Actor-based Large Neighborhood Search for weekly maintenance scheduling

Christian Brunbjerg Jespersen[a], Kristoffer Sigsgaard Wernblad[a], Thomas Jacob Riis Stidsen[b], Kasper Barslund Hansen[a], Jingrui Ge[a], Simon Didriksen[a], Niels Henrik Mortensen[a]

[a] *DTU Construct, Technical University of Denmark, Anker Egelundsvej 1, Kongens Lyngby, 2800, Hovedstaden, Denmark*
[b] *DTU Management, Technical University of Denmark, Anker Egelundsvej 1, Kongens Lyngby, 2800, Hovedstaden, Denmark*

Abstract

Many planning problems facing the operations research field have proven difficult to solve due to their inherent uncertainty and highly dynamic nature. Stochastic Programming (Birge and Louveaux, 2011), fuzzy logic (Zadeh, 1988), and robust optimization (Ben-Tal et al., 2009) are some of the methods that have been proposed to solve these issues. These methods make an implicit assumption of a static data setting and a static problem setting. Maintenance scheduling is one such problem where the best available information continually updates and then therefore the scheduling continuously needs to be updated. Maintenance scheduling is a complex process often more associated with operation management, but here we will argue that is possible to implement general maintenance scheduling approaches, if the solution approach is designed to be integrated into a business process of the kind that are usually developed by the principles of operation management. This paper proposes a novel optimization method that is capable to optimizing a scheduling problem in the following setting: Primary data source is changing in real-time; external inputs affects the optimization process; multiple actors are making interdependent decision whose objectives may differ significantly. The proposed solution approach is an actor-based framework including a large neighborhood search metaheuristic implementation. The framework is tested on the real-world problem of maintaining scheduling of oil platforms for Total in the North Sea, but the approach is very general and can be applied to a wide variety of other planning problems.

Keywords: Large Neighborhood Search, Actor Framework, Maintenance scheduling, Real-time Optimization, Human-centered Computing, Interactive Systems and Tools, Decision Support Systems, Interactive Optimization.

1. Introduction

Maintenance scheduling is part of a class of operational problems that have proven hard to solve in practice (NP-hard (Garey and Johnson, 1979)). Furthermore, for optimization to be useful in dynamic and uncertain environments where maintenance scheduling is performed requires tight integration into existing IT infrastructure to allow the tacit knowledge of decision makers to influence the planning process easily. Often a number of different decision makers at different company levels take part in the planning process. In this way the industry usually assigns responsibility for decision-making to an individual representing only a small part of the complete process. These multiple smaller planning processes are often difficult to map to a single mathematical model describing the whole system as elaborated by (Barthélemy et al., 2002). Solving operation research problems that are operational in nature have additional requirements over more typical static problems: they have to be responsive to changing parameters; able to be assimilated into the decision-makers workflow; allow for integration with dynamic data sources such as databases and APIs (Meignan et al., 2015b). Operational aspects of operation research, as opposed to higher level strategic and tactical ones, are characterized by extensive amounts coordination and negotiation on proposed schedules often in a short amount of time.

The lack of integration into the schedulers and supervisors workflow and lack of responsiveness can lead to a situation where solutions are not directly implemented in practice but instead only provides initial suggestions (Meignan et al., 2015b). Theses initial suggestions are then iterated on elsewhere in the scheduling process usually through much more manual means. In (Barthélemy et al., 2002) the authors argue that many problems that operation research aim to solve are often composed of a group of individuals whose decisions are consolidated into an "epistemic subject" for which a mathematical model can be formulated and solved, with many scheduling problems being good examples. However often multiple actors have different views on what constitutes an optimal schedule hence resulting in multiple-objectives. Even if multi-objective optimization (Ehrgott and Gandibleux,

2002) is applied to find a Pareto Front (Pareto, 1897) a negotiating process still is needed between the actors to select the final schedule.

This paper proposes a solution method that will allow for real-time optimization based on actor/user interaction and connection to a dynamic data source, effectively managing the changes to the parameter space as they occur. The proposed solution method will be tested on the weekly maintenance scheduling problem Palmer (2019) which closely resembles a variant of the multi-compartment multi-knapsack problem (MCMKP). It should be noted that the scientific maintenance scheduling literature deviates significantly from its practical implementation which is also detailed in (Palmer, 2019). The solution method is based on the large neighborhood search (LNS) metaheuristic. LNS was chosen due to its properties of naturally being able to work with and fix infeasible solutions and its state of the art performance on various scheduling problems (Gendreau and Potvin, 2019).

To understand the need for actor-based methods some background knowledge will be required about the maintenance scheduling process. In figure **??** illustrates the general setup of a maintenance planning and scheduling system. The system's actors have the following responsibilities: the planner generates the work orders that are to be scheduled; the scheduler creates weekly schedules based on a knapsack formulation; based on the weekly schedule the supervisor assigns work order activities that the work order is composed of (the assignment problem); the technicians executes the work in sequential pattern (single machine scheduling).

