Reactive Maintenance Scheduling*

Christian Brunbjerg Jespersen[1 [0000−1111−2222−3333]]

Technical University of Denmark, Kongens Lyngby 2800, Denmark cbrje@dtu.dk

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: Hierarchical Models · Reactive Scheduling · Actor based Large Neighborhood Search

1 Reactive Maintenance Scheduling

Maintenance scheduling is by its nature an uncertain process, breakdowns are difficult to predict and due to the often critical nature of corrective maintenance resources (technicians) often have to be ready to repair the underlying equipment. Traditionally maintenance is scheduled as a resource constrained project scheduling problem (RCPSP), that inherently has some level of aggregation build in to handle the uncertainty.

In the literature there have been developed different theories and principles to make the RCPSP able to handle increasing levels of uncertainty. These include but are not limited to:

- Stochastic optimization
- Fuzzy logic
- Robust optimization

There are multiple hidden assumptions behind the use of these approaches and this paper will argue that to handle the uncertainty present in maintenance scheduling a different approach is needed. First, **stochastic optimization** is often impractical to apply due to the singular nature of many maintenance tasks, meaning that acquiring the amount of data needed to perform the scenario generation is challenging. Second, **fuzzy logic** is actually a good approach and is a concept that could be applicable to maintenance optimization but is not considered in this paper. This is primarily due to maintenance being scheduled so technicians and team leaders get a pool of jobs to select from, so that they can adjust the schedule daily during execution, this real-time shuffling makes something like fuzzy logic ineffective as the days that are assigned for each project can move around continuously. To overcome this issue you would have to enforce the statically generated schedule which will be daunting challenging in practice. Third, **robust optimization** robust can be seen as a more practical implementation of the stochastic optimization, where instead of optimizing the schedule

* Supported by organization x.

over a set of scenarios you instead include buffers on the jobs and activities. Robust optimization is usually applied in practice where it contributes to one of the main inefficiencies found in maintenance scheduling. Maintenance scheduling is usually focused on availability of the underlying equipment, so a common approach is to schedule in buffers on the resources (technicians) so that you are always ready to respond to a breakdown on critical equipment.

This paper proposes a different approach based on reactive and interactive models. To achieve these properties, compromises and novelties have to be made and introduced for the scheduling system as a whole to have the desired characteristics. In this paper these compromises and novelties are grouped into the following three subsections: hierarchical model setup, continuous optimization, and user interaction.

1.1 Hierarchical Model Setup

In standard maintenance scheduling the process is usually divided up into three distinct processes:

- **A long term strategic schedule**
- **A weekly tactical schedule**
- A daily operational schedule

Each of these schedules represents a process with distinct objectives and constraints, and this naturally leads to a each schedule having its own model. As uncertainty aggregates as time passes we get into a situation where there is a considerable more uncertainty on the long term strategic perspective than in the two shorter term models. This means that if you were to try and model the long term strategic model using the weekly tactical model, it would be hard to assign meaning to the solution of the model as the variables and constraints model data which we have little ability to reason about. This paper will look at the interplay between the strategic schedule and the tactical schedule leaving the operational schedule for future work.

The Strategic Model The strategic schedule has three main goals.

- prioritization of jobs
- levelization of resources
- creation of a stable and unique schedule

To achieve these goals within the additional requirements of reactivity and interactivity while also being scalable for problem instances containing up to 10000 jobs (20000 activities). The maintenance scheduling problem is formulated as a multi-compartment multi-knapsack knapsack problem (MCMKKP). This model has been carefully chosen to give some desirable model characteristics which the RCPSP lacks either directly or indirectly. First, getting a stable and unique schedule with 20000 activities in a RCPSP is extremely challenging as the

problem is NP-complete with the RCPSP you will have a variable for each **day** and for each **activity** with corresponding constraints, where for the MCMKKP we only have to variable for each **peried** and **job**. This of course simplifies the problem considerably but it also provides a new characteristic that is crucial for a long term maintenance schedule. This is the property of allowing for gradual reduction in the resource capacity through out the periods. Modifing the RCPSP to get this property is not trivial as it requires that jobs and activities should be handled differently based of the period they are scheduled in. Besides being difficult it also alters the interpretation of the problem in an undesirable way by assigning scheduling a job and its activities differently depending on when in the schedule it is placed. Given this justification we will move on to explaining the model itself.

