints (11) ensure each maintenance task fits according to duration in the work periods where it is planned to be executed.

$$d_m \leq d_{E_m^k}^{\max}, \quad \forall m \in \mathcal{M}, \forall k \in \mathcal{K} \quad (11)$$

Variables B_m^w are defined by constraints (12) using the COUNT global constraint. The latter simply counts how many times the boolean statement is revealed to be true, here how many times maintenance task $m \in \mathcal{M}$ is planned in work period $w \in \mathcal{W}$ among the occurrences $\mathcal{K}$. Finally, these variables are used by constraints (13) to force each work period in having a total labor time (as the sum of d_m where B_m^w is 1, $m \in \mathcal{M}$) respecting its capacity.

$$B_m^w = 1 \iff \textsc{Count}(k \in \mathcal{K}, E_m^k = w) \geq 1 \quad \forall m \in \mathcal{M}, \forall w \in \mathcal{W}' \quad (12)$$

$$\sum_{m\in\mathcal{M}} d_m B_m^w \leq c_w, \quad \forall w \in \mathcal{W}' \quad (13)$$

4.2 User Options

The model of Section 4.1 involves many user options that affect its definition and thus its behavior. These options have been considered in order to keep a certain flexibility in the developed approach. This allowed initial users to maintain their current planning habits, while providing them with new (and potentially better) possibilities.

Table 1: Available user options to define **target**(m,k).

Option	Definition
`Closest`	Closest work period *before* or *after* the due date T_m^k.
`Latest`	Latest work period in the acceptable execution range, i.e. closest *before* $T_m^k + f_m$.

4.2.1 Target Options

First, for certification tasks $m \in \mathcal{M}^*$ and occurrences $k \in \mathcal{K}$, **target**(m,k) is always defined as the closest work period *before* the due date T_m^k. In Figure 2, this would correspond to work period 4.

Then, for maintenance tasks $m \in \mathcal{M} \setminus \mathcal{M}^*$, **target**$(m,k)$ must be defined by one of the choices described in Table 1. Option `Closest` focuses on following periodicity recommendations as much as possible. In Figure 1, this would correspond to work period 4. Option `Latest` focuses instead on finding opportunities in safety recommendations to overall reduce the number of occurrences. In Figure 1, this would correspond to work period 5. In our implementation, we use a pre-computed function that maps each time point in $\mathcal{T}'$ to its closest (*before*/*after*) work period in $\mathcal{W}'$.

4.2.2 Clock Date Options

For maintenance tasks $m \in \mathcal{M}$ and occurrences $k \in \mathcal{K} \setminus \{1\}$, **clockDate**$(m,k)$ is defined by selecting one option from each of the two sets of options described below. As a basis, the clock date usually corresponds to the due date of the last occurrence, T_m^{k-1}, so that occurrence k is due on $T_m^{k-1} + p_m$ based on constraints (3). However, in the usual case where a planned occurrence does not exactly match its due date, this considered clock date is at high risk of causing non-compliant periodicity intervals. Thus, there is a need to *update* it to the planned execution date. This is particularly true for tasks that are bound to a certification, $m \in \mathcal{M}^*$, which always have their clock date updated. For other tasks $m \in \mathcal{M} \setminus \mathcal{M}^*$, the update condition must be selected from the choices described in Table 2. Option `Never` simply follows the given initial due date, regardless of the model decisions. Alternatively, option `A/D` updates the clock date when the last occurrence is planned outside its flexibility, i.e. when $A_m^{k-1} = 1$ or $D_m^{k-1} = 1$. Finally, option `Always` systematically updates the clock date at each occurrence, which is in theory the best way to follow periodicity recommendations and avoid constant replanning efforts.

Table 2: Available user options to define the update condition of **clockDate**(m,k).

Option	Definition
`Never`	Clock date never updated.
`A/D`	Clock date only updated when last occurrence is an advancement or a deferral.
`Always`	Clock date always updated at each occurrence.

Table 3: Available user options to define the updated date of **clockDate**(m,k).

Option	Definition
`Start`	Work period start date, $s_{E_m^{k-1}}$.
`Mid`	Work period midpoint date, $\lfloor 1/2 (s_{E_m^{k-1}} + e_{E_m^{k-1}}) \rfloor$.
`End`	Work period end date, $e_{E_m^{k-1}}$.