A final point on the necessity of actor-based approaches to model such a setup is the idea of ownership of a work order. Throughout the scheduling process a work order is owned by a specific actor and he alone is allow to modify/execute it. This means that a single model approach is very difficult to implement in practice as a work order is modelled differently depending on the actor that currently owns it. This also highlight another an point in maintenance scheduling: that the stochastic nature of the maintenance scheduling process can be handled using a change of model each with different levels of aggregation and different sets of constraints, opposed to more academic approaches such as fuzzy logic and stochastic optimization. When the inherrent uncertainties manifest themselves during planning or execution, work orders are rescheduled by moving between the different actors, meaning that the stochastic elements of maintenance scheduling are handled by **dynamic rescheduling between the actors**.

This article has the following contribution:

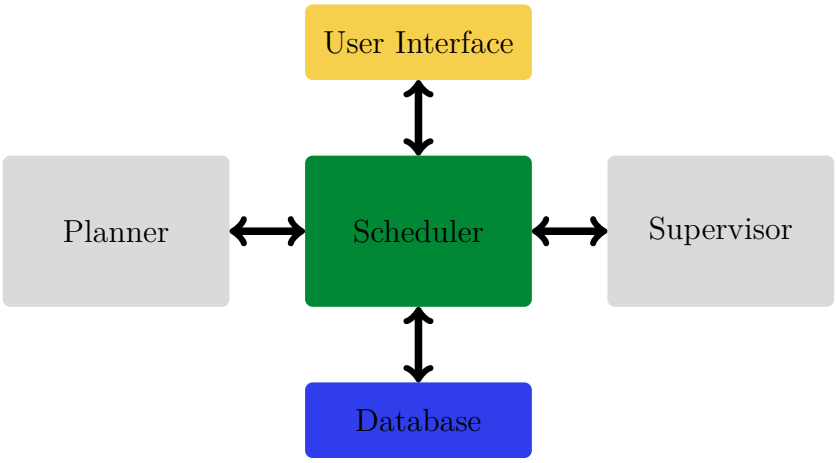


Figure 1: Simple overview of the scheduling process with its primary types of actors. The planner, the scheduler, the supervisor(s), and the technicians. The green color highlights the scheduler as it the actor in the maintenance scheduling process that is the foundation for the paper.

- To provide a modular scalable optimization component based on the large neighborhood search metaheuristic that will allow business processes to be composed from smaller units of implementing mathematical models, instead of larger integrated formulations.

The paper is divided into four different sections. Section 2 explains the weekly maintenance scheduling model in detail and forms the foundation of the paper. Section 3 shows that results coming from the implemented system where the implementation will be affected by simulated user-interaction. Section 4 will discuss the implications of the research and possible future research directions.

1.1. The Weekly Maintenance Scheduling Model

A company performing maintenance needs to create weekly maintenance plans for the next $p \in P(\tau)$ weeks. The weekly schedule is created centrally and consists of scheduling the $w \in W(\tau)$ work orders, i.e. maintenance tasks, such that all w are scheduled into a specific week p. Each work order w requires some resources $r \in R(\tau)$ to be carried out, e.g. man-power with different qualifications. Each of these resources are available in limited amounts $resource_{pr}(\tau, \beta(\tau))$. To correct for possible manual interventions that can make the problem infeasible a penalty *strategic_penalty* is introduced. The

urgency of the different maintenance operations varies and is reflected in a penalty for carrying out a maintenance work order in a certain week given by $strategic\_value_{wp}(\tau)$. The $strategic\_value_{wp}(\tau)$ also captures the tardiness of the individual work orders w, meaning that the value gained from scheduling work orders late is decreasing. Urgent tasks have increasingly less value the the larger that the week p becomes. Furthermore, two sets exists which will either require work order w to be carried out in week p or not carried out in week p These sets are $(w, p) \in include(\tau)$ for inclusion and $(w, p) \in exclude(\tau)$ for exclusion. The weekly maintenance scheduling model for the problem is a variant of the Multi-compartment Multi-knapsack Problem with capacity penalties MCMKP. The notation used to describe the dynamical aspects in the model is based on the notation from the dynamic metaheuristics literature as found in Yang et al. (2013). Here the τ is added as a time variable on all sets, parameters, and variables that are subject to change while the metaheuristic is running. This enables us to be precise in the timing on the messages that are send to the Ab-LNS and understand how it reacts in real-time

Meta variables:

$$s \in S \tag{1}$$

$$\beta(\tau) \tag{2}$$

$$\tau \in [0, \infty] \tag{3}$$

Maximize:

$$\sum_{w \in W(\tau)} \sum_{p \in P(\tau)} strategic\_value_{wp}(\tau) \cdot \alpha_{wp}(\tau)$$
$$- \sum_{p \in P(\tau)} \sum_{r \in R(\tau)} strategic\_penalty \cdot \epsilon_{pr}(\tau)$$
$$+ \sum_{p \in P(\tau)} \sum_{w1 \in W(\tau)} \sum_{w2 \in W(\tau)} clustering\_value_{w1,w2} \cdot \alpha_{w1p}(\tau) \cdot \alpha_{w2p}(\tau) \tag{4}$$