The first model is the strategic model. The model is comprised of five different sets. Here: P is the number of periods; O is the number of work orders; R is the number of resources, Q is a set which defines the work orders which should be excluded from a specific period; P is an inclu- sion set that defines the work orders which should be included in each period. The model introduces four different parameters. Here: $v_{po} is the value associated with work order o in period p$;penalty$paid for exceeding the resource capac$
$variables, here : x_{po} is a binary decision-variable equal to one if work order o is in period p and zero otherwise; p_{pr} is$
$negative decision variable equal to the amount of excess capacity above the c_{pr} in period p for resource r.$

$$\text{Min} \quad \sum_{p=1}^{P}\sum_{o=1}^{O} cost_{po} \cdot x_{po} - \sum_{p=1}^{P}\sum_{r=1}^{R} penalty \cdot excess_{pr} \tag{1}$$

subject to:

$$\sum_{o=1}^{m} w_{or} \cdot x_{po} \leq \; c_{pr} + p_{pr} \qquad \forall p \in \{1..P\}, \forall r \in \{1..R\} \tag{2}$$

$$\sum_{p=1}^{P} x_{po} \leq \; 1 \qquad \forall o \in \{1..O\} \tag{3}$$

$$x_{po} = 0 \qquad \forall (p,o) \in Q \tag{4}$$

$$x_{po} = 1 \qquad \forall (p,o) \in P \tag{5}$$

$$x_{po} \in \{0,1\} \qquad \forall p \in \{1..P\}, \; \forall o \in \{1..O\} \tag{6}$$

$$excess_{pr} \in \mathbb{R}^{+} \qquad \forall p \in \{1..P\}, \; \forall r \in \{1..R\} \tag{7}$$

The Tactical Model

$$\min \sum_{o \in O} \omega_o \cdot T_o + \sum_{wc \in WC, d \in D} \lambda \cdot s_{wc,d} \tag{8}$$

subject to:

$$\sum_{d \in D} x_{k,d} = Q_k \sum_{d \in D} p_{k,d} \qquad \forall k \in K \tag{9}$$

$$x_{k,d} \leq q_k \sum_{\pi} \Pi_{k,\pi,d} \cdot p_{k\pi} \quad \forall d \in D, \forall k \in K \tag{10}$$

$$\sum_{\pi} p_{k\tau} = 1 \qquad \forall k \in K \tag{11}$$

$$p_{k,d} = 0 \qquad \forall k \in K, \forall d < \alpha_k \tag{12}$$

$$\sum_{d \in D} d \cdot p_{k_1,d} = \sum_{d \in D} d \cdot p_{k_2,d} \qquad \forall (k_1, k_2) \in \eta \tag{13}$$

$$\sum_{d \in D} (d + \delta_k) \cdot p_{k_1,d} + \mu_{k_2} = \sum_{d \in D} d \cdot p_{k_2,d} \qquad \forall (k_1, k_2) \in \theta \tag{14}$$

$$\sum_{k} P_k \cdot x_{k,d} \leq C_{wc} \cdot c_{wc,d} + s_{wc,d} \qquad \forall d \in D, \forall k \in K \tag{15}$$

$$\sum_{d \in D} (d + \delta_k - \beta_k) \cdot p_{k,d} \leq T_k \qquad \forall k \in K \tag{16}$$

$$\sum_{wc \in WC} c_{wc,d} \; c_{wc,d} = c_{wc,d+1} \qquad \forall d \in D \tag{17}$$

$$\sum_{wc \in WC_m} c_{wc,d} = MGT_{m,d} \qquad \forall m \in MS, \forall d \in D \tag{18}$$

$$p_{k,d} \in \{0, 1\} \qquad \forall k \in K, \forall d \in D \tag{19}$$

$$\{c_{wc,d}\} \in \mathbb{Z}^{|WC| \times |D|}_{[\min],[\max]} \qquad \forall wc \in WC, \forall d \in D \tag{20}$$