Since the produced maintenance plan is at the tactical level, execution dates of tasks are not known within each work period. Thus, when updating the clock date, the chosen new date must be selected according to the choices described in Table 3. Option `Start` is the most conservative option, since it assumes the whole work period duration should be included in the next due date computation. Alternatively, option `Mid` will on average be closer to the true execution. In our implementation, the midpoint dates are simply pre-computed. Finally, option `End` is the most opportunist option, since it assumes the clock is only updated at the next ship deployment. An example of clock date update using option `Start` is illustrated in Figure 4.

4.3 Optional Constraints

The model presented in Section 4.1 has been extended with optional constraints to better suit the maintenance program planning reality described in Section 2. First, users may specify if maintenance tasks are to be considered nested (in the following, option `Nested`). If so, these constraints are included in the model by encoding an implication on nested maintenance task executions, using variables B_m^w. Then, constraints on the ship deployment readiness can be added. As further input parameters, we consider a set of readiness levels, while each maintenance task can be associated with a subset of these levels. These constraints are included in the model by restricting the values of E_m^k. Finally, in the user interface of the solution, users are able to override the optimizer decisions by forcing occurrences to be planned (or not be planned) in some work periods. Thus, when re-optimizing, the model must consider these additional

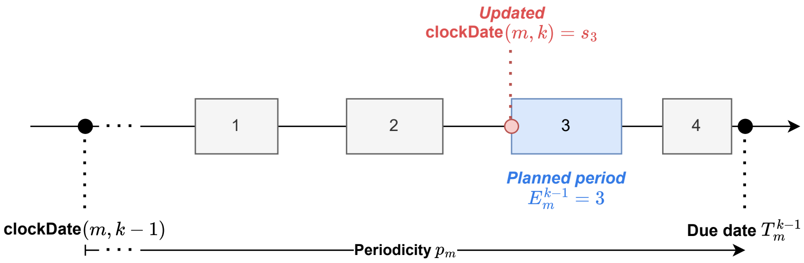

Figure 4: Example of updating the clock date for occurrence k of maintenance task m using user option `Start`. Here, the planned work period for occurrence $k-1$ was 3, thus the clock date for occurrence k will be set to s_3.

constraints, which are encoded by forcing the values of B_m^w.

4.4 Search Heuristic

In our approach, we consider the following search heuristic for the CP model of Section 4.1. Each pair of maintenance task $m \in \mathcal{M}$ and occurrence $k \in \mathcal{K}$ is selected in input order, i.e. $(m,k) \in ((1,1),(1,2),\ldots,(1,k^{\max}),(2,1),\ldots,(2,k^{\max}),\ldots,(|\mathcal{M}|,k^{\max}))$. Then, it assigns, in order, D_m^k to its smallest possible value and the difference to the target from objective (1), $\left|\textbf{target}(m,k) - E_m^k\right|$, to its smallest possible value, both prioritizing an assignment to 0 first. In other words, this heuristic focuses on executing the tasks in their targeted work period or around it, which directly relates to the objective function, and prefers initially avoiding deferrals. The latter preference comes into play in cases where work periods around the target may involve either an advancement or a deferral, thus will explore advancement executions beforehand, assuming its associated penalty weight w_A is smaller than the one associated to deferrals, w_D. The resulting heuristic can be formulated with the *priority search* annotation from the MiniZinc modeling language (Feydy et al., 2017; Nethercote et al., 2007), and is supported by the Chuffed solver (Chu, 2011).

5 EXPERIMENTS

The main objective of the experiments is to evaluate the performance of our CP approach from Section 4 on a real naval use case, including all of our considered options, while comparing it to a currently used planning method performed in Excel. This method, which we call `Heuristic` in the following, systematically assigns the maintenance task occurrences to their closest work period *before* their due date. For clock updates, it replicates options `Never` and `Start` from Section 4.2.2. Thus, `Heuristic` is prone to generate unnecessary advancements, requiring constant replanning effort, while ignoring over-maintenance reduction opportunities. Furthermore, it ignores any constraint related to work period capacities, i.e. constraints (11) and (13), and all optional constraints from section 4.3 due to a current partial planning approach. Although this might allow `Heuristic` to produce unfeasible solutions, in contrast to the CP approach, it serves as a valuable experimental comparison point as we demonstrate in the following.

