# 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 an operational problem that have proven hard to solve (NP-hard (Garey and Johnson, 1979)). Furthermore, for optimization to be utilized the dynamic and uncertain nature of maintenance scheduling requires a tight integration with third party administration software to enable 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 and 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 conventional 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 RESTapi (Meignan et al., 2015). Operational aspects of operation research, as opposed to higher level strategic and tactical aspects, are characterized by extensive amounts negotiation and feedback on proposed schedules. The lack of integration and responsiveness can lead to schedules that are not directly implemented in practice but instead provides initial suggestions (Meignan et al., 2015), which are then iterated else where in the scheduling process. 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 the 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. The proposed solution method will be tested on the multi-compartment multi-knapsack problem (MCMKP) for maintenance scheduling on a large dataset from a company. The MCMKP naturally models what in the practical maintenance is called the weekly schedule, taken form (Palmer, 2019). It should be noted that the scientific maintenance scheduling literature deviates significantly from its practical implementation which is detailed in (Palmer, 2019). The solution method will by based on the large neighborhood search (LNS) metaheuristic. This meta heuristic 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.

To understand the need for actor-based methods some background knowledge will be required about the maintenance scheduling process. In figure 1 illustrates the general setup of a healthy maintenance planning and scheduling system. The systems 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 should 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 it. This means that a single model approach is very difficult to implement in practice as a work order looks different 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, opposed to more academic approaches such as fuzzy logic and stochastic optimization.

When the fundamental uncertainties manifest themselves during planning or execution work orders are rescheduled by moving between the different actor (models), meaning that the stochastic elements of maintenance scheduling are handled by dynamic rescheduling between the actors.

This article describes a number of contributions:

- A novel actor/optimization framework

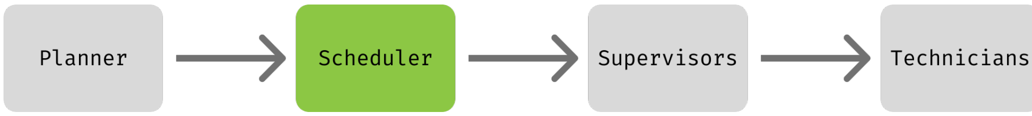


Figure 1: Simple overview of the scheduling process with its primary types of actors. The planner, the scheduler, the supervisor(s), and the technicians. The green color highlights the scheduler as it the actor in the maintenance scheduling process that is the foundation for the paper.

- A novel specialized (A)LNS metaheuristic, to be utilized in an actor framework
- Implementation and test on a large realworld maintenance scheduling problem

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

### *1.1. A generic maintenance scheduling model*

A large company needs to create a weekly maintenance plan for the next $p \in P$ weeks. The maintenance plan is planned centrally and consists of scheduling the $w \in W$ work orders, i.e. maintenance tasks, such that all are scheduled into different weeks. Each work order requires some resourses to be carried out, e.g. man-power with different qualifications, equipment etc. all of these resourses are available in limited amounts and are called traits $\tau \in T$. To simplify matters, we will assume that the recourse limits are not hard but extra workers can be paid overtime, extra equipment can be rented etc., at the cost (penalty) of $pen_{p\tau}$. 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 $v_{wp}(t)$. Urgent tasks have quickly increasing penalties for the later weeks week $p$. Furthermore, two sets exists which will either will require the work order to be carried out in week $p$, i.e. $(w, p) \in E$ and a set which forbids a work order to carried out in week $p$, i.e. $(w, p) \in I$. The model for the problem is the Multi-compartment Multi-knapsack Problem with capacity penalties MCMKP.

The notation used in the model is based on the notation from the dynamic metaheuristics literature as found in Yang et al. (2013), where $t$ is added as a time variable on all sets, parameters, and variables that are subject to change. This allows us to be precise in the timing on the messages that are send to the Ab-LNS.

## 2. The Strategic Model

The Strategic Model have multiple different purposes.

- Schedule Work Order out across the weekly periods
- Prioritize all the different released work orders
- Respect the available weekly hours available for each trait

The Strategic model is responsible for grouping work orders into weekly or biweekly periods depending on which kind of maintenance setup that one is running. This kind of model closely resembles a variant of the multi-compartment multi-knapsack problem.

