Optimizing a Maintenance Scheduling System Through the Use of Atomic Pointer Swap

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

This paper presents a way to modularize metaheuristics to enable optimization of an industrial maintenance scheduling problem. The maintenance scheduling problem has been difficult to optimize due to its reliance on multiple actors that each make up the scheduling process. The approach will use atomic pointer swaps to share solutions between the different metaheuristics keeping state consistent while having as little performance overhead as possible. The system is implemented in Rust due to the high requirements on memory safety presented setup, and data is coming from Total Energies.

Keywords: Large Neighborhood Search, Actor Framework, Maintenance scheduling, Real-time Optimization, Human-centered Computing, Decision Support Systems, Concurrent Software Architecture

1. Introduction

This paper will be structured into four sections: Section 1 will explain the problem setting that necessitates the solution method; describe the relevant software architecture literature to fix the problem; and provide mathematical models for each component of the system for a detailed understanding of the problem that the optimization system aims to solve. Section **??** will describe the chosen approach to solve the system, highlighting why low-level memory operations are necessary to effective scale metaheuristics. Section **??** will provide a high level understanding of how the system is behaving. Finally

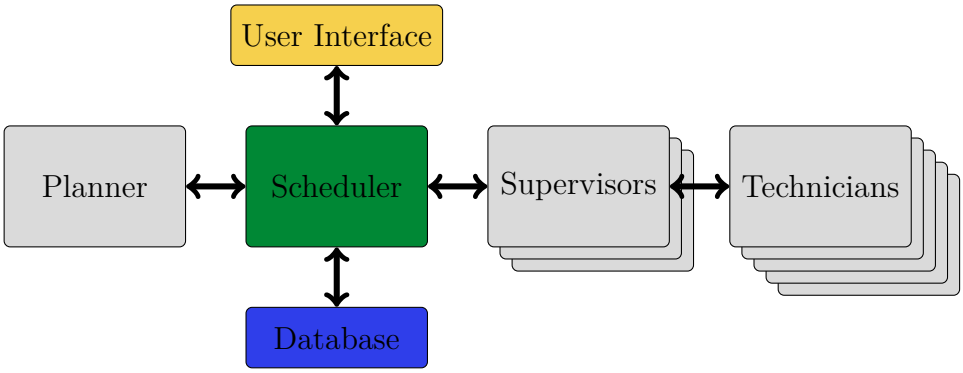

Fig 1: Simple overview of the scheduling process with its primary types of actors. The planner, the scheduler, the supervisor(s), and the technicians. The green color highlights the scheduler as it the actor in the maintenance scheduling process that is modelled in this paper.

Section **??** will discuss the implication of the provided solution approach and which other optimization problems could benefit from the presented optimization approach.

1.1. Problem Setting

Many operational optimzation problems are difficult to solve in practice as the underlying business process is made up of many different actors. This makes single model approaches difficult as it is difficult to assign responsibility for the output of the solution provided by the optimization approach (**?**). Furthermore this decomposition of business process into specific actors is not only done to make the underlying optimization problem easier to solve, but also to handle dynamic changes in the optimization process and make each process compliant with respect to safety, regulation, etc.

Maintenance scheduling has all these components and it is one of the keys reasons that the presented solution method was developed.

1.2. Software Architecture

Metaheuristics present unique challenges when it comes to designing a high level software architecture where the metaheuristics can be reliably implemented and scale. One of the reasons for this is that metaheuristics have very high requirements when it comes to state mutation. State mutation is the primary way in which metaheuristics achieve optimization by changing

the solution space rapidly according to special rules (Gendreau and Potvin, 2019).

If you want to scale metaheuristics if becomes necessary to determine a software architecture that can accomodate high levels of state changes. Most attempts at this has centered around message passing (Talbi, 2009) and in more advanced setup might also incorporate methods from the multi-agent systems literature to speed up the optimization process (Mul).

This paper will present a different way of scaling metaheuristics using low-level memory operations such as atomic operations. (**?**).

