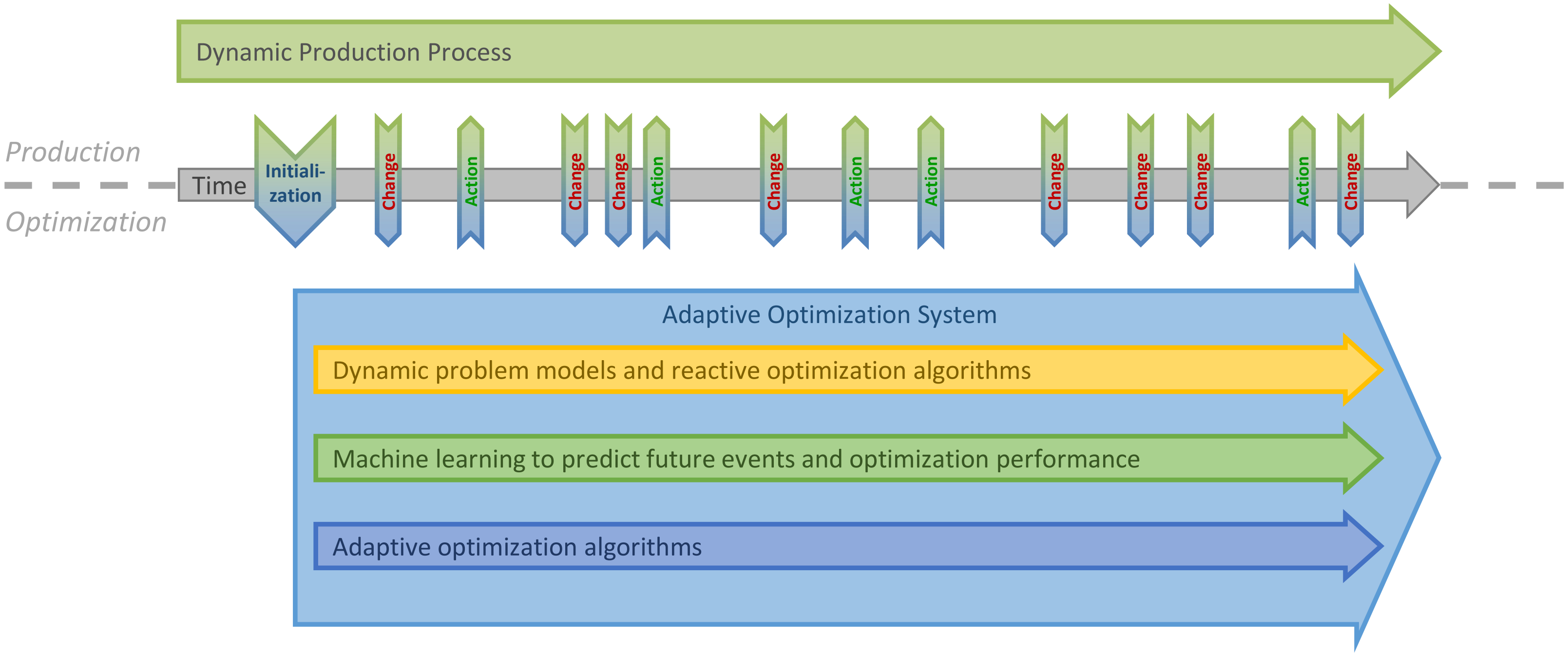
Analysis and Handling of Dynamic Problem Changes in Open-Ended Optimization

