Evaluating Dynamic Scheduling Strategies for a Multi-Mode RCPSP/max Problems with generalised time-lags/no-wait constraints

Author: Jeffrey Meerovici (J.G.MeeroviciGoryn@student.tudelft.nl)
Supervisors: Mathijs de Weerdt, Kim van den Houten, Léon Planken

01. Background information

Dynamic Constraint programming: Constraint programming with stochastic durations.

The RCPSP: Find a schedule given a set of tasks and resource constrains. NP-hard

The Multi-Mode RCPSP/max with generalised time-lags/no-wait constraints extend RCPSP by:

- Multi-Mode: Multiple execution modes per task each with different durations and resource requirements.
- Generalised time-lags constraints:
  - Start of task A + lag ≤ start of task B
  - Start of task A + lag ≤ end of task B
  - End of task A + lag ≤ start of task B
  - End of task A + lag ≤ End of task B
- No-wait constraint: End of task A = Start of task B

Algorithms to compare for stochastic scheduling problems:

- Proactive approach: Creates a solution offline anticipating uncertainty. Uses the upper bound of the duration to create the solution.
- Reactive approach begins with an offline solution and modifies it during execution.
- STNU: create a STNU from model and solve it.

02. Research Question

Which dynamic scheduling algorithm (reactive, stnu or proactive) is best for solving an instance of a multi-mode RCPSP/max with generalised time-lags/no-wait constraints problem when evaluating the solution's quality, the computation time before the execution, and the computational time during the execution?

03. Methodology

Modelling the problem:

- Use pre-existing instances of deterministic multi-mode RCPSP problems
- Modify the instances to include generalised time-lags and no-wait constraints.
- Model the instance using PyJobShop modelling

Adding uncertainty to a solution:

- Solve the model
- Simulate stochastic duration using the modes' deterministic duration as mean and noise factor as variance.

Try to solve the instance using proactive, reactive and STNU algorithms.

Metrics for comparison:

- The quality of the solution
- The computation time before the execution
- The computational time during the execution

03. Comparison tests

- Wilcoxon:
  - Strong
  - Only needs one of the algorithms finding a solution; Affected by feasibility
  - Represented by bold lines in Partial ordering
- Proportion:
  - Weak
  - Only needs one of the algorithms finding a solution; Affected by feasibility
  - Represented by dash lines in partial ordering
- Magnitude
  - Strongest
  - Needs both algorithms to find a solution; Not affected by feasibility but low matches
  - Not part of partial order due to low matches

04. Feasibility rates

Problem with all constraints

Number of tasks	Noise factor 1 pro	Noise factor 1 react	Noise factor 1 STNU	Noise factor 2 pro	Noise factor 2 react	Noise factor 2 STNU
10	0.245	0.256	0.213	0.208	0.229	0.201
20	0.041	0.038	0.021	0.044	0.042	0.036

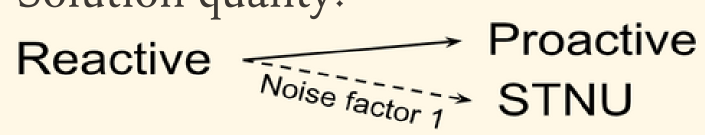
Problem without start-to-end and end-to-end constraints

Number of tasks	Noise factor 1 pro	Noise factor 1 react	Noise factor 1 STNU	Noise factor 2 pro	Noise factor 2 react	Noise factor 2 STNU
10	1.000	1.000	0.113	1.000	1.000	0.073
20	1.000	1.000	0.015	1.000	1.000	0.008

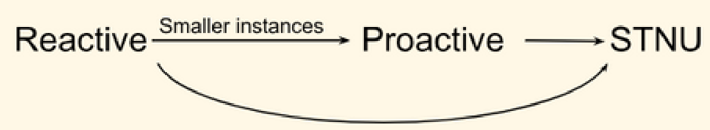
05. Results of the comparison tests

Problem with all constraints

Solution quality:



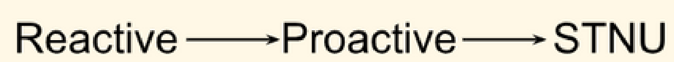
Offline time:



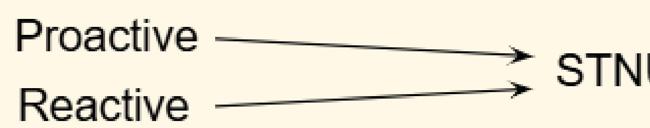
Online time:

Problem without start-to-end and end-to-end constraints

Solution quality:



Offline time:



Online time:

06. Discussion and Conclusion

Makespan:

- Reactive and STNU outperform proactive with magnitude test
- Magnitude test inconclusive between reactive and STNU

Offline time:

- Reactive and proactive mostly equal
- STNU considerably worse

Online time:

- Proactive is the fastest
- Magnitude test shows STNU beating reactive by a big margin

07. Future work

- Allowing precedence constraints between modes
- Limiting mode selection due to precedent task
- Allowing the mode of tasks to change dynamically during reschedules
- Giving all tasks an independent noise factor

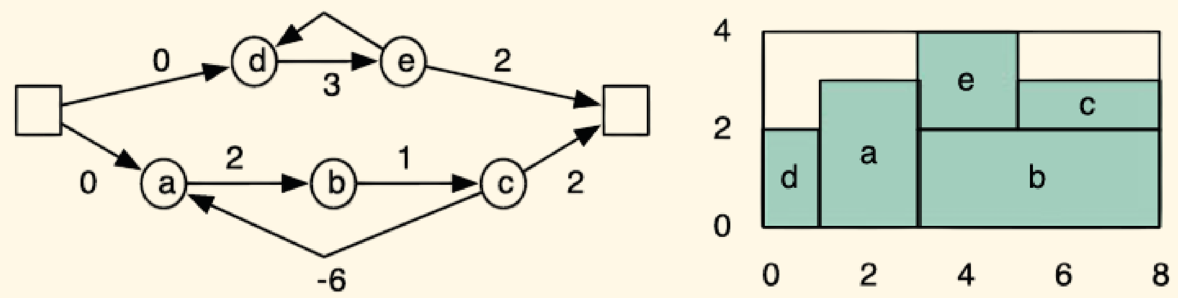


Figure 1: Precedence graph and Gantt chart of a solution of a small RCPSP/max Andreas Schutt, Thibaut Feydy, Peter J. Stuckey, and Mark G. Wallace. Solving RCPSP/max by lazy clause generation. Journal of Scheduling, 16(3):273–289, June 2013.