Subject to:

$$\sum_{w \in W(\tau)} work\_order\_work_{wr} \cdot \alpha_{wp}(\tau) \leq \ resource_{pr}(\tau, \beta(\tau)) + \epsilon_{pr}(\tau) \quad \forall p \in P(\tau) \quad \forall r \in R(\tau) \tag{5}$$

$$\sum_{w \in W(\tau)} \alpha_{wp}(\tau) = 1 \quad \forall p \in P(\tau) \tag{6}$$

$$\alpha_{wp}(\tau) = 0 \quad \forall (w, p) \in exclude(\tau) \tag{7}$$

$$\alpha_{wp}(\tau) = 1 \quad \forall (w, p) \in include(\tau) \tag{8}$$

$$\alpha_{wp}(\tau) \in \{0, 1\} \quad \forall w \in W(\tau) \quad \forall p \in P(\tau) \tag{9}$$

$$\epsilon_{pr}(\tau) \in \mathbb{R}^{+} \quad \forall p \in P(\tau) \quad \forall r \in R(\tau) \tag{10}$$

The meta variables defines the broader setting that the model in implemented in. (1) specicies that the model is implemented for scheduler s. (2) is the value of a decision variable coming from a different model, here the supervisor. (3) is the time variable that binds the whole system together.

The objective function (4), maximizes the total weighted delay of all work order assignments together with the penalty *strategic_penalty* for exceeding the resource capacity given in constraint (5). The third term of the model contains the $clustering\_value_{w1,w2}$ which turns the model into a quadratic problem. This term optimizes the value of putting two work orders in the same period, if they have share similarity like close proximity, same functional location, etc. Constraint (5) ensures that all the weights $work_{wr}(\tau)$ for each activity in an work order, given that it has been assigned, is lower than the capacity for each period and for each resources r. *strategic_penalty* is the amount of exceeded capacity that is needed for the current assignment of work order to be feasible. Constraint (6) makes sure that each work order is assigned to at least a single period. Constraint (8) excludes a work order from a certain period and constraint (7) forces a specific work order to be in a specific period. Constraint (9) and (10) specify the variable domain for $\alpha_{wp}(\tau)$ and *strategic_penalty* respectively. The effects of changing $include(\tau)$, $exclude(\tau)$, $resource_{pr}(\tau, \beta(\tau))$, and $strategic\_value_{wp}(\tau)$ in real-time will be examined to determine their effects on the weekly schedules and objective value.

2. Solution Method

2.1. Actor-based Large Neighborhood Search

A problem formulation that is affected by the end user and requires real-time feedback inherently requires an optimization approach that is able to repair infeasible solution and while also converging quickly. For this the large neighborhood search (LNS) metaheuristic has been shown satisfy these requirements in the literature Gendreau and Potvin (2019). The most general form of the LNS metaheuristic is defined for static problems, meaning that the parameters that make up the problem instance is not subject to change after the algorithm has been started. To make the LNS able adapt to changing parameters in real-time a message system have been implemented into the existing framework. This extension is shown in algorithm 1.

2.1.1. Messages And Destructors

LNS in its basic form has one constructor and one destructor which repeatedly destroy and rebuild the solution. For the AbLNS we will generalize on this concept by including messages as destructors of the classic LNS implementation. This generalization can be seen as being somewhat similar to how the adaptive LNS (ALNS) is formulated, but where the different constructors and destructors are also chosen from sources external to the algorithm.

Extending on the classic setup we define the following set of destructors, $m \in M$.

- m_1: Inclusion destruct message
- m_2: Exclusion destruct message
- m_3: Capacities destruct message
- m_4: Weights destruct message
- m_5: Random destruct message

Each of these messages affect different parts of the weekly maintenance scheduling problem. Notice here that the first four messages destruct the solution by changing the parameter space and the last message is a random destructor.

For correct implementation it is critical that the that destruct messages are handled one by one, and are not allowed to simply write to the solution and problem instance whenever they appear. Generalizing the destructors from being static structures into messages allows the solution to change in real-time to a changing parameter space. This means that the algorithm does not need to restart to handle changes in data.