5.1 Benchmark

Our benchmark consists of five instances created from a real maintenance program dataset for one ship. This dataset contains the complete description of 707 preventive maintenance tasks for different systems over various functional locations of the ship. Among these tasks, around 50 are related to certified systems (in $\mathcal{M}^*$). The periodicity of tasks p_m ranges from 1 to 180 months (avg. 26), while they have a duration d_m between 0.25 and 240 hours (avg. 10). Their flexibility f_m is fixed to 20% of the periodicity, in days, with a maximum of 90 days, which leads to values between 6 and 90 days (avg. 69). Each maintenance task is given its initial due date t_m^{init} based on past execution dates, which may fall in the planning timeline depending on the chosen horizon. The dataset also describes the ship schedule over five years, including 14 Short Work Periods (SWPs) of between 14 and 39 days (avg. 24), as well as three Dry-Dock Work Periods, DDWP-1, DDWP-2, and DDWP-3 of respectively 68, 86 days, and 123 days. To allow comparison between short-term and long-term planning, we generated instances with a varied number of work periods. For each instance, the horizon h is fixed to the start date of the following not included work period (if possible, else 30 days after the last work period). Thus, we considered:

- `1-year`, including DDWP-1 and 3 SWPs, leading to 493 tasks in the planning timeline;
- `2-years`, including `1-year`'s work periods and

four additional SWPs, leading to 497 tasks in the planning timeline;

- `3-years`, including `2-years`' work periods, two additional SWPs, and DDWP-2, leading to 605 tasks in the planning timeline;
- `4-years`, including `3-years`' work periods, two additional SWPs and DDWP-3, leading to 669 tasks in the planning timeline;
- `5-years`, including `4-years`' work periods and two additional SWPs, leading to 684 tasks in the planning timeline.

Note that the dataset allows to deduce nested maintenance tasks. On our instances, this may involve up to 598 implied execution constraints using option `Nested` (see Section 4.3).

Since some inputs are missing from the dataset, we make the following additional assumptions based on current planning habits:

- Each work period has a maximal task duration $d_w^{\max}$ based on 8 hours per working day (excluding weekends), thus ranging from 88 to 712 hours (avg. 227);
- Each work period has a labor-time capacity c_w of 320 hours per working day (excluding weekends), leading to values between 3200 and 28400 hours (avg. 9000);
- No optional ship deployment readiness constraints are considered;
- No optional planning override constraints are considered.

5.2 Experimental Setup

We implemented the CP approach from Section 4 in the MiniZinc 2.8.3 language (Nethercote et al., 2007) and used Chuffed 0.13.1 as solver (Chu, 2011). User options of Section 4.2 were implemented using "if-then-else" expressions and intermediate variables representing the different cases. Furthermore, we implemented the above-described `Heuristic` scheme in Python 3.11. The experiments were performed on a MSI GP63 Leopard 8RE machine with an Intel i7-8750H CPU at 2.2 GHz, 6 cores and 32 GB of RAM in a WSL environment. Each optimization execution was given a timeout of 30 minutes, while the number of occurrences to plan was limited to the number of work periods, i.e. $k^{\max} := n$. We also fixed the objective penalty weights as $(w_A, w_D, w_{LC}) := (2, 5, 100)$. These weights were empirically determined using iterative feedback from planners.

For our experiments, each instance was solved by the CP approach for each combination of user options from Section 4.2, i.e. (`Closest` or `Latest`) and (`Never` or `A/D` or `Always`) and (`Start` or `Mid` or `End`). Additionally, each of these configurations was tested using (or not) option `Nested` from Section 4.3. In the following, we denote `Not-Nested` for the case where option `Nested` is not used. Each instance was also solved using `Heuristic`, where the generated solution was evaluated according to the `Closest` and `Latest` options.

5.3 Results

Detailed results including each instance and each option configuration are available in Appendix. Table 4 summarizes these results for each instance using `Heuristic` and the CP approach for the `Not-Nested` and `Nested` cases, averaged over all option configurations. As stated before, the objective of the model is to minimize advancements and deferrals, while satisfying the certification requirements, according to a user-defined target preference. Results show that using `Heuristic` creates solutions with at least 21% of occurrences being strictly advancements, while the CP approach always finds solutions with less than 17% being advancements and deferrals. The total reduction is on average 67% and can go up to 93% on `4-years` with `Closest-Always-End-Not-Nested`. We also observe that the number of late certifications is always no more than one, while late occurrences are only generated by the CP approach on the `5-years` instance.

Results on `Not-Nested` configurations show that using the CP approach instead of `Heuristic` always reduces the objective value, with an average reduction of 17% and 21% with `Closest` and `Latest`, respectively. The most noticeable improvement is with the `1-year` instance and the `Latest` option, where the objective function is reduced by up to 31%. However, on `4-years` and `5-years` instances, using `Always` and `Start` reduces the objective value simply by up to 2%. Overall, the best configuration varies depending on the size of the instance. For larger instances (`3-years` and more), the smallest objective value is obtained using `Closest-Always-End`. For the `1-year` instance, it is rather obtained using `Mid`, while for the `2-years` instance, it is obtained using `Latest` and `Start` instead. In both cases, the difference between the best solution and the solution obtained using `Closest-Always-End` is less than 1%.