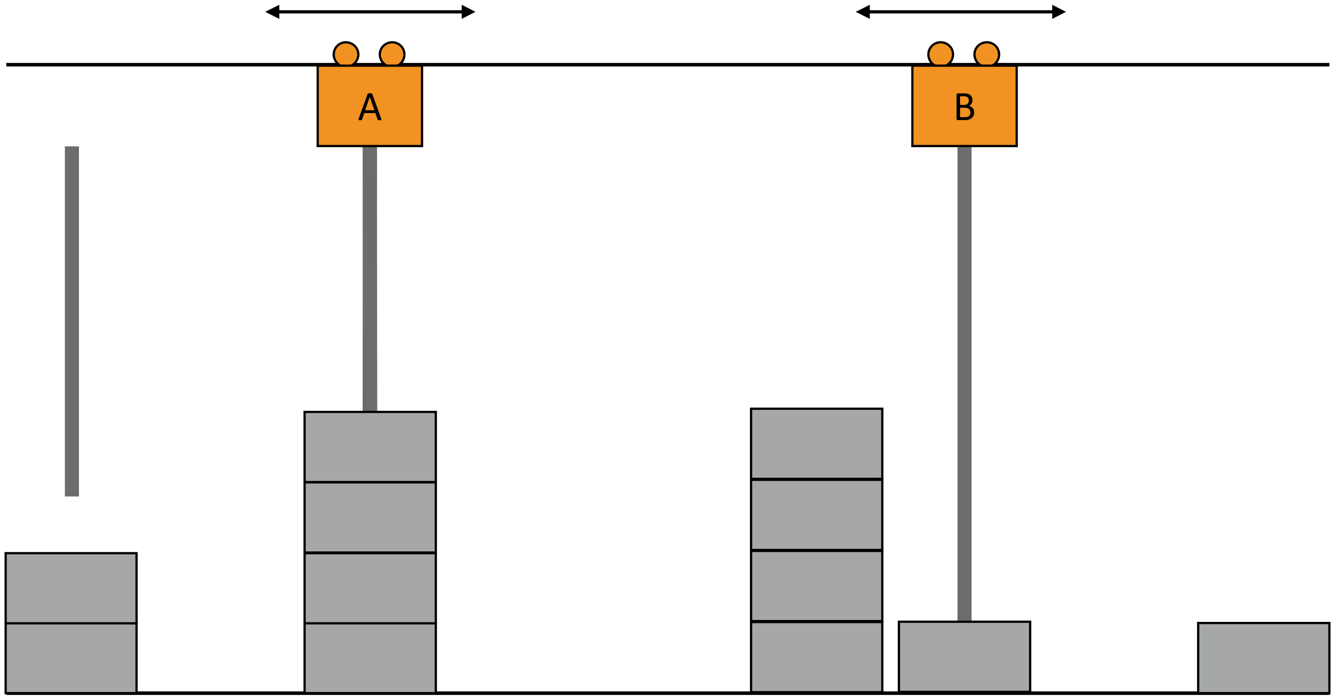
Johannes Karder, Bernhard Werth, Andreas Beham, Stefan Wagner, Michael Affenzeller