**Meta variables:**

$s \in S$

$\tau \in [0, \infty]$

**Minimize:**

$$\sum_{w \in W(\tau)} \sum_{p \in P(\tau)} strategic\_value_{wp}(\tau) \cdot \alpha_{wp}(\tau) \tag{1}$$

$$+ \sum_{p \in P(\tau)} \sum_{r \in R(\tau)} strategic\_penalty \cdot \epsilon_{pr}(\tau) \tag{2}$$

$$+ \sum_{p \in P(\tau)} \sum_{w1 \in W(\tau)} \sum_{w2 \in W(\tau)}$$

$$clustering\_value_{w1,w2} \cdot \alpha_{w1p}(\tau) \cdot \alpha_{w2p}(\tau) \tag{3}$$

**Subject to:**

$$\sum_{w \in W(\tau)} work\_order\_work_{wr} \cdot \alpha_{wp}(\tau)$$

$$\leq \ resource_{pr}(\tau) + \epsilon_{pr}(\tau)$$

$$\forall p \in P(\tau) \quad \forall r \in R(\tau) \tag{4}$$

$$\sum_{w \in W(\tau)} \alpha_{wp}(\tau) = 1$$

$$\forall p \in P(\tau) \tag{5}$$

$$\alpha_{wp}(\tau) = 0$$

$$\forall (w, p) \in exclude(\tau) \tag{6}$$

$$\alpha_{wp}(\tau) = 1$$

$$\forall (w, p) \in include(\tau) \tag{7}$$

$$\alpha_{wp}(\tau) \in \{0, 1\}$$

$$\forall w \in W(\tau) \quad \forall p \in P(\tau) \tag{8}$$

$$\epsilon_{pr}(\tau) \in \mathbb{R}^+$$

$$\forall p \in P(\tau) \quad \forall r \in R(\tau) \tag{9}$$

$$\tau \in [0, \infty] \tag{10}$$

The objective function (**??**), (**??**), and (**??**) minimizes the total weighted delay of all work order assignments together with the penalty *strategic_penalty* for exceeding the resource capacity given in constraint (4). The thrid 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 (4) 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 trait $\tau$. $pen_{p\tau}$ is the amount of exceeded capacity that is needed for the current assignment of work order to be feasible. Constraint (5) makes sure that each work order is assigned to at least a single period. Constraint (6) excludes a work order from a certain period and constraint (7) forces a specific work order to be in a specific period. Constraint (8) and (9) specify the variable domain for $x_{wp}$ and $pen_{p\tau}$ respectively. The effects of changing $E$, $I$, $cap$, and $v$ in real-time will be examined to determine their effects on the weekly schedules and objective value.

The objective function (3) minimizes the total weight of all work order assignments together with the penalty $d$ for exceeding the capacity given in constraint (4). Constraint (4) ensures that all the weights $c_{w\tau}$ for each activity in an work order, given that it has been assigned, is lower than the capacity for each period and for each trait $\tau$. $pen_{p\tau}$ is the amount of exceeded capacity that is needed for the current assignment of work order to be feasible. Constraint (5) makes sure that each work order is assigned to at least a single period. Constraint (6) excludes a work order from a certain period and constraint (7) forces a specific work order to be in a specific period. Constraint (8) and (9) specify the variable domain for $x_{wp}$ and $pen_{p\tau}$ respectively. The effects of changing $E$, $I$, $cap$, and $v$ in real-time will be examined to determine their effects on the weekly schedules and objective value.

## 3. Solution Method

### *3.1. Actor-based Large Neighborhood Search*

A problem which is affected by user-interaction and requires real-time feedback needs an optimization approach that is able to repair infeasible schedules and while also converging quickly. For this the large neighborhood search metaheuristic has been shown satisfy these requirements in the literature Gendreau and Potvin (2019).

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.

### *3.1.1. Messages And Destructors*

LNS in its most basic form has one constructor and one destructor which repeatedly destroy and rebuild the schedule. 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 chosen externally as well.

Extending on the classic setup we define the following set of destructors, $M$:

- $m_1$: 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 MCMK problem (weekly schedule). Notice here that the first four messages destruct the solution by changing the parameter space and the last message is a random destructor.

Generalizing the destructors from being static structures into messages allows the solution to change in real-time to a changing paramenter space meaning that the algorithm does not need to restart to handle changes in data.