Algorithm 1 Actor-based Large Neighborhood Search

```
1: Input Q = message queue
2: Input P = problem instance
3: Input X = initial schedule
4: Input S = SharedSolution
5: repeat
6:     X^t = clone(X)
7:     while Q.has_message() do
8:         P.update(S, m)
9:         X^t.destruct(S, m)
10:    end while
11:    X^t.repair(S)
12:    if accept(X^t, X) then
13:        X.update(X^t)
14:    end if
15:    if c(X^t) < c(X) then
16:        X.update(X^t)
17:        S.atomic_pointer_swap(X)
18:    end if
19:    Q.push(m)
20: until
```

The basic LNS setup have here been extended with a 'message queue' Q. The message queue will be read from on every iteration of the LNS's main iteration loop. Here we notice that the incoming message is able to change both the solution X but also the problem instance P itself. Here we see the defining feature of the LNS metaheuristic, that due to its inherent property of being able to optimize a solution that have become infeasible. This is something that is going happen when you change the parameter of the problem instance P itself. Another less obvious property of the message queue is that the algorithm can run indefinitely and opposed to restarting the algorithm you instead pass messages to it to allow it be adjust both the solution space and the parameter space. This property avoid the issue of time consuming initial convergence as the algorithm will be found in an optimal state when the solution is perturbed. Notice that:

- The algorithm responds to changes very quickly, in each iteration line 7. We call this a fine-grained response algorithm

- If an improved solution is found, it is immediately being pushed to the data (base) on line 15
- That the optimization occurs in the repair function on line 11, which inserts operations in a greedy fashion.

3. Results

The results section will: 1. introduce the data instance from the case company; 2. show the effect of forcing work orders into specific weekly schedules $p \in P(\tau)$; 3. show the effect of changing the period capacities $resource_{pr}(\tau, \beta(\tau))$, and 4. show the effect of dynamically changing the $strategic\_value_{wp}(\tau)$ associated with $p \in P(\tau)$ and $w \in W(\tau)$.

3.1. Data Instance

	$\lvert W(\tau)\rvert$	$\lvert R(\tau)\rvert$	$\lvert P(\tau)\rvert$
Instance 1	3487	16	52

Table 1: List of data instances from the case company. Here $W(\tau)$ is the set of work orders, $R(\tau)$ is the set of resources, and $P(\tau)$ is the set of weekly periods.

3.2. Response to Inclusion

The response to the inclusion of a work order is given by I parameter of the model which is constrained in 7 of model given in .

The inclusion is made of forcing certain allocations of work orders to be in specific periods. Below a table is provided to show what changes will occur and at what and at what point in time.

	$\tau_1 = 60$	$\tau_2 = 120$	$\tau_3 = 180$	$\tau_4 = 240$	$\tau_5 = 300$
$\Delta\lvert P(\tau)\rvert$	10	20	30	40	50

With the inputs defined we will explain the main results which are shown in the figure below.