21.02.2022

Open-Ended Optimization



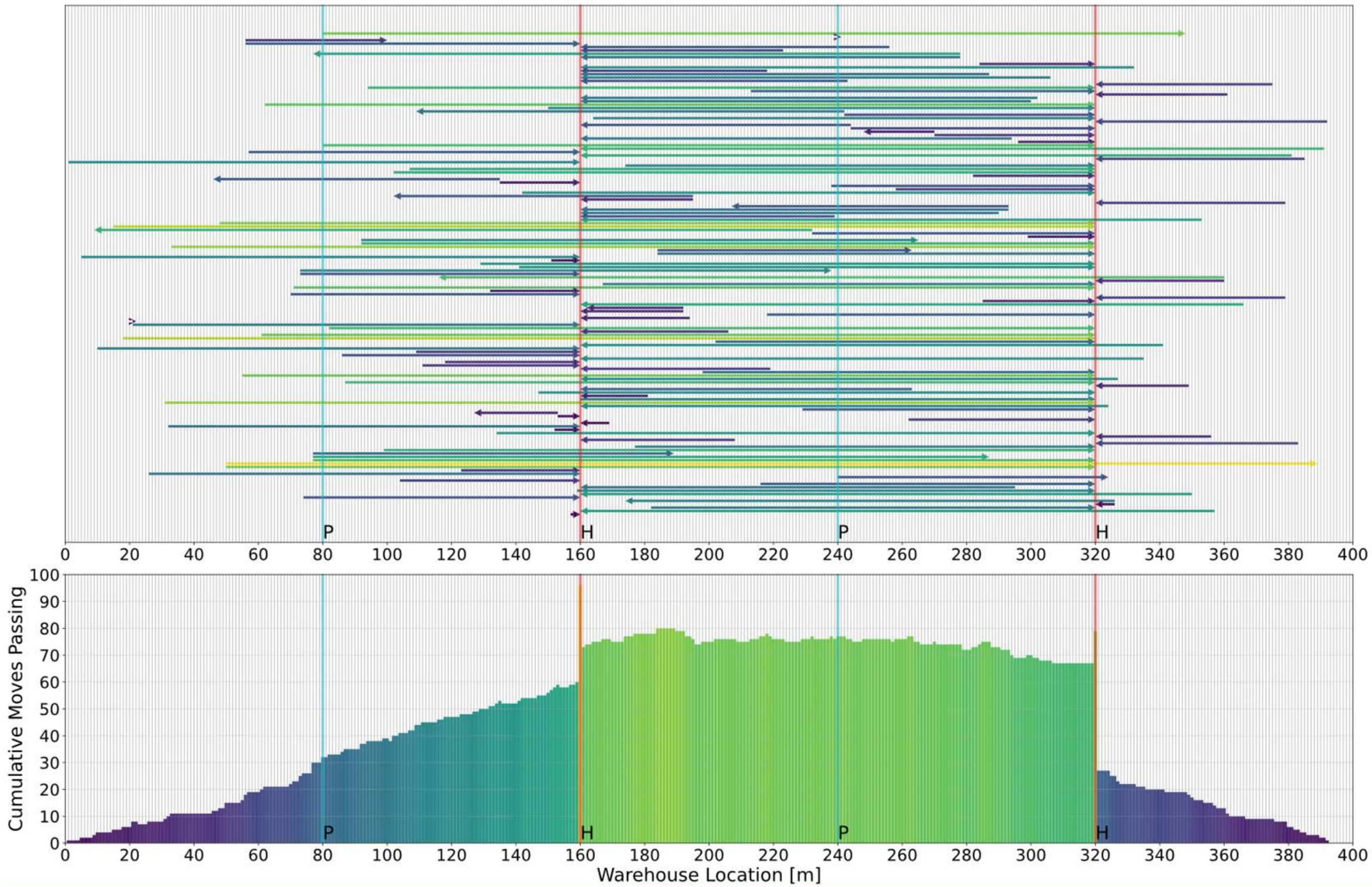
Crane Scheduling Animation



Simulated Real World

- Sim# Simulation
  - Simulates a warehouse with 2 cranes
  - Hosts multiple pickup, dropoff and storage locations
  - New moves (pickups, dropoffs, relocations) are generated over time
  - Periodically publishes world state
    - Active moves
    - Planned moves (obsolete + new)
    - Operational cranes
    - Locations (pickup, dropoff, storage)
    - …

Warehouse Hotspots



Related Literature

- Related Literature
  - Various publications on crane scheduling
  - Multiple problem formulations
  - Surveys by Boysen et al. [1, 2]
  - Only 5 publications deal with dynamic set of jobs
  - No dynamic adaption to changing process conditions

- Scientific contribution
  - Analyze effects of events on algorithm state
  - Algorithm extension to update algorithm convergence

[1] Boysen, N., Stephan, K., 2016. A survey on single crane scheduling in automated storage/retrieval systems. European Journal of Operational Research 254, 691–704.
[2] Boysen, N., Briskorn, D., Meisel, F., 2017. A generalized classification scheme for crane scheduling with interference. European Journal of Operational Research 258, 343–357.

Motivation for Self-Adaption

- Static algorithm parameterization
  - Tradeoff between fast response time and solution quality

- System is dynamic
  - Calm system state, i.e. no disruptive changes
    - Try to achieve higher solution quality
  - Disruptive changes
    - Adapt to changes fast
    - Yield good, adapted solutions fast

- Adaptive algorithm parameterization
  - Fast response time via higher selection pressure and reduced effort
  - Better solution quality via lower selection pressure and higher effort

