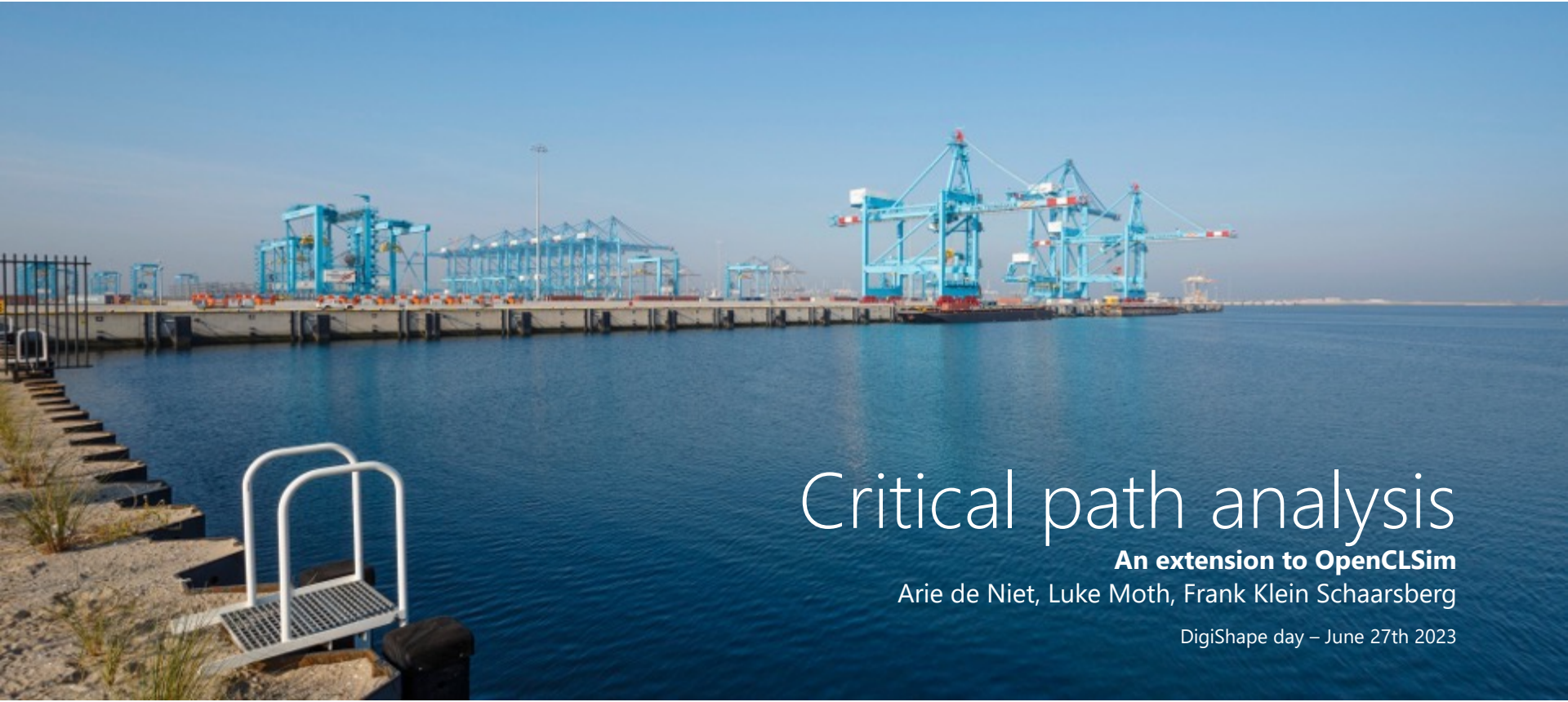


Critical path analysis

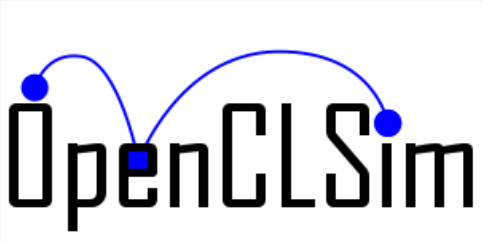
An extension to OpenCLSim

Arie de Niet, Luke Moth, Frank Klein Schaarsberg

DigiShape day – June 27th 2023

Contents

Introduction to OpenCLSim



- Open source `Python` tool for Complex Logistics Simulation
- Open community, active members TU Delft, Deltares, Van Oord and Witteveen+Bos
- Based on generic *discrete event simulation* package `SimPy`
- `OpenCLSim`: additional layer to mimic concepts of maritime transport
  - rule driven scheduling of cyclic activities
  - aims at in-depth comparison of alternative operating strategies
  - loading and unloading of material (bulk of discrete goods)
  - transport/moving

Background of critical path extension

Van Oord would like to use OpenCLSim to compare simulations:

- optimal utilization of most expensive assets
- sensitivity of the planning to disturbances
- possibilities for green steaming – sustainability

Requires identification of the critical path

Assignment to W+B and Deltares with two goals:

(1) extension of OpenCLSim with critical path module

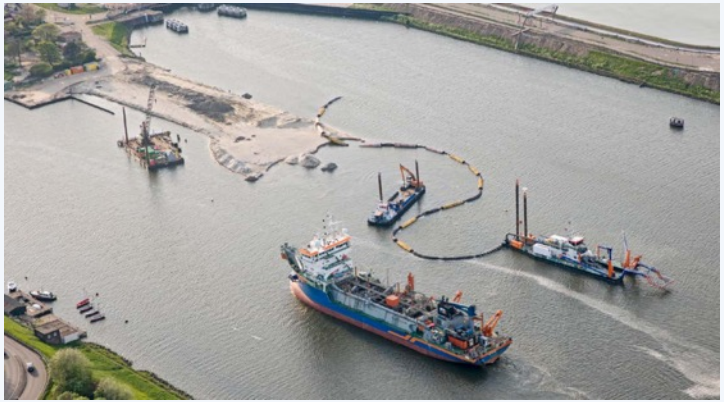
(2) increase OpenCLSim community

Critical path

- Critical path in a project:
  - the sequence of activities
  - determining the minimum time needed to complete the project
- Delay in activity on critical path ⇒ delay in project delivery
- Project management aim:
  - maximize utilization of most expensive assets

Example case – illustration

- Simple example: *one cutter, many barges*
  - *cutter* collects dredging material from a *source location*
  - material is moved by a fleet of *n barges* to a *reclamation site* until full
- Cyclic sequence of tasks for each *barge*:
  - sail empty towards the *source location*
  - *cutter* fills the *barge*
  - sail full towards the *reclamation site*
  - unload



Example case – critical path

- General aim: *full utilization of the (expensive) cutter*
  - *how many barges needed at minimum to achieve?*
- Relation to the critical path
  - *the cutter is always (mostly) on the critical path*
  - *if moving activity of barges is not critical, it can apply green/slow steaming*
- For complex (stochastic) simulations
  - *insights into processes/activities that are vulnerable*

Witteveen + Bos

Technical approach – outline

- Usual workflow
  - define `OpenCLSim` vessels and locations
  - define `OpenCLSim` activities
  - run the simulation and inspect the results
- Result: sequence of *activities* through simulation time
  - *something that happens in a certain timespan for some reason at some location with some objects involved, and possibly triggers another activity*

Witteveen + Bos

Technical approach – outline

- Usual workflow
  - define `OpenCLSim` vessels and locations
  - define `OpenCLSim` activities
  - run the simulation and inspect the results
- Result: sequence of *activities* through simulation time
  - *something that happens in a certain timespan for some reason at some location with some objects involved, and possibly triggers another activity*

Technical approach – outline

- Usual workflow
  - define `OpenCLSim` vessels and locations
  - define `OpenCLSim` activities
  - run the simulation and inspect the results
- Result: sequence of *activities* through simulation time
  - *something that happens in a certain timespan for some reason at some location with some objects involved, and possibly triggers another activity*