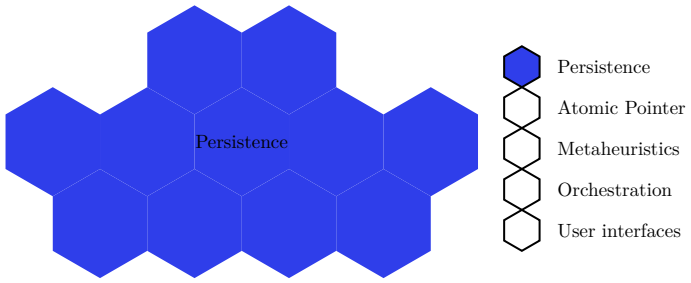

Fig 2: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

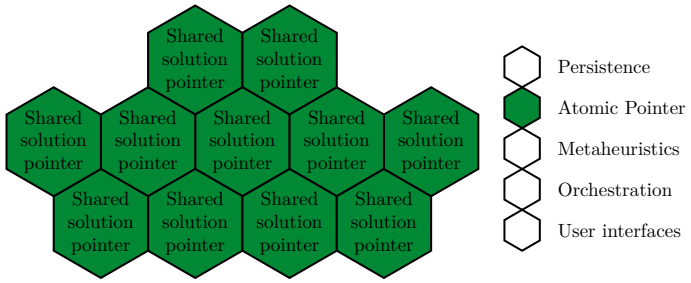

Fig 3: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

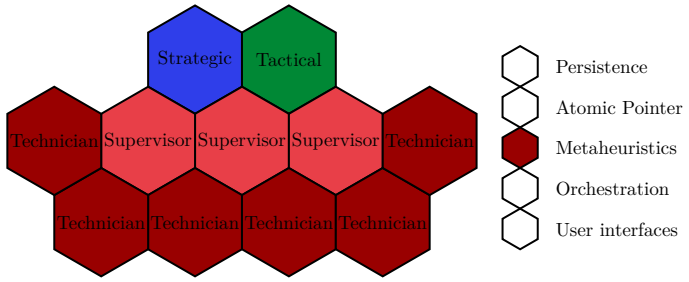

Fig 4: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

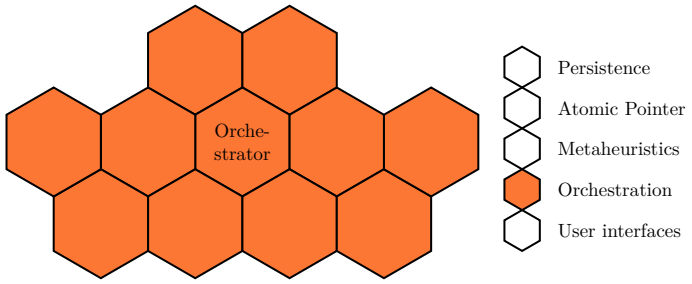

Fig 5: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

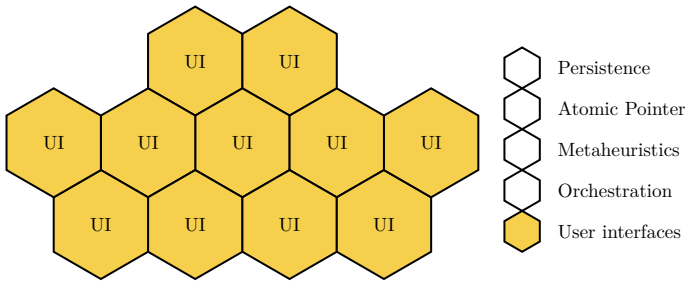

Fig 6: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

1.3. Mathematical Models

Each actor of the maintenance scheduling process will here be modelled by a mathematical model. There are multiple driving ideas behind these approaches:

- A model should encapsulate the decision that a single decision-maker is responsible for and nothing more
- Coordination between different actors is done by sharing relevant information by turning the solution of one actor/model into parameters of the other model.

1.3.1. Strategic Model: Scheduler

Meta variables:

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

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

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

Minimize:

$$+ \sum_{w \in W(\tau)} \sum_{p \in P(\tau)} strategic\_urgency_{wp}(\tau) \cdot \alpha_{wp}(\tau)$$

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

$$- \sum_{p \in P(\tau)} \sum_{w1 \in W(\tau)} \sum_{w2 \in W(\tau)} clustering\_value_{w1,w2} \cdot \alpha_{w1p}(\tau) \cdot \alpha_{w2p}(\tau) \tag{4}$$

Subject to:

$$\sum_{w \in W(\tau)} work\_order\_workload_{wr} \cdot \alpha_{wp}(\tau) \leq \ resource_{pr}(\tau, \beta(\tau)) + \epsilon_{pr}(\tau)$$

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

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

$$\alpha_{wp}(\tau) = 0, \quad if \quad exclude_{wp}(\tau) \ \ \forall w \in W(\tau) \quad \forall p \in P(\tau) \tag{7}$$

$$\alpha_{wp}(\tau) = 1, \quad if \quad include_{wp}(\tau) \ \ \forall w \in W(\tau) \quad \forall p \in P(\tau) \tag{8}$$