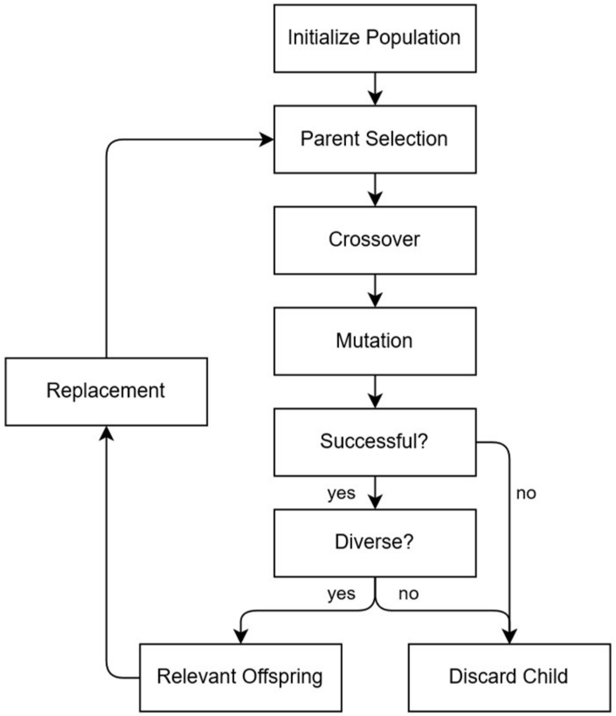
RAPGA

- Relevant Alleles Preserving Genetic Algorithm (RAPGA) [3]
  - Offspring Selection (OS) [4]
  - Elitism
  - Population size varies and is bounded

- Only accepts successful and diverse offspring
  - Relevant offspring becomes new population
  - OS criterion for minimization:

$$\mathcal{S}(c, p_1, p_2, \phi) = \begin{cases} \texttt{true}, & \text{if } f(c) < \max(f(p_1), f(p_2)) - \phi|f(p_1) - f(p_2)| \\ \texttt{false}, & \text{otherwise} \end{cases}$$

- Termination criteria
  - Maximum generations
  - Population size lower bound



[3] Affenzeller, M., Wagner, S., Winkler, S., 2007. Self-adaptive population size adjustment for genetic algorithms. In International Conference on Computer Aided Systems Theory (pp. 820-828). Springer, Berlin, Heidelberg.
[4] Affenzeller, M., Wagner, S., 2005. Offspring selection: A new self-adaptive selection scheme for genetic algorithms. In Adaptive and natural computing algorithms (pp. 218-221). Springer, Vienna.

Self-Adaptive Open-Ended RAPGA (SAOERAPGA)

- Open-Ended
  - Problem data and solutions are updated in between generations
  - Population is reseeded once algorithm converges

- Self-Adaptive
  - Effort increases over time
  - Disruptive events influence parameters
  - Switches between two basic algorithm configurations
    - Explorative mode
      - Proportional selection, weak OS, increasing effort
      - Lower selection pressure
    - Exploitative mode
      - Tournament selection, strict OS, static effort
      - Higher selection pressure