$$T_o \in \mathbb{R}^+ \qquad \forall o \in O \tag{21}$$

$$\mu_k \in [0, 1] \qquad \forall k \in K \tag{22}$$

$$s_{wc,d} \in \mathbb{R}^+ \qquad \forall wc \in WC, \forall d \in D \tag{23}$$

1.2 User interaction

User interaction is crucial for allowing the models to provide usable and implementable schedules. This aspect touches on a fundamental aspect of scheduling model in maintenance. While it is unlikely that a model will generate an implementable schedule, due to the numerous technical and human aspects found in the scheduling process itself, this does not mean that the solution space which a model creates and based on it provides solutions does not provide considerable value. So here we are faced with a problem, on the one hand it is difficult to generate implementable schedules but on the other hand constraining the schedules is valuable. To solve this problem this paper proposes a metaheuristic that is designed to continually provide schedules to the scheduler, while also accepting input from the scheduler to adjust for any unforseen job or other unknowables that the model cannot take into account.

The metaheuristic is here called **Actor based Large Neighborhood Search** and it is a extension of the Large Neighborhood Metaheuristic. Which can be seen in algorithm (!)

Algorithm 1 Actor-based large neighborhood search

```
1: Input M = message queue
2: Input schedule = initial/empty schedule
3: while true do
4:
5:     while !M.is_empty() do
6:         m = M.pop()
7:         schedule.update_schedule(m)
8:         best_schedule.update_schedule(m)
9:     end while
10:
11:    schedule^t = construct(destruct(schedule))
12:    if accept(schedule^t, schedule) then
13:        schedule = schedule^t
14:    end if
15:
16:    if C(schedule^t) < C(best_schedule) then
17:        best_schedule = schedule^t
18:    end if
19:
20:    if best_schedule.state_change() then
21:        best_schedule.send_message()
22:    end if
23: end while
```

The algorithm allows us to change the underlying parameter space of the implemented metaheuristic in real-time. In practice this means that the schedule will always be able to change to new information coming into the system, and

the metaheuristic will adapt to new parameter space by finding a new solution. This also highlight the key reason behind the choice of the large neighborhood search meta-heuristic as it has as an intrinsic part of its implementation the ability to fix infeasible solutions which is an essential properity to have when the paraspace space is subject to change.

1.3 Continuous Optimization

The idea of having a single model running continuously may not seem to produce a lot of value and this is partly correct, in practice a model may only need to be run a couple of times per week in a practical setting making it run in real-time look a little excessive. But where this approach starts gaining value is when we consider a multi-process setup where each process would ideally be allowed to change independently of each other in an asynchronuos fashion, but in a practical setting is unable to as the multi-process coordination is based on physical meetings. By having continuous optimization it becomes possible for processes to handle changes in the parameter space as they occur and the process experiencing the change will be able to optimize around it and communicate the new solution to its neighboring processes. This idea is illustrated in Figure **??**

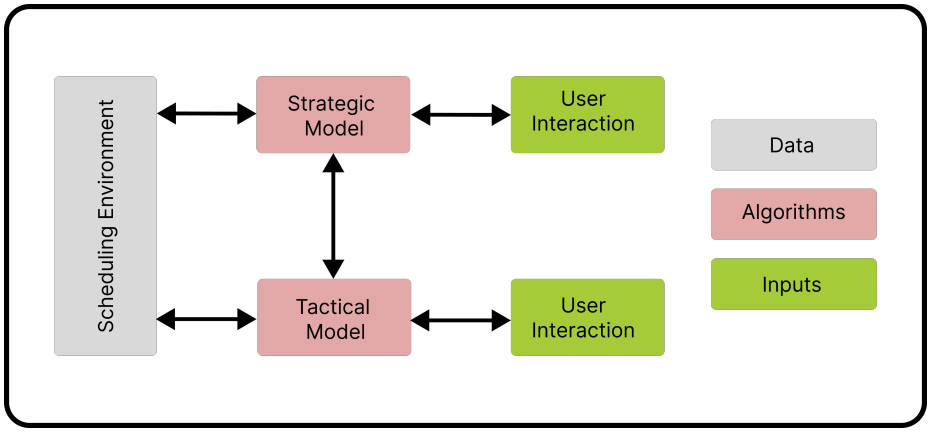

Fig. 1. A figure caption is always placed below the illustration. Please note that short captions are centered, while long ones are justified by the macro package automatically.