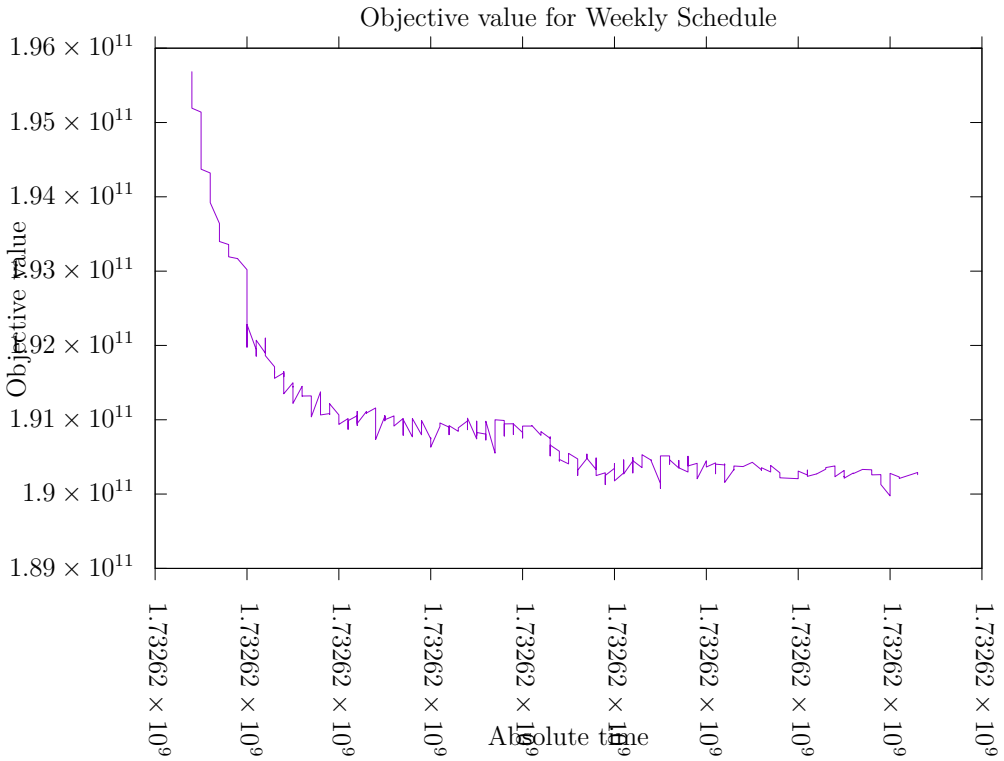


Figure 2: Figure Caption

3.3. Response to Exclusion

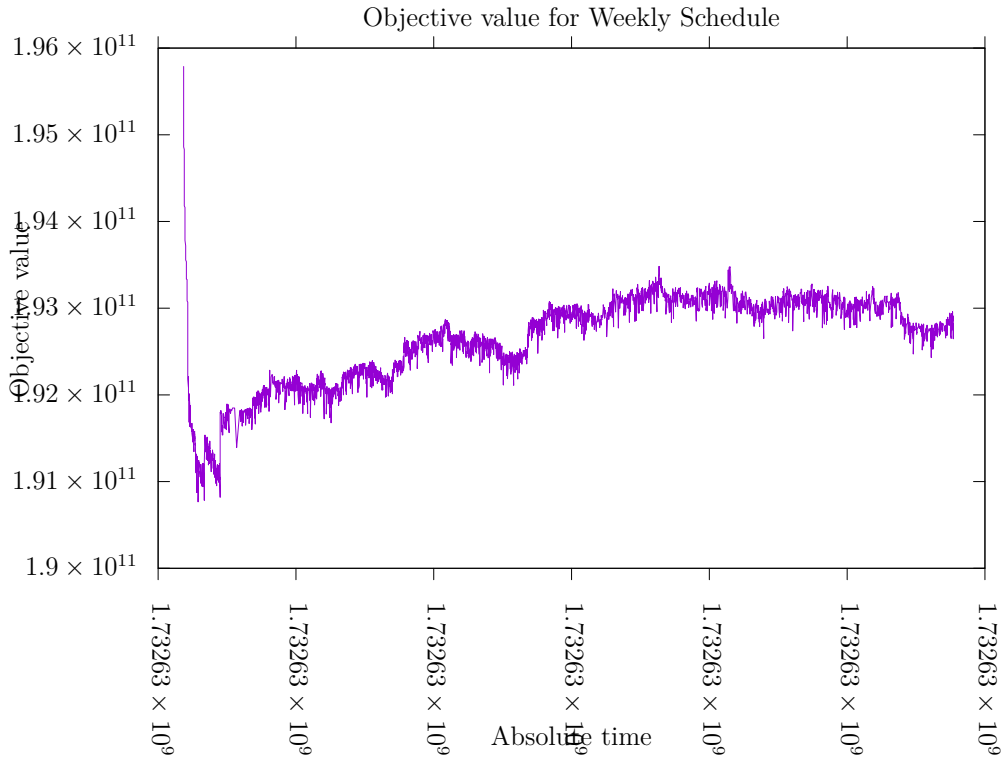


Figure 3: Figure Caption

3.4. Response to Changes in Knapsack Capacities

The effects of changes to capacities will be illustrated in the same way as it was with the response to inclusion and below we see the table that shows which inputs that the AbLNS will be affected by.

	$\tau_1 = 60$	$\tau_2 = 120$	$\tau_3 = 180$	$\tau_4 = 240$	$\tau_5 = 300$
$\Delta\|P(\tau)\|$	16	16	16	16	16
$\Delta\|R(\tau)\|$	16	16	16	16	16
$\Delta\|resource_{pr}(\tau, \beta(\tau))\|$	100	200	400	800	1600

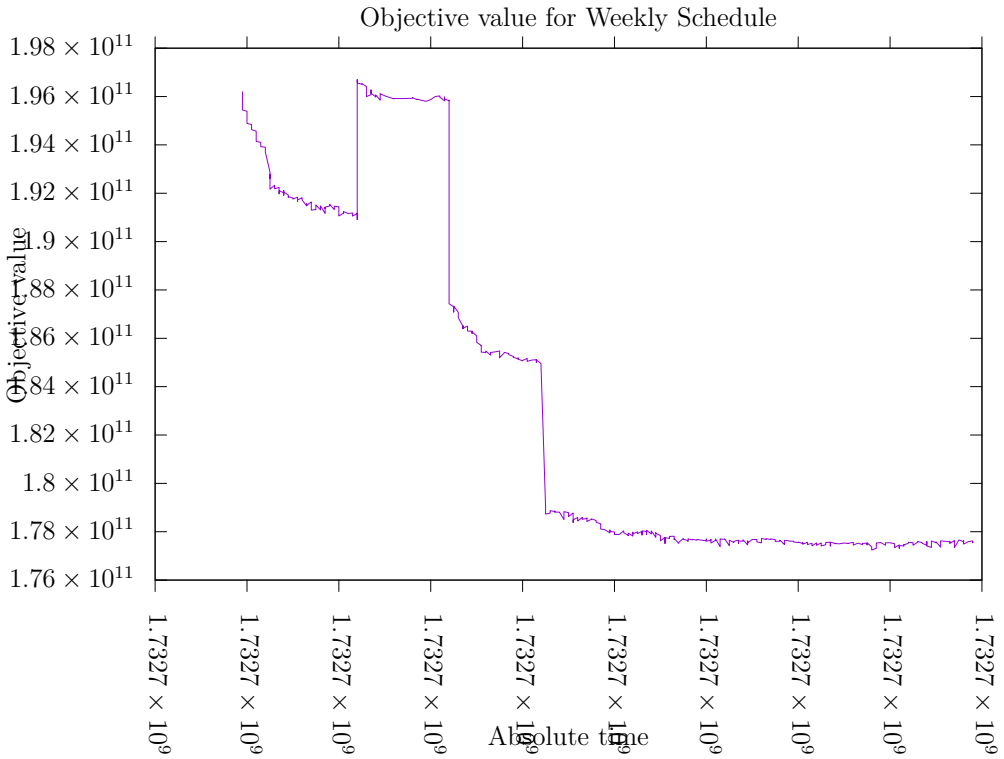


Figure 4: Figure Caption

Correspondingly we also have the figure below in which the resources are decreasing.