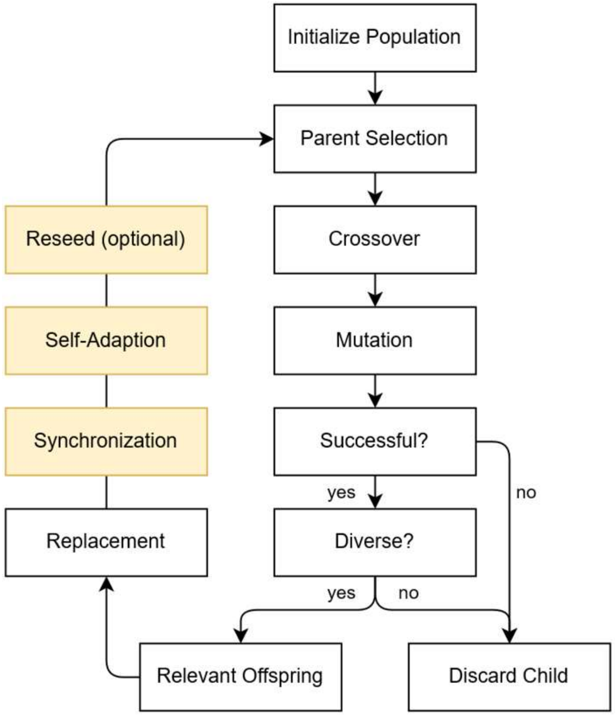
SAOERAPGA

- Adds three new steps to evolutionary cycle

1. Synchronization
   - Applies update procedures
   - Applies repair procedures
   - Communicates best solution

2. Self-Adaption
   - Switches between algorithm modes
   - Adjusts parameters

3. Population Reseeding
   - No relevant offspring found



Update and Repair Procedures

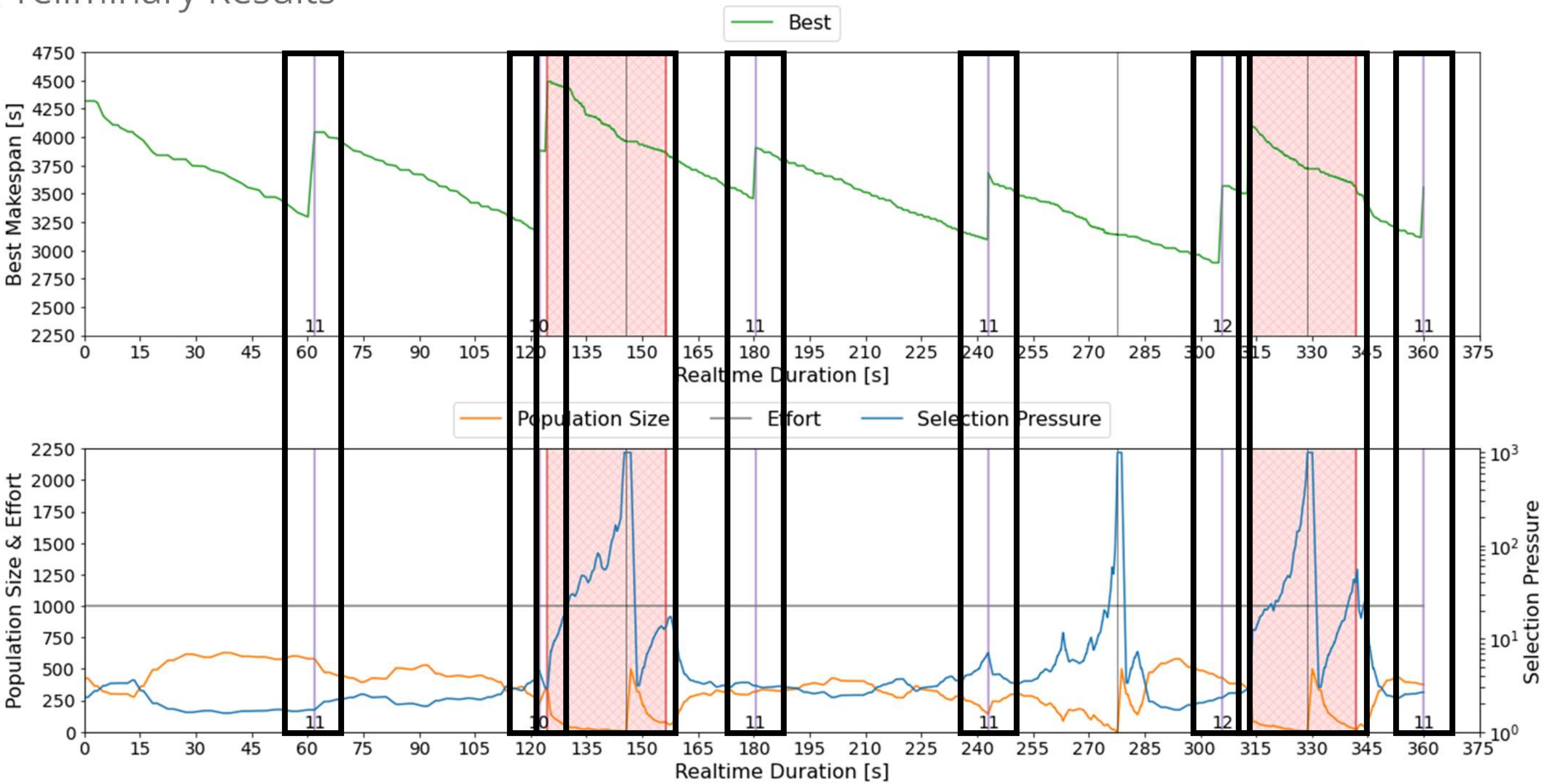
- Update Procedures
  - Problem is updated with new world state
  - New moves are inserted randomly
  - New moves are assigned random (operational) cranes

- Repair Procedures
  - Applied after each update procedure, crossover (and mutation) operation
  - Fix violated precedence constraints
  - Fix violated operational crane constraints

Behavior: Static Config

Preliminary Results

Explorative: Proportional Sel, CF 0.0, Effort 1000 (const)