Considering the `Nested` option, the objective value increases by up to 5% compared to `Not-Nested`. For all instances, the `Closest` and

Table 4: Summarized results obtained on the benchmark instances using `Heuristic` and the CP approach, on average, with `Not-Nested` or `Nested` options. Objective value of the best solution found (**Obj.**), its number of occurrences in $\mathcal{T}$ (**#O**), advancements (**#A**), and deferrals (**#D**).

Instance	Heuristic				Not-Nested				Nested			
	Obj.	**#O**	**#A**	**#D**	**Obj.**	**#O**	**#A**	**#D**	**Obj.**	**#O**	**#A**	**#D**
1-year	2878	849	215	0	2008	679	12	1	2145	707	47	2
2-years	2587	755	162	0	2073	632	48	1	2159	650	67	2
3-years	8565	2340	488	0	6924	2236	84	12	7458	2294	183	30
4-years	12 201	3165	697	0	10 186	3124	167	20	10 919	3205	313	46
5-years	14 155	3570	752	0	11 850	3522	173	50	12 805	3643	337	82

Table 5: Average computation time, in seconds, to find an initial solution using option `Always`.

Instance	Nested			Not-Nested		
	Start	**Mid**	**End**	**Start**	**Mid**	**End**
1-year	9	6	5	5	7	5
2-years	6	5	5	5	5	5
3-years	191	88	80	57	39	36
4-years	633	269	247	114	66	66
5-years	1088	461	399	166	80	75

`Always` options similarly lead to a smaller objective value. In contrast to the `Not-Nested` configurations, there is no tendency on the way the clock date should be chosen (`Start`, `Mid`, or `End`). Due to the additional constraints, we also observe that using `Always-Start` for the `4-years` and `5-years` instances leads to a greater objective value than using `Heuristic`.

Considering the time performance of our approach, we observe that at least one solution is found for all instances and configurations. Around 60% of the CP executions ended with exactly one solution, while for the others, five intermediate solutions are found on average. Intermediate solutions never reduce the objective value more than 1.4% before the timeout is reached. Furthermore, 39 CP executions over 180 (22%) ended with an optimality proof, but only for `1-year` and `2-years` instances.

Table 5 reports the average computation time required to find an initial solution using option `Always`, in seconds. We omit `A/D` and `Never` here, since they produce similar results. We observe that the time varies from 5 seconds to 18 minutes depending on the instance size and selected options. While for smaller instances (`1-year`, `2-years`) the difference is negligible, instances `3-years` and more show a noticeable increase in the time required, with the longest scenario being the `5-years` instance solved with `Start-Always-Nested`.

6 DISCUSSION

The main goal of this research was to create Maintenance Optimizer™, a solution developed to increase operational availability of ships while decreasing in-service support costs. Building a feasible preventive maintenance plan over a long-term horizon satisfying every constraint is known to be a complex and time-consuming task. Usability assessments with domain experts allowed us to estimate that creating a one-year plan takes a minimum of 4 hours (scattered over 2 weeks to minimize cognitive overload) without any compliance verification. Thus, our experiments demonstrated that we can significantly reduce this process, while guaranteeing rules compliance.

In addition to the number of advancements and deferrals, we were also interested in reducing over-maintenance by looking at the proposed number of occurrences. Unsurprisingly, selecting the latest possible work period to perform the maintenance tasks using `Latest` leads to fewer occurrences. Figure 5 highlights this behavior, by comparing the distribution of maintenance executions in relation to their due date. We observe that around 75% of occurrences are planned on or after their due date using `Latest`, compared to 65% with `Closest`, and less than 40% with `Heuristic`. The fact that this is not reflected in the result, since the `Closest` option offers the smallest objective value in almost all cases, suggests the need for further sensitivity analyses. When looking more in depth at the difference between `Closest` and `Latest`, it is apparent that the plans made by selecting the latest acceptable work period requires more advancements than the plan made by selecting the one closest to the due date. However, in the majority of cases, using `Latest` creates a similar number of deferrals or less. Overall, compared to `Heuristic`, using the CP approach reduces the number of occurrences by up to 25%. This potential reduction yet remains to be

Figure 5: Average distribution of occurrences in relation to their due date, in percentage, for `Heuristic`, `Closest`, and `Latest`.

evaluated in terms of monetary savings for in-service support.