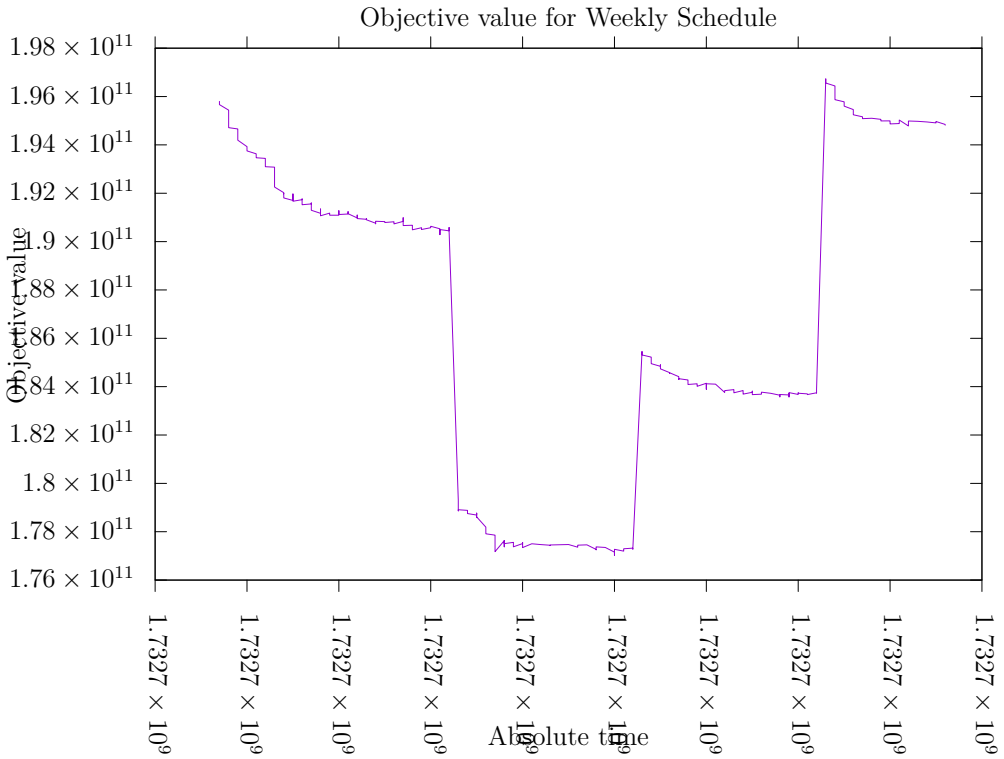


Figure 5: Figure Caption

3.5. Response to Changes in Item Weights

The final parameter that will be changed is the work order value $strategic\_value_{wp}(\tau)$. This section will be more elaborate as we have to show how that the work orders are rearranged due to the changes in their value across the different periods.

	$\tau_1 = 60$	$\tau_2 = 120$	$\tau_3 = 180$	$\tau_4 = 240$	$\tau_5 = 300$
$\Delta\|W(\tau_n) \triangle W(\tau_{n-1})\|$	20	40	80	160	320
$\Delta\|P(\tau_n) \triangle P(\tau_{n-1})\|$	26	26	26	26	26
$\Delta\|strategic\_value_{wp}(\tau_n)\| - \|strategic\_value_{wp}(\tau_{n-1})\|$	$1 \cdot 10^5$	$2 \cdot 10^5$	$4 \cdot 10^5$	$8 \cdot 10^5$	$1.6 \cdot 10^6$