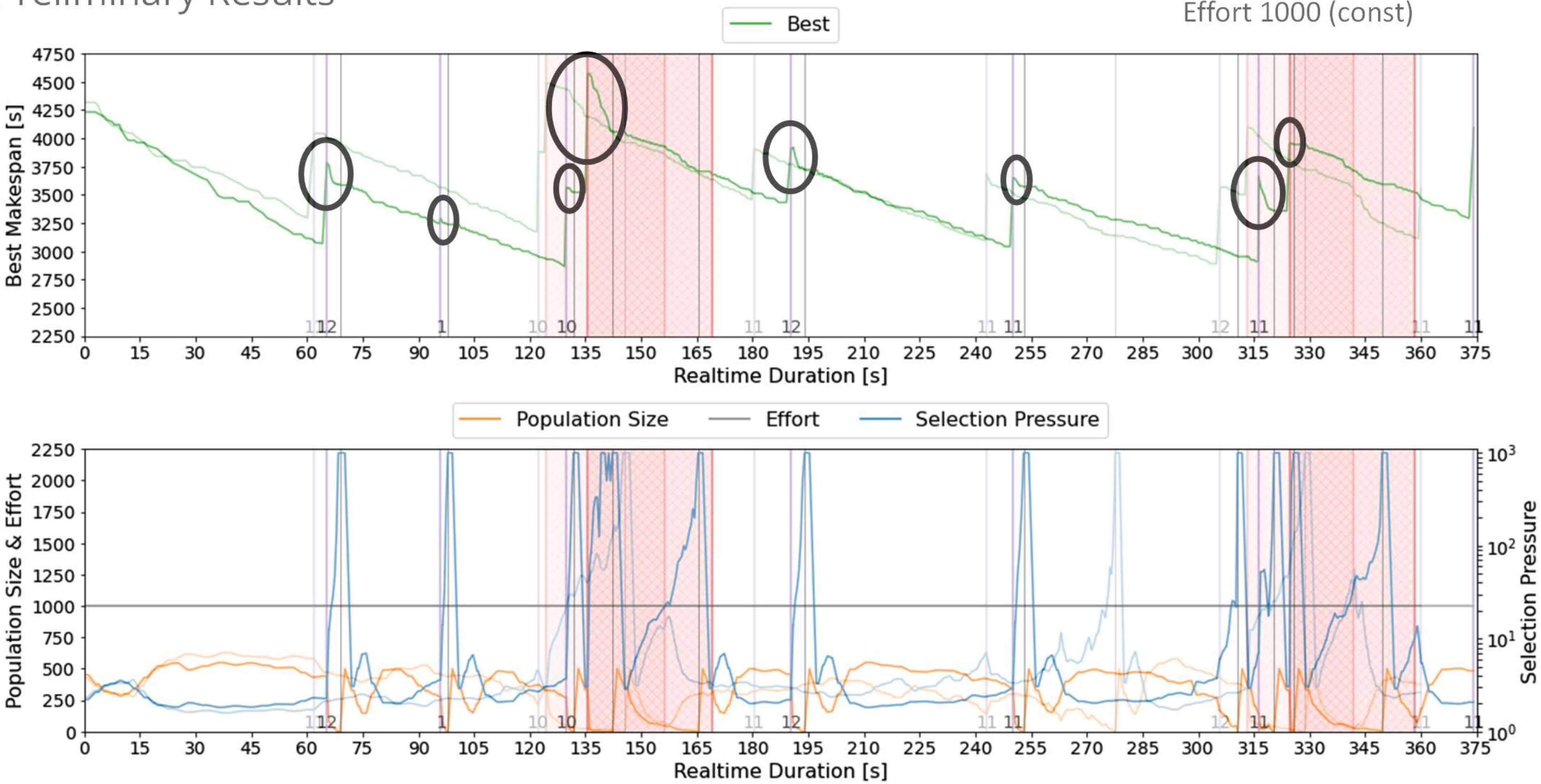


Behavior: Self-Adaptive Config 1

Preliminary Results

Explorative: Proportional Sel, CF 0.0, Effort 1000 (const)