An important observation is that our CP approach introduces deferrals where the heuristic does not. Even if deferrals are never more than 8% of all occurrences, this is an aspect where the model could be improved and that should be validated with users. Indeed, these deferrals may be avoided by potentially creating more advancements. Thus, the penalty weights in the objective function may have to be adjusted to offer solutions with even fewer deferrals. Furthermore, the `Never` option seems to offer solutions with fewer deferrals than the `A/D` and `Always` options. Indeed, the results show that every `Not-Nested` and `Never` cases have not a single deferral in their solution. This is however expected, since `Never` is allowed to cause non-compliant periodicity intervals due to its clock date (and target) definition. Deferrals with the `A/D` and `Always` options can be explained by maintenance tasks with smaller periodicity, where it is sometimes impossible to execute them on time when the clock date is updated since the work periods are so far apart. The number of deferrals is even higher starting from instance `3-years` where there are only two to three work periods per year.

During our experiments, we observed the current computation limitations of our approach. Even though we always found an initial solution using our search heuristic, 16 CP executions over 180 (8.9%) terminated with an "ERROR" status before the timeout due to the solver reaching its maximal allocated memory (RAM) during the solving phase, i.e. more than 30 GB. These cases were encountered on a subset of `4-years` and `5-years` configurations. This increase in memory requirements is notably due to the number of variables and constraints contained in the generated FlatZinc file given to the solver. On the `5-years` instance, it contained on average 433K variables and 480K constraints, while requiring a compilation time between 13 and 25 seconds. In comparison, on the `1-year` instance, it contained 53K variables and 61K constraints with a compilation time of 1 or 2 seconds. Thus, it may be worthwhile to test the approach on a different setup or with a different solver. Note that we did try the OR-Tools CP-SAT solver (Perron and Furnon, 2024) via its FlatZinc implementation, but preliminary results showed a greater computation time for similar or worse solutions. Furthermore, only Chuffed could directly support the *priority search* MiniZinc annotation we used to construct our search heuristic. Other search heuristics we considered before using this one include a similar one that did not assigned variables D_m^k, as well as a simple sequential search on distance $\left|\textbf{target}(m,k) - E_m^k\right|$ exploring work periods with $E_m^k \leq \textbf{target}(m,k)$ first (towards "left"). Following the approach of (Boudreault et al., 2022), we also tried the free search of Chuffed, as well as the *Solution-Based Phase Saving* (SBPS) value-selection heuristic (Demirović et al., 2018), but observed again greater computation times for similar or worse solutions.

7 CONCLUSION

In this paper, we introduced a CP approach for a preventive maintenance planning problem in the naval domain. Our solution was validated on a benchmark of instances with varying planning horizons created from a real ship maintenance program dataset. Each instance was solved according to a combination of user options that changes the behavior of the CP model, and compared to a currently used planning method. Our approach clearly demonstrated its worth in long-term maintenance planning, cutting the considered objective value by up to 31%, resulting in a significant decrease of 93% in advancements and deferrals. In addition, we have shown that the CP approach can greatly reduce over-maintenance, and thus in-service support costs, with plans having up to 25% less maintenance occurrences.

Directions for future work include tackling the current observed limitations of the CP model, such as time and memory scaling issues as well as objective function unwanted behavior (e.g., the fact that deferrals are introduced), notably through a rolling horizon approach. We also aim at comparing this approach to an optimization algorithm based on mixed-integer programming, as well as going towards multi-objective optimization by integrating an objective of resource leveling. Finally, we plan to evaluate our results in terms of in-service support cost savings and address an extended use case which proposes modifications to the given ship schedule.

ACKNOWLEDGEMENTS

Thanks are due to the many members of the Thales project team, as well as our collaborators from Université Laval, Polytechnique Montréal, and Dalhousie University, for their invaluable feedback on the problem definition and for the solution implementation.

REFERENCES

Ahluwalia, R. and Pinha, D. (2014). Decision support system for production planning in the ship repair industry. *Industrial and Systems Engineering Review*, 2(1):52–61.

Basri, E. I., Abdul Razak, I. H., Ab-Samat, H., and Kamaruddin, S. (2017). Preventive maintenance (PM) planning: A review. *Journal of Quality in Maintenance Engineering*, 23(2):114–143.

Boudreault, R., Simard, V., Lafond, D., and Quimper, C.-G. (2022). A constraint programming approach to ship refit project scheduling. In Solnon, C., editor, *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*, volume 235 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:16, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Bousdekis, A., Lepenioti, K., Apostolou, D., and Mentzas, G. (2019). Decision making in predictive maintenance: Literature review and research agenda for industry 4.0. *IFAC-PapersOnLine*, 52(13):607–612.