**Algorithm 1** Actor-based Large Neighborhood Search

1: **Input** queue = message queue
2: **Input** P = problem instance
3: **Input** x = initial schedule
4: **while** true **do**
5:     **while** queue.has_message() **do**
6:         $P.update(m)$
7:         $x.destruct(m)$
8:     **end while**
9:     $x^t = x.repair()$
10:     **if** $c(x^t) < c(x)$ **then**
11:         $x = x^t$
12:         queue.send($x$)
13:     **end if**
14:     queue.push($m_5$)
15: **end while**
16: **return** $x^b$

The basic LNS setup have here been extended with a ‘message queue‘. This 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 but also the problem instance itself. Here we see one of the defining features of the LNS metaheuristic in play, that due to its inherent property of being able to optimize a solution that have become infeasible which is something that is very likely to happen when you change the parameter of the problem instance itself.

Another less obvious property the message queue allows is for the algorithm to run indefinitely and instead of 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 optimimal state when the solution is perturbed.

Notice that:

- The algorithm responds to changes very quickly, in each iteration (line 5-8. We call this a fine-grained response algorithm
- If an improved solution is found, it is immediately being pushed to the data (base). (line 12)

- That the optimization occurs in the repair function (line 9), which inserts operations, not scheduled yet in a greedy fashion. While not being an optimal insertion, it is fast

## 4. Results

The results section will: 1. introduce the real-world data instance; 2. show the effect of forcing item set in the specific weekly schedules; 3. show the effect of changing the period capacities, and 4. show the effect of dynamically changing the value of the work orders $v$.

### *4.1. Data Instance*

| | Number of Item Sets | Number of Compartments | Number of Knapsacks |
|---|---|---|---|
| Instance 1 | 3487 | 16 | 52 |

Table 1: Table Caption

### *4.2. 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.

| | At Time: 01:00 | At Time: 02:00 | At Time: 03:00 | At Time: 04:00 | At Time: 05:00 |
|---|---|---|---|---|---|
| $\Delta\lvert P\rvert$ | 10 | 20 | 30 | 40 | 50 |