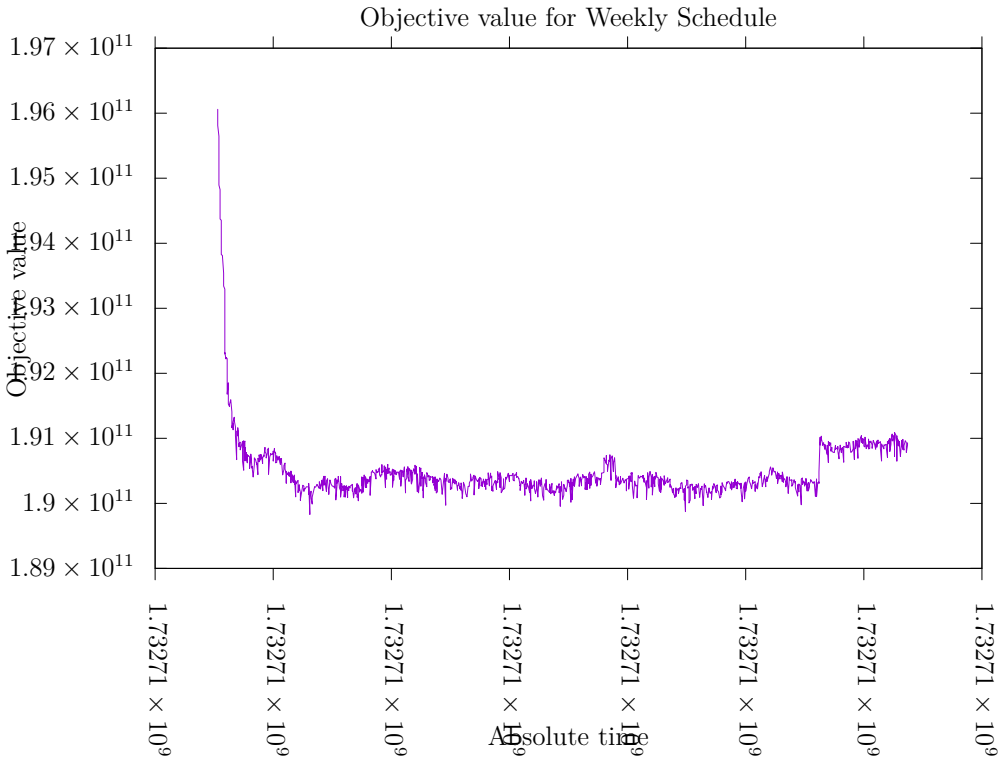


Figure 6: Figure Caption

4. Discussion

Maintenance scheduling efficiently solves a complex scheduling problem by the use of multiple actors. Through the use of actors the process handles uncertainty that is difficult to reason about, using models with different levels of aggregation where each actors understands how to exploit his model. As uncertainties manifest themselves the actors/models handle the uncertainty through communication. To further understand the implications of the approach the discussion will be divided into three sections: 1. actors and integration; 2. continuous optimization allows asynchronous optimization; 3. future research.

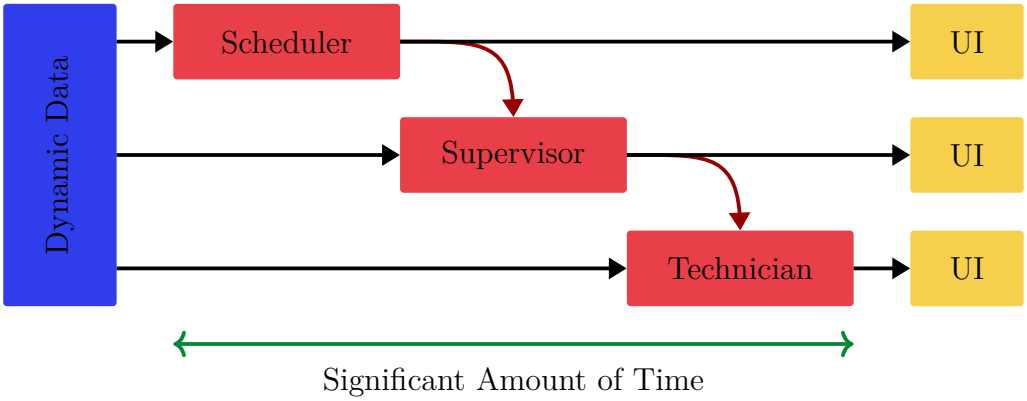
4.1. Actors & Integration

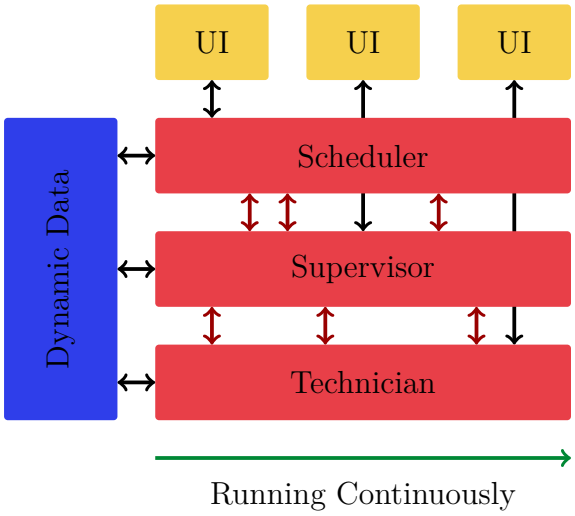
Often in operation research the failure to reliably solve operational problems in industry are not due to the problems being computationally in-

tractable Gendreau and Potvin (2019) but a practical problem of connecting data streams so that the solution approach continually receives dynamic data to handle changes and then afterwards connecting the resulting solutions to the relevant stakeholders (actors) through a relevant interface Meignan et al. (2015a). The actor-based approach proposed in this paper makes integration easier by naturally encapsulating a model with a reliable interface.