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

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

Meta variables:

$$s \in S \tag{11}$$

$$\alpha(\tau) \tag{12}$$

$$\tau \in [0, \infty] \tag{13}$$

Minimize:

$$\sum_{o \in O(\tau,\alpha(\tau))} \sum_{d \in D(\tau)} tactical\_value_{do}(\tau) \cdot \beta_{do}(\tau) + \sum_{r \in R(\tau)} \sum_{d \in D(\tau)} tactical\_penalty \cdot \mu_{rd}(\tau) \tag{14}$$

Subject to:

$$\sum_{o \in O(\tau,\alpha(\tau))} work_o(\tau) \cdot \beta_{do}(\tau) \leq tactical\_resource_{dr}(\tau) + \mu_{rd}(\tau) \forall d \in D(\tau) \quad \forall r \in R(\tau) \tag{15}$$

$$\sum_{d=earliest\_start_o(\tau)}^{latest\_finish_o(\tau)} \sigma_{do}(\tau) = duration_o(\tau) \quad \forall o \in O(\tau, \alpha(\tau)) \tag{16}$$

$$\sum_{d^* \in D_{duration_o(\tau)}(\tau)} \sigma_{d^* o}(\tau) = duration_o(\tau) \cdot \eta_{do}(\tau) \quad \forall o \in O(\tau, \alpha(\tau)) \quad \forall d \in D(\tau) \tag{17}$$

$$\sum_{o \in O(\tau,\alpha(\tau))} \eta_{do}(\tau) = 1, \quad \forall d \in D(\tau)$$

$$\sum_{d \in D(\tau)} d \cdot \sigma_{do1}(\tau) + \Delta_o(\tau) = \sum_{d \in D(\tau)} d \cdot \sigma_{do2}(\tau) \quad \forall (o1, o2) \in finish\_start_{o1,o2} \tag{18}$$

$$\sum_{d \in D(\tau)} d \cdot \sigma_{do1}(\tau) = \sum_{d \in D(\tau)} d \cdot \sigma_{do2}(\tau) \quad \forall (o1, o2) \in start\_start_{o1,o2} \tag{19}$$

$$\beta_{do}(\tau) \leq number_o(\tau) \cdot operating\_time_o \quad \forall d \in D(\tau) \quad \forall o \in O(\tau, \alpha(\tau)) \tag{20}$$

$$\beta_{do}(\tau) \in \mathbb{R} \qquad \forall d \in D(\tau) \quad \forall o \in O(\tau, \alpha(\tau)) \tag{21}$$

$$\mu_{rd}(\tau) \in \mathbb{R} \qquad \forall r \in R(\tau) \quad \forall d \in D(\tau) \tag{22}$$

$$\sigma_{do}(\tau) \in \{0, 1\} \qquad \forall d \in D(\tau) \quad \forall o \in O(\tau, \alpha(\tau)) \tag{23}$$

$$\eta_{do}(\tau) \in \{0, 1\} \qquad \forall d \in D(\tau) \quad \forall o \in O(\tau, \alpha(\tau)) \tag{24}$$

$$\Delta_o(\tau) \in \{0, 1\} \quad \forall o \in O(\tau, \alpha(\tau)) \tag{25}$$

---

(26)

1.3.3. Supervisor Model: Supervisor

Meta variables:

$$z \in Z \tag{27}$$

$$\alpha(\tau) \tag{28}$$

$$\theta(\tau) \tag{29}$$

$$\tau \in [0, \infty] \tag{30}$$

Maximize:

$$\sum_{a \in A(\tau, \alpha(\tau))} \sum_{t \in T(\tau)} supervisor\_value_{at}(\tau, \lambda_t(\tau), \Lambda_t(\tau)) \cdot \gamma_{at}(\tau) \tag{31}$$