2 Results

3 Future Research

The project is quite extensive and requires collaboratos that have a solid foundation in both operation research and programming. Future succes of the project

will depend heavily on correct implementaion of the multi-agent system that underlies the model interactions. Future research will

Sample Heading (Third Level) Only two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run-in headings.

Sample Heading (Fourth Level) The contribution should contain no more than four levels of headings. Table **??** gives a summary of all heading levels.

Table 1. Table captions should be placed above the tables.

Heading level	Example	Font size and style
Title (centered)	**Lecture Notes**	14 point, bold
1st-level heading	**1 Introduction**	12 point, bold
2nd-level heading	**2.1 Printing Area**	10 point, bold
3rd-level heading	**Run-in Heading in Bold.** Text follows	10 point, bold
4th-level heading	*Lowest Level Heading.* Text follows	10 point, italic

Displayed equations are centered and set on a separate line.

$$x + y = z \tag{24}$$

Please try to avoid rasterized images for line-art diagrams and schemas. Whenever possible, use vector graphics instead (see Fig. **??**).

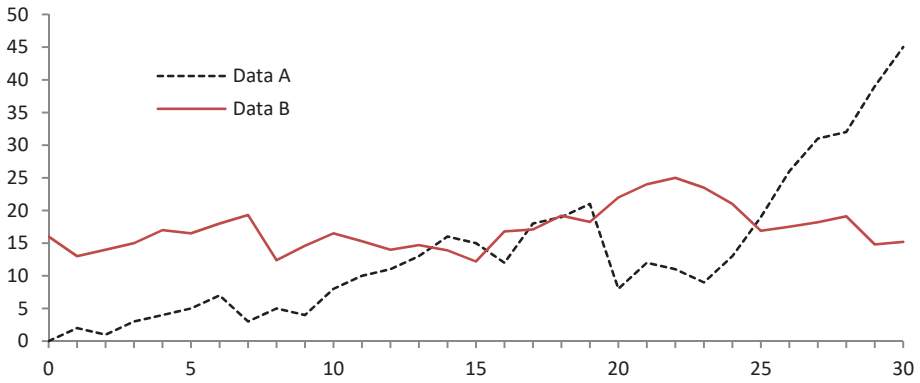

Fig. 2. A figure caption is always placed below the illustration. Please note that short captions are centered, while long ones are justified by the macro package automatically.

Program listings or code snippets in the text like **var** i:**integer**; should be set in typewriter font. The "listings" macro package can be used for reader-friendly syntax highlighting as in the following Pascal sample code:[1]

```
for i:=maxint to 0 do
begin
  { do nothing }
end;
Write('end of sample code');
```

Theorem 1. *This is a sample theorem. The run-in heading is set in bold, while the following text appears in italics. Definitions, lemmas, propositions, and corollaries are styled the same way.*

Proof. Proofs, examples, and remarks have the initial word in italics, while the following text appears in normal font.

For citations of references, we prefer the use of square brackets and consecutive numbers. Citations using labels or the author/year convention are also acceptable. The following bibliography provides a sample reference list with entries for journal articles [**?**], an LNCS chapter [**?**], a book [**?**], proceedings without editors [**?**], and a homepage [**?**]. Multiple citations are grouped [**?**,**?**,**?**], [**?**,**?**,**?**,**?**].

Acknowledgements Please place your acknowledgments at the end of the paper, preceded by an unnumbered run-in heading (i.e. 3rd-level heading).

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/123456789_0
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017

[1] For typesetting pseudocode, the "algorithmic", "algorithm2e", "algorithmicx", and "program" packages are also worth considering.