4.2. Continuous Optimization

With actor-based metaheuristics the optimization loop can run indefinitely optimizing based on the latest available information. This may seem like a detail as you could argue that you should only ever optimize the schedule when there is an explicit need for it, but consider the case when you start adding more than two actor to a scheduling system, then there arises a need to coordinate people in time as each will have to run their optimizer on after another.





4.3. Future Research

The primary future research of this project is to demonstrate that the system described here can be used to model and optimize larger scale scheduling processes. A possible extension here is shown in 7 where each of the actors run a metaheuristic and that each metaheuristic will shared its solutions with the other metaheuristics through atomic pointer swapping. Communicate with the end-user through message passing, integrate with a persistence storage through mutex locks, and the lifecycle of each of the metaheuristics will be controlled by the orchestrator through message passing.

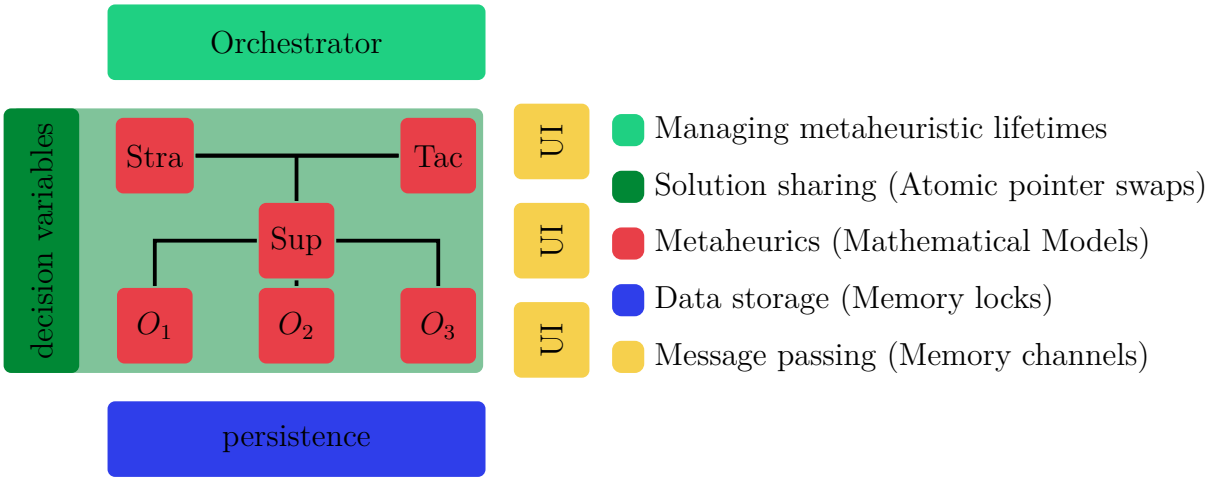


Figure 7: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

References

Barthélemy, J.P., Bisdorff, R., Coppin, G., 2002. Human centered processes and decision support systems. European Journal of Operational Research 136, 233–252.

Ben-Tal, A., Nemirovski, A., El Ghaoui, L., 2009. Robust optimization .

Birge, J.R., Louveaux, F., 2011. Introduction to stochastic programming. Springer Science & Business Media.

Ehrgott, M., Gandibleux, X., 2002. Multiple criteria optimization: state of the art annotated bibliographic surveys .

Garey, M.R., Johnson, D.S., 1979. Computers and intractability. volume 174. freeman San Francisco.

Gendreau, M., Potvin, J.Y. (Eds.), 2019. Handbook of Metaheuristics. volume 272 of *International Series in Operations Research & Management Science*. Springer International Publishing, Cham. doi:10.1007/978-3-319-91086-4.

Meignan, D., Knust, S., Frayret, J.M., Pesant, G., Gaud, N., 2015a. A Review and Taxonomy of Interactive Optimization Methods in Operations Research. ACM Transactions on Interactive Intelligent Systems 5, 1–43. doi:10.1145/2808234.

Meignan, D., Knust, S., Frayret, J.M., Pesant, G., Gaud, N., 2015b. A Review and Taxonomy of Interactive Optimization Methods in Operations Research. ACM Transactions on Interactive Intelligent Systems 5, 1–43. URL: https://dl.acm.org/doi/10.1145/2808234, doi:10.1145/2808234.

Palmer, R.D., 2019. Maintenance Planning and Scheduling Handbook, 4th Edition . 4th edition ed., McGraw Hill.

Pareto, V., 1897. The new theories of economics. The Journal of Political Economy 5, 485–502.

Yang, S., Jiang, Y., Nguyen, T.T., 2013. Metaheuristics for dynamic combinatorial optimization problems. IMA Journal of Management Mathematics 24, 451–480. doi:10.1093/imaman/dps021.

Zadeh, L.A., 1988. Fuzzy logic. Computer 21, 83–93.