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

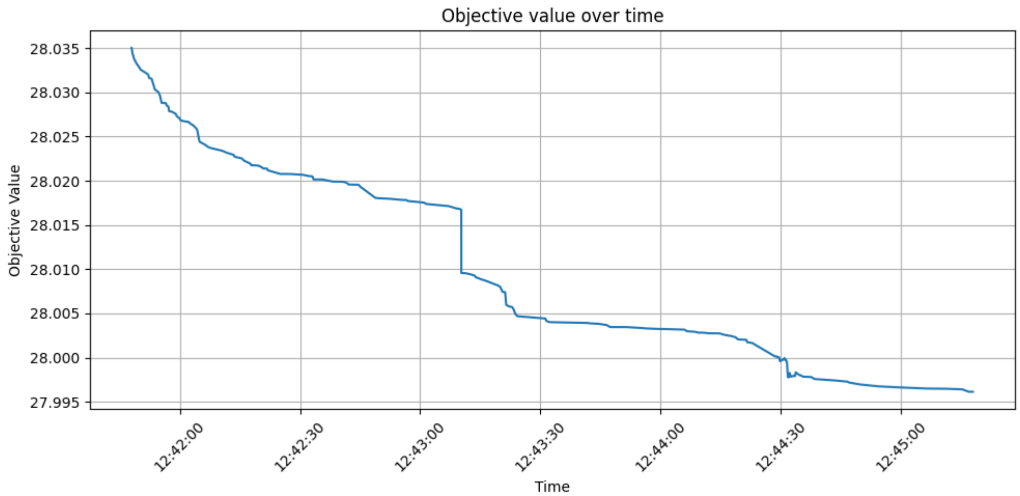


Figure 2: Figure Caption

*4.3. Response to Exclusion*

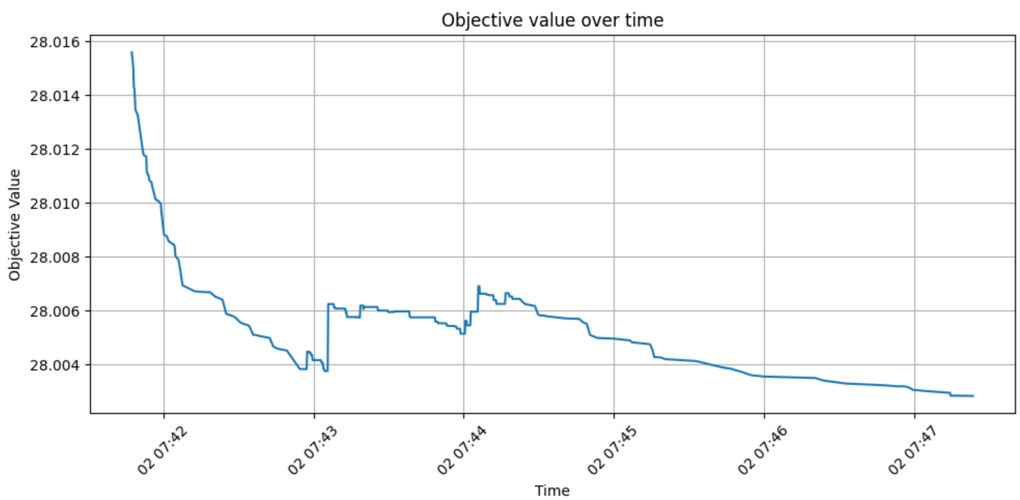


Figure 3: Figure Caption

*4.4. 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.

| | At Time: 01:00 | At Time: 02:00 | At Time: 03:00 | At Time: 04:00 | At Time: 05:00 |
|---|---|---|---|---|---|
| $\Delta\|p\|$ | 16 | 16 | 16 | 16 | 16 |
| $\Delta\|\tau\|$ | 16 | 16 | 16 | 16 | 16 |
| $\Delta\|cap\|$ | 100 | 200 | 400 | 800 | 1600 |

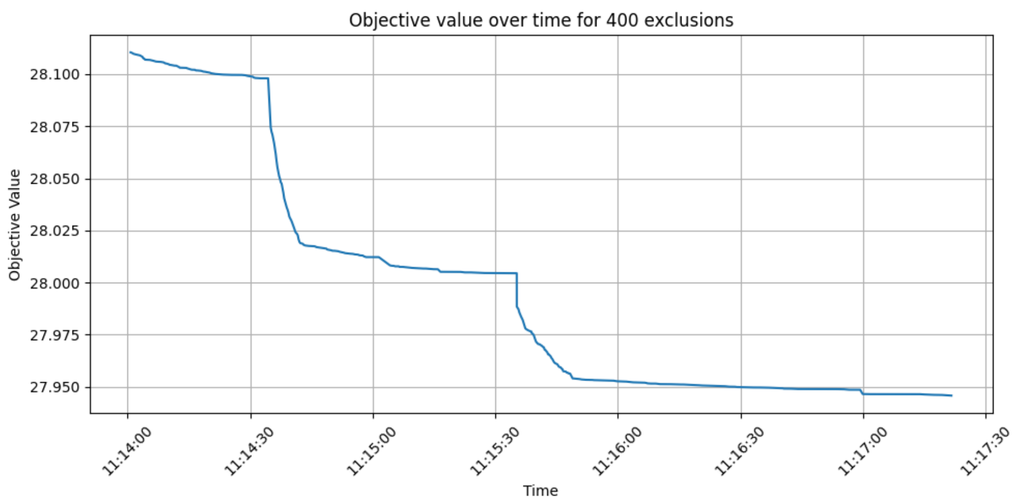


Figure 4: Figure Caption

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

### *4.5. Response to Changes in Item Weights*

The final parameter that will be changed is the work order value $v$. 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.

| | At Time: 01:00 | At Time: 02:00 | At Time: 03:00 | At Time: 04:00 | At Time: 05:00 |
|---|---|---|---|---|---|
| $\Delta\|w\|$ | 20 | 40 | 80 | 160 | 320 |
| $\Delta\|p\|$ | 26 | 26 | 26 | 26 | 26 |
| $\Delta\|v\|$ | $1 \cdot 10^5$ | $2 \cdot 10^5$ | $4 \cdot 10^5$ | $8 \cdot 10^5$ | $1.6 \cdot 10^6$ |

## 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., 2015. 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.