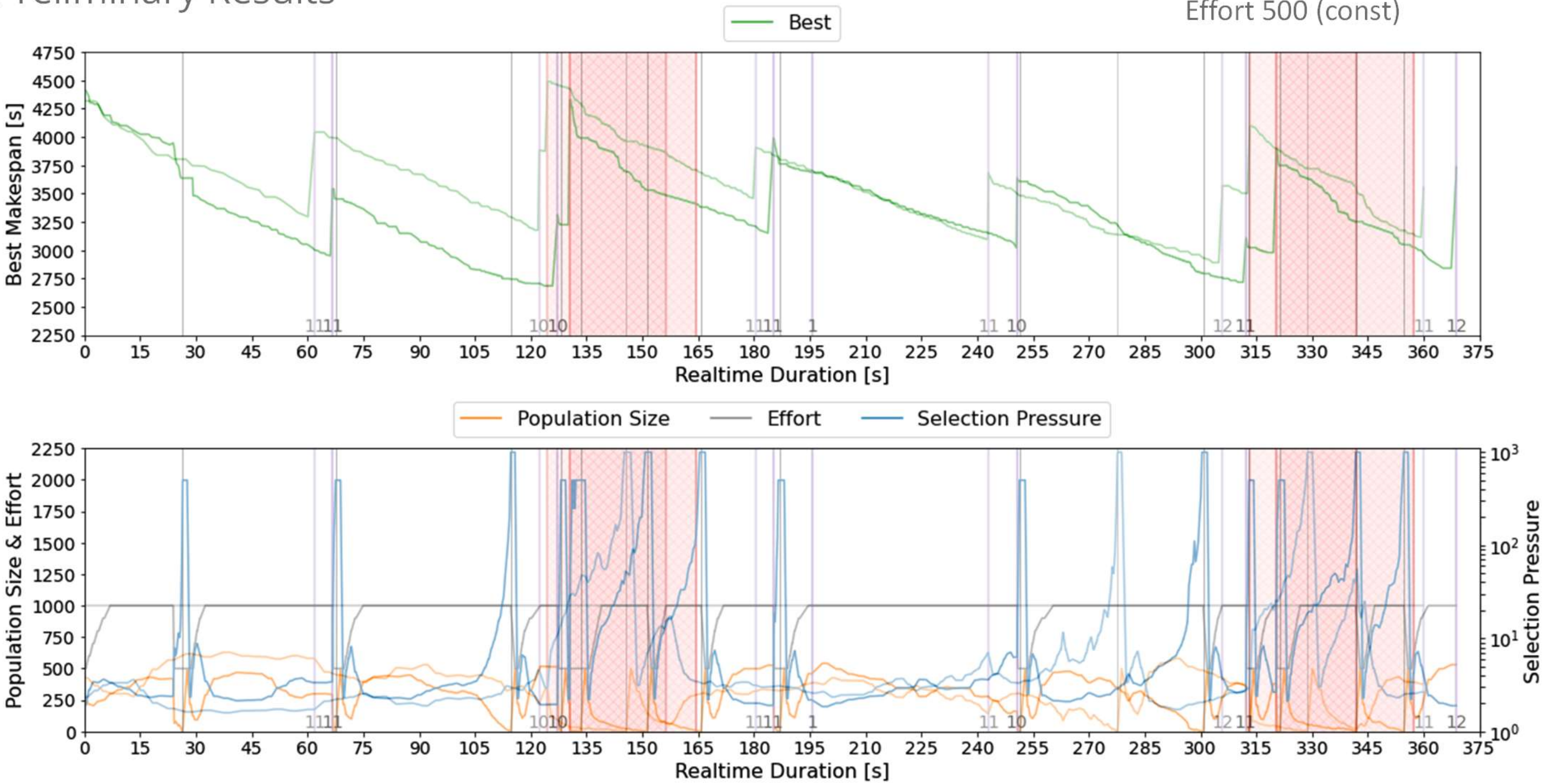
Exploitative: Tournament Sel, CF 1.0, Effort 1000 (const)



Behavior: Self-Adaptive Config 2

Preliminary Results

Explorative: Proportional Sel, CF 0.0, Effort 500-1000 (step 50)
Exploitative: Tournament Sel, CF 1.0, Effort 500 (const)



Conclusions & Outlook

- Behavior analysis of static OERAGPA (baseline) and SAOERAPGA

- SAOERAPGA able to quickly adapt to changes
  - Exploitative mode: converges faster than OERAPGA
  - Explorative mode: converges as fast as OERAPGA
    - Increased effort does not cause significantly different behavior

- Dynamic scenarios are hard to compare
  - Time-Linkage: Decisions affect problem state
  - Systems diverge over time

- More experiments necessary
  - Compare more parameter configurations and scenarios
  - Problem: CranesChangedEvent (1 → 2)
    - New genetic material only introduced after reseeding or mutation
    - Possible solution approach: Adaptive mutation rate

- We need to implement and test more open-ended versions of algorithms, e.g.
  - ALPS
  - MemPR

HEURISTIC AND EVOLUTIONARY ALGORITHMS LABORATORY