Chu, G. G. (2011). *Improving Combinatorial Optimization*. PhD thesis, The University of Melbourne. GitHub: https://github.com/chuffed/chuffed.

Cullum, J., Binns, J., Lonsdale, M., Abbassi, R., and Garaniya, V. (2018). Risk-based maintenance scheduling with application to naval vessels and ships. *Ocean Engineering*, 148:476–485.

Demirović, E., Chu, G., and Stuckey, P. J. (2018). Solution-based phase saving for CP: A value-selection heuristic to simulate local search behavior in complete solvers. In Hooker, J., editor, *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pages 99–108, Cham. Springer International Publishing.

Deris, S., Omatu, S., Ohta, H., Shaharudin Kutar, L. C., and Abd Samat, P. (1999). Ship maintenance scheduling by genetic algorithm and constraint-based reasoning. *European Journal of Operational Research*, 112(3):489–502.

Diallo, C., Khatab, A., and Venkatadri, U. (2019). Developing a bi-objective imperfect selective maintenance optimization model for multicomponent systems. *IFAC-PapersOnLine*, 52(13):1079–1084.

Feydy, T., Goldwaser, A., Schutt, A., Stuckey, P. J., and Young, K. D. (2017). Priority Search with MiniZinc. In *ModRef 2017: The Sixteenth International Workshop on Constraint Modelling and Reformulation*.

Gu, H., Joyce, M., Lam, H. C., Woods, M., and Zinder, Y. (2019). A genetic algorithm for assigning train arrival dates at a maintenance centre. *IFAC-PapersOnLine*, 52(13):957–962.

Javanmard, H. and al Wahhab Koraeizadeh, A. (2016). Optimizing the preventive maintenance scheduling by genetic algorithm based on cost and reliability in National Iranian Drilling Company. *Journal of Industrial Engineering International*, 12(4):509–516.

Lafond, D., Couture, D., Delaney, J., Cahill, J., Corbett, C., and Lamontagne, G. (2021). Multi-objective schedule optimization for ship refit projects: Toward geospatial constraints management. In Ahram, T., Taiar, R., and Groff, F., editors, *Human Interaction, Emerging Technologies and Future Applications IV*, Advances in Intelligent Systems and Computing, pages 662–669, Cham. Springer International Publishing.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., and Tack, G. (2007). MiniZinc: Towards a standard CP modelling language. In Bessière, C., editor, *Principles and Practice of Constraint Programming – CP 2007*, Lecture Notes in Computer Science, pages 529–543, Berlin, Heidelberg. Springer. Website: https://www.minizinc.org/.

Perron, L. and Furnon, V. (2024). OR-Tools. Google. Website: https://developers.google.com/optimization/.

Rossi, F., van Beek, P., and Walsh, T., editors (2006). *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier.

Van Horenbeek, A., Pintelon, L., and Muchiri, P. (2010). Maintenance optimization models and criteria. *International Journal of System Assurance Engineering and Management*, 1(3):189–200.

Wu, S. (2011). Preventive maintenance models: A review. In Tadj, L., Ouali, M.-S., Yacout, S., and Ait-Kadi, D., editors, *Replacement Models with Minimal Repair*, pages 129–140. Springer, London.

APPENDIX

Detailed results of the experiments from Section 5 are presented in Table 6. For each instance and each option configuration, we report the objective value of the best solution found (**Obj.**), as well as its number of occurrences in the timeline $\mathcal{T}$ (**#O**), advancements (**#A**), deferrals (**#D**), and late certifications (**#LC**). We also report the number of intermediate solutions found (**#S**), along with the solving time (**Time**) to find the best solution, in seconds. A "*" next to a solving time value indicates that the instance was optimally solved before the timeout was reached. For each instance, `Closest`/`Latest`, and `Not-Nested`/`Nested` configuration, we highlight in bold the smallest objective value and number of occurrences obtained with the different clock date user options.

Table 6: Results obtained on the benchmark instances using `Heuristic` and the CP approach.