Example

- cutter starts *loading* barge 1 at time t_1, the loading finishes at t_2
- *because* barge 1 arrives at the cutter, and the cutter is available
- *after finishing*, barge 1 starts sailing to the reclamation site

Technical approach – outline

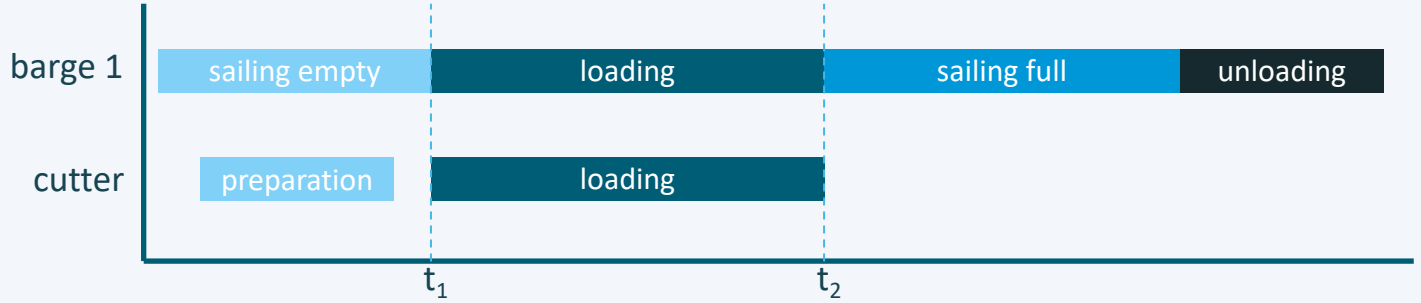
- Usual workflow
  - define `OpenCLSim` vessels and locations
  - define `OpenCLSim` activities
  - run the simulation and inspect the results
- Result: sequence of *activities* through simulation time
  - *something that happens in a certain timespan for some reason at some location with some objects involved, and possibly triggers another activity*
- All results are stored in the OpenCLSim logging
  - aim: exctract the critical path

Example

- cutter starts *loading* barge 1 at time t_1, the loading finishes at t_2
- *because* barge 1 arrives at the cutter, and the cutter is available
- *after finishing*, barge 1 starts sailing to the reclamation site

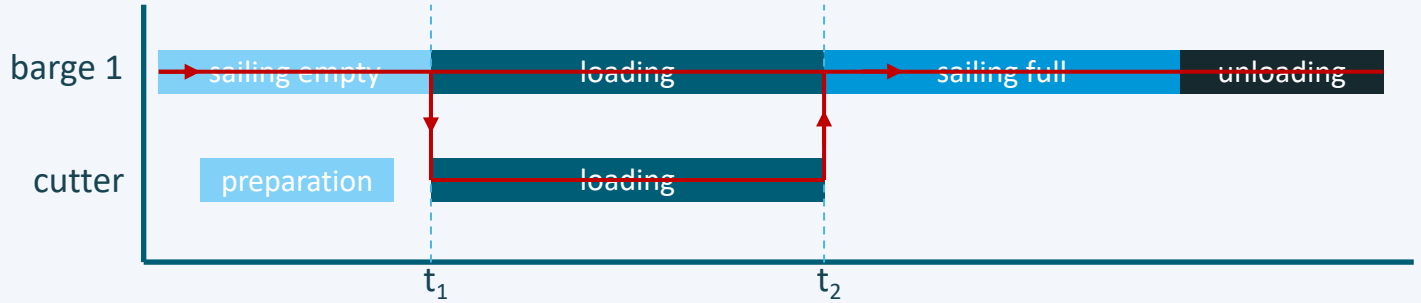
Technical approach – outline

- Example illustrates two aspects
  - which activity happened when, and which objects were involved?
  - why does something happen, specifically the relations between activities?



