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

1.1.

The future research direction is to demonstrate that the actor-based approach described here can be used to model and optimize multi-actor/multi-level scheduling processes. Figures 1, 2, 3, 4, 5 show a larger scale setup of the actor-based approach which is being developed with Total Energies. Here figure 3 shows the metaheuristics of scheduling system architecture where each of the actors run an AbLNS and that each metaheuristic will share its solutions with the other metaheuristics through atomic pointer swapping

shown in figure 2. Communicate with the end-user through userinterfaces and message passing as shown in figure 5, integrate with a persistent storage through mutex locks as shown in figure 1, and the lifecycle of each of the metaheuristics will be controlled by the orchestrator also through message passing as shown in figure 4.

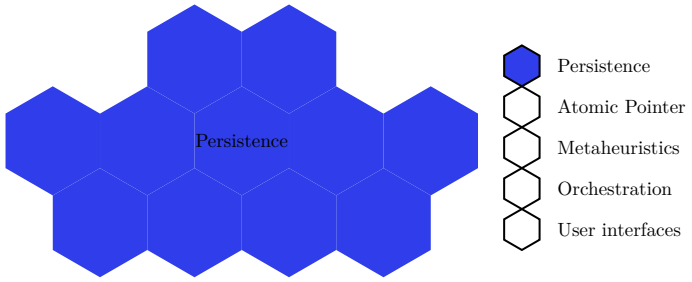

Fig 1: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

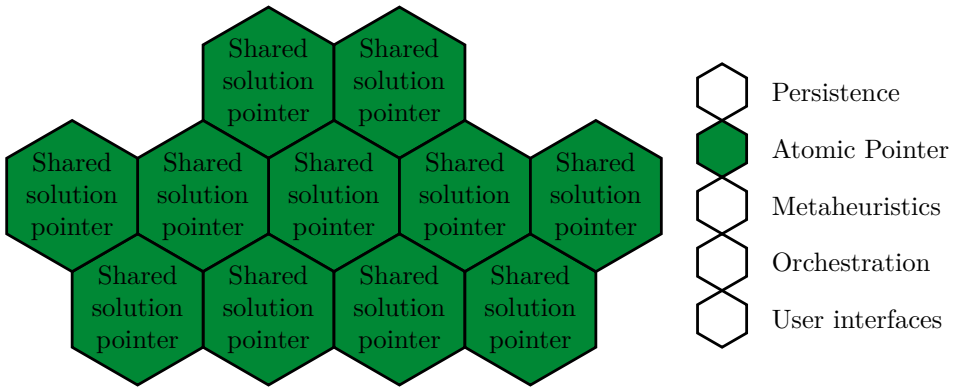

Fig 2: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

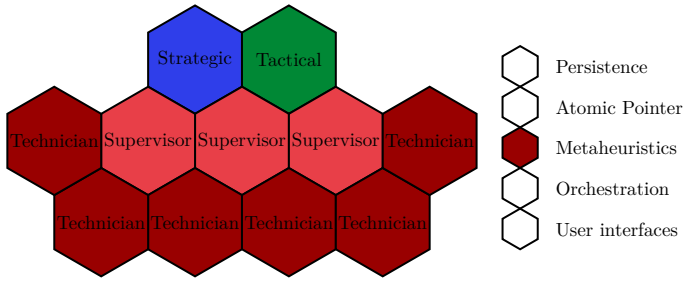

Fig 3: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

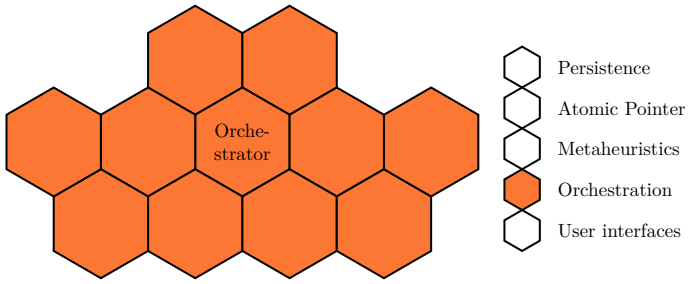

Fig 4: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

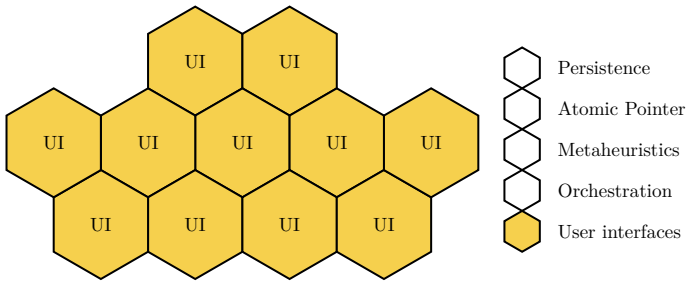

Fig 5: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.