Subject to:

$$\sum_{a \in A_o(\tau, \alpha(\tau))} \rho_a(\tau) = work_o(\tau) \quad \forall o \in O(\tau, \alpha(\tau)) \tag{32}$$

$$\sum_{t \in T(\tau)} \sum_{a \in A_o(\tau, \alpha(\tau))} \gamma_{at}(\tau) = \phi_o(\tau) \cdot number_o(\tau) \quad \forall o \in O(\tau, \alpha(\tau)) \tag{33}$$

$$\sum_{o \in O_w(\tau, \alpha(\tau))} \phi_o(\tau) = |O_w(\tau, \alpha(\tau))| \quad \forall w \in W(\tau, \alpha(\tau)) \tag{34}$$

$$\sum_{a \in A_o(\tau, \alpha(\tau))} \gamma_{at}(\tau) \leq 1 \quad \forall o \in O(\tau, \alpha(\tau)) \quad \forall t \in T(\tau) \tag{35}$$

$$\gamma_{at}(\tau) \leq feasible_{at}(\theta(\tau)) \quad \forall o \in O(\tau, \alpha(\tau)) \quad \forall t \in T(\tau) \tag{36}$$

$$\gamma_{at}(\tau) \in \{0, 1\} \quad \forall o \in O(\tau, \alpha(\tau)) \quad \forall t \in T(\tau) \tag{37}$$

$$\rho_a(\tau) \in [lower\_activity\_work_a(\tau), work_a(\tau)] \quad \forall a \in A(\tau, \alpha(\tau)) \tag{38}$$

---

(39)

Meta variables:

$t \in T(\tau)$ (40)

$\alpha(\tau)$ (41)

$\gamma(\tau)$ (42)

$\tau \in [0, \infty]$ (43)

Maximize:

$$\sum_{a \in A(\tau, \gamma_t(\tau))} \sum_{k \in K(\gamma(\tau))} \delta_{ak}(\tau) \tag{44}$$

Subject to:

$$\sum_{k \in K(\gamma(\tau))} \delta_{ak}(\tau) \cdot \pi_{ak}(\tau) = activity\_work_a(\tau, \rho(\tau)) \cdot \theta_a(\tau) \quad \forall a \in A(\tau, \gamma_t(\tau)) \tag{45}$$

$$\lambda_{a21}(\tau) \geq \Lambda_{a1last(a1)}(\tau) + preparation_{a1,a2} \quad \forall a1 \in A(\tau, \gamma_t(\tau)) \quad \forall a2 \in A(\tau, \gamma_t(\tau)) \tag{46}$$

$$\lambda_{ak}(\tau) \geq \Lambda_{ak-1}(\tau) - constraint\_limit \cdot (2 - \pi_{ak}(\tau) + \pi_{ak-1}(\tau)) \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \tag{47}$$

$$\delta_{ak}(\tau) = \Lambda_{ak}(\tau) - \lambda_{ak}(\tau) \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \tag{48}$$

$$\lambda_{ak}(\tau) \geq event_{ie} + duration_{ie} - constraint\_limit \cdot (1 - \omega_{akie}(\tau)) \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \quad \forall i \in I(\tau) \quad \forall e \in E(\tau) \tag{49}$$

$$\Lambda_{ak}(\tau) \leq event_{ie} + constraint\_limit \cdot \omega_{akie}(\tau) \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \quad \forall i \in I(\tau) \quad \forall e \in E(\tau) \tag{50}$$

$$\lambda_{a1}(\tau) \geq time\_window\_start_a(\beta(\tau)) \quad \forall a \in A(\tau, \gamma_t(\tau)) \tag{51}$$

$$\Lambda_{alast(a)}(\tau) \leq time\_window\_finish_a(\beta(\tau)) \quad \forall a \in A(\tau, \gamma_t(\tau)) \tag{52}$$

$$\pi_{ak}(\tau) \in \{0, 1\} \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \tag{53}$$

$$\lambda_{ak}(\tau) \in [availability\_start(\tau), availability\_finish(\tau)] \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \tag{54}$$

$$\Lambda_{ak}(\tau) \in [availability\_start(\tau), availability\_finish(\tau)] \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \tag{55}$$

$$\delta_{ak}(\tau) \in [0, work_{a\_to\_o(a)}(\tau)] \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \tag{56}$$

$$\omega_{akie}(\tau) \in \{0, 1\} \quad \forall a \in A(\tau, \gamma_t(\tau)) \quad \forall k \in K(\gamma(\tau)) \quad \forall i \in I(\tau) \quad \forall e \in E(\tau) \tag{57}$$

$$\theta_a(\tau) \in \{0, 1\} \quad \forall a \in A(\tau, \gamma_t(\tau)) \tag{58}$$

1.4. Contribution

Provide a software architectual pattern to allow metaheuristics to be integrated together in real-time.

2. Solution Method

2.1. Atomic Pointer Swapping

3. Results

4. Discussion

References

, . A Multi-agent Metaheuristic Optimization Framework with Cooperation .

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.

Talbi, E.G., 2009. Metaheuristics: From Design to Implementation. John Wiley & Sons, Hoboken, N.J.