Instance and User Options				Not-Nested							Nested						
				Obj.	#O	#A	#D	#LC	#S	Time	Obj.	#O	#A	#D	#LC	#S	Time
1-year	Closest	Heuristic		2723	849	215	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	1995	688	13	0	0	1	*5	2003	689	14	0	0	7	*7
			Mid	1995	687	13	0	0	1	*3	2003	688	14	0	0	7	*8
			End	1995	683	13	0	0	1	*4	2004	685	14	0	0	7	*8
		A/D	Start	2008	693	18	1	0	1	*8	2022	692	18	2	0	7	*525
			Mid	2013	691	18	0	0	2	*9	2032	693	21	1	0	4	618
			End	2014	683	8	1	0	6	*264	2035	687	13	2	0	10	834
		Always	Start	2085	780	0	11	0	2	*14	2300	845	68	12	0	2	10
			Mid	**1986**	682	0	0	0	2	*5	**2002**	679	0	1	0	13	*17
			End	2007	**666**	21	0	0	2	*6	2030	**667**	24	1	0	10	1487
	Latest	Heuristic		2878	849	215	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	1988	665	13	0	0	1	*5	2220	691	75	0	0	1	5
			Mid	1988	664	13	0	0	1	*5	2220	690	75	0	0	1	5
			End	1988	660	13	0	0	1	*6	2221	**687**	75	0	0	1	5
		A/D	Start	2001	670	18	1	0	1	*8	2237	729	79	2	0	1	9
			Mid	2006	668	18	0	0	2	*10	2242	726	79	1	0	2	10
			End	2002	665	18	1	0	1	*8	2239	724	79	2	0	1	8
		Always	Start	2086	663	0	11	0	2	*33	2316	721	59	12	0	2	12
			Mid	**1987**	661	0	0	0	2	*9	**2219**	718	59	1	0	2	8
			End	2008	**647**	21	0	0	2	*9	2256	710	85	1	0	2	8
2-years	Closest	Heuristic		2505	755	162	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	2138	658	70	0	0	1	*6	2157	662	69	4	0	10	*110
			Mid	2138	658	70	0	0	1	*3	2157	662	69	4	0	10	*111
			End	2138	**655**	70	0	0	1	*3	2157	**659**	69	4	0	10	*81
		A/D	Start	2093	663	49	2	0	1	8	2108	669	52	2	0	1	6
			Mid	2099	663	54	2	0	1	6	2114	669	57	2	0	1	7
			End	2094	660	54	2	0	1	6	2109	666	57	2	0	1	6
		Always	Start	**2038**	666	10	0	0	2	*7	**2050**	677	10	3	0	6	*59
			Mid	2058	666	30	0	0	2	*15	2075	678	32	3	0	3	1588
			End	2048	656	22	3	0	1	*4	2061	665	23	3	0	6	1614
	Latest	Heuristic		2587	755	162	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	2065	609	70	0	0	1	*5	2226	628	109	0	0	8	1694
			Mid	2065	609	70	0	0	1	*5	2228	629	111	0	0	7	1264
			End	2065	606	70	0	0	1	*4	2228	626	111	0	0	7	1276
		A/D	Start	2056	614	49	2	0	1	8	2210	646	87	2	0	1	7
			Mid	2062	614	54	2	0	1	7	2220	646	96	2	0	1	8
			End	2062	611	54	2	0	1	8	2220	643	96	2	0	1	8
		Always	Start	**2023**	595	10	0	0	2	*9	**2148**	629	34	3	0	4	63
			Mid	2043	595	30	0	0	2	*17	2190	629	65	3	0	2	34
			End	2035	**582**	22	0	0	2	*13	2202	**621**	67	3	0	2	30
3-years	Closest	Heuristic		8318	2340	488	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	6929	2280	118	0	0	1	14	7239	2295	120	33	0	9	1002
			Mid	6929	2279	118	0	0	1	21	7239	2294	120	33	0	9	975
			End	6929	2276	118	0	0	1	21	7248	2290	121	33	0	9	1068
		A/D	Start	7255	2312	156	16	0	1	37	7608	2350	187	55	0	6	1688
			Mid	6958	2282	84	8	0	1	40	7235	2310	115	33	0	8	661
			End	6951	2283	76	9	0	3	195	7144	2308	100	25	0	4	211
		Always	Start	7155	2377	24	52	0	2	106	7979	2517	232	50	0	5	458
			Mid	6812	2253	21	20	0	1	42	**7007**	2295	66	18	0	4	182
			End	**6768**	**2047**	34	7	0	2	1524	7018	**2105**	99	6	0	4	175

