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. Problem Setting

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 (**?**)

1.3. Mathematical Models

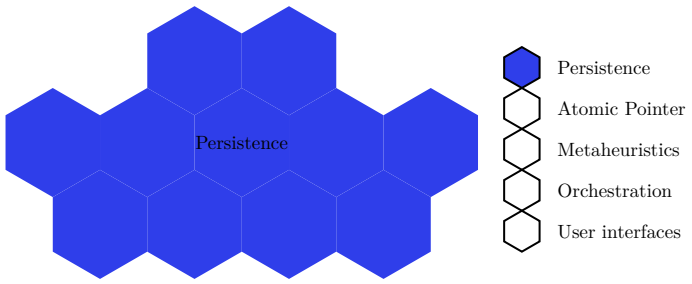

Fig 1: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

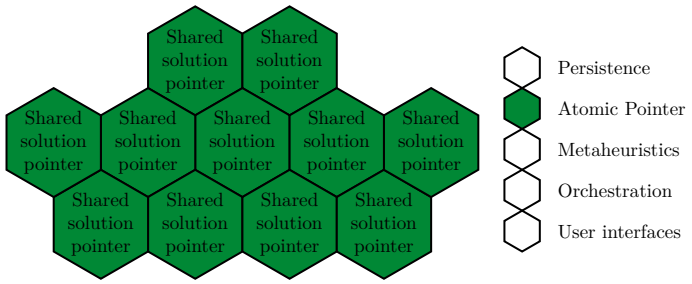

Fig 2: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

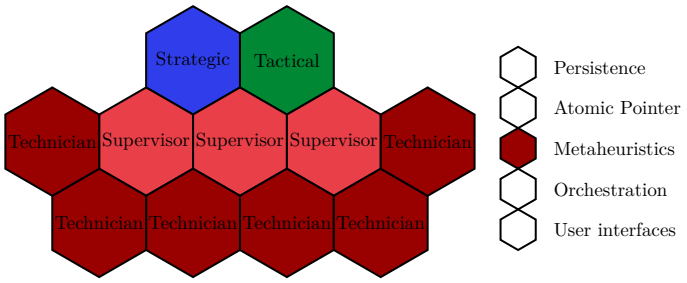

Fig 3: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

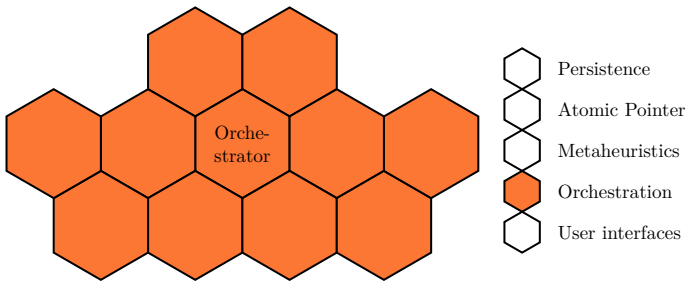

Fig 4: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

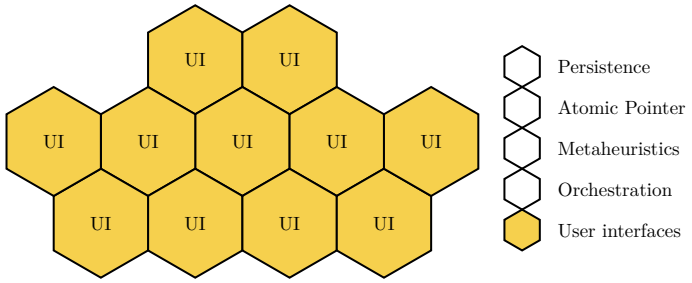

Fig 5: Overview of the scheduling process when modelled as actors. When LNS is encapsulated is an actor it becomes possible to optimize parts of a large process individually instead of optimizing the scheduling problem globally from a single model implementation.

2. Solution Method

3. Results

4. Discussion

References