Instance and User Options				Not-Nested Obj.	#O	#A	#D	#LC	#S	Time	Nested Obj.	#O	#A	#D	#LC	#S	Time
3-years	Latest	Heuristic		8565	2340	488	0	0	1	-	-	-	-	-	-	-	-
3-years	Latest	Never	Start	6806	2235	118	0	0	1	30	7639	2263	265	23	0	1	32
			Mid	6806	2234	118	0	0	1	38	7639	2262	265	23	0	1	38
			End	6806	2231	118	0	0	1	36	7648	2258	266	23	0	1	38
		A/D	Start	6995	2267	156	16	0	1	39	7977	2390	329	52	0	5	877
			Mid	6853	2237	84	8	0	1	34	7830	2362	294	30	0	10	1604
			End	6844	2238	78	8	0	1	31	7758	2357	301	23	0	5	893
		Always	Start	7182	2219	29	51	0	3	177	7560	2305	145	49	0	6	514
			Mid	6877	2187	22	20	0	1	36	**7238**	2237	110	18	0	4	231
			End	**6782**	**2006**	35	8	0	1	34	7241	**2085**	158	5	0	5	298
4-years	Closest	Heuristic		12 035	3165	697	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	10 052	3097	155	0	0	1	24	10 582	3131	210	40	0	4	1384
			Mid	10 052	3096	155	0	0	1	23	10 779	3124	207	41	0	5	1491
			End	9853	3081	155	0	0	1	22	10 587	3112	208	41	0	5	1791
		A/D	Start	10 515	3203	235	29	0	1	53	10 784	3270	285	87	0	1	69
			Mid	10 190	3145	161	11	0	1	65	10 655	3201	237	47	0	1	56
			End	9816	3086	92	11	0	1	49	10 162	3132	149	38	0	3	223
		Always	Start	11 816	3327	181	92	0	1	117	12 501	3482	397	88	0	6	848
			Mid	10 701	3281	335	23	0	1	64	11 015	3317	407	24	0	1	278
			End	**9514**	**2929**	34	11	0	1	63	**9925**	**3009**	150	8	0	5	640
	Latest	Heuristic		12 201	3165	697	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	9815	3088	155	0	0	1	61	10 960	3134	373	37	0	1	58
			Mid	9815	3087	155	0	0	1	58	11 159	3128	373	37	0	1	57
			End	9616	3072	155	0	0	1	54	10 967	3116	374	37	0	1	57
		A/D	Start	10 321	3194	235	29	0	1	58	11 344	3368	432	88	0	1	76
			Mid	9924	3136	161	11	0	1	65	11 174	3319	468	49	0	1	76
			End	9662	3077	92	11	0	1	62	10 800	3234	355	37	0	1	75
		Always	Start	12 021	3248	186	92	0	1	110	12 271	3335	322	85	0	9	865
			Mid	10 137	3216	336	23	0	1	69	10 620	3274	453	24	0	1	259
			End	**9529**	**2866**	35	11	0	1	69	**10 259**	**3005**	236	12	0	1	252
5-years	Closest	Heuristic		13 840	3570	752	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	11 675	3592	166	0	1	1	30	12 651	3637	225	56	1	5	1467
			Mid	11 675	3590	166	0	1	1	45	12 644	3633	223	56	1	5	1575
			End	11 476	3574	166	0	0	1	31	12 454	3618	227	55	0	4	1570
		A/D	Start	12 123	3667	235	57	1	1	66	12 857	3786	325	128	1	1	74
			Mid	11 805	3602	161	11	1	1	66	12 466	3726	243	71	1	1	74
			End	11 511	3569	111	13	0	1	60	12 124	3658	229	55	0	1	70
		Always	Start	13 650	3836	181	285	1	1	164	14 678	4077	419	280	1	7	1282
			Mid	12 530	3649	335	59	1	1	70	13 059	3761	459	60	1	1	471
			End	**11 145**	**3260**	34	25	0	1	66	**11 681**	**3382**	182	26	0	1	397
	Latest	Heuristic		14 155	3570	752	0	0	1	-	-	-	-	-	-	-	-
		Never	Start	11 415	3433	166	0	1	1	77	12 825	3483	395	36	1	1	78
			Mid	11 415	3431	166	0	1	1	76	12 819	3480	393	36	1	1	72
			End	11 216	3415	166	0	0	1	79	12 627	3464	394	36	0	1	75
		A/D	Start	11 953	3538	235	57	1	1	74	13 223	3721	438	112	1	1	90
			Mid	11 514	3473	161	11	1	1	75	12 831	3671	468	57	1	1	91
			End	11 294	3440	111	13	0	1	79	12 528	3628	394	45	0	1	82
		Always	Start	13 914	3714	186	285	1	1	169	14 436	3861	329	276	1	11	1342
			Mid	12 030	3565	336	59	1	1	90	12 570	3632	466	60	1	1	451
			End	**11 215**	**3214**	35	25	0	1	85	**12 014**	**3359**	255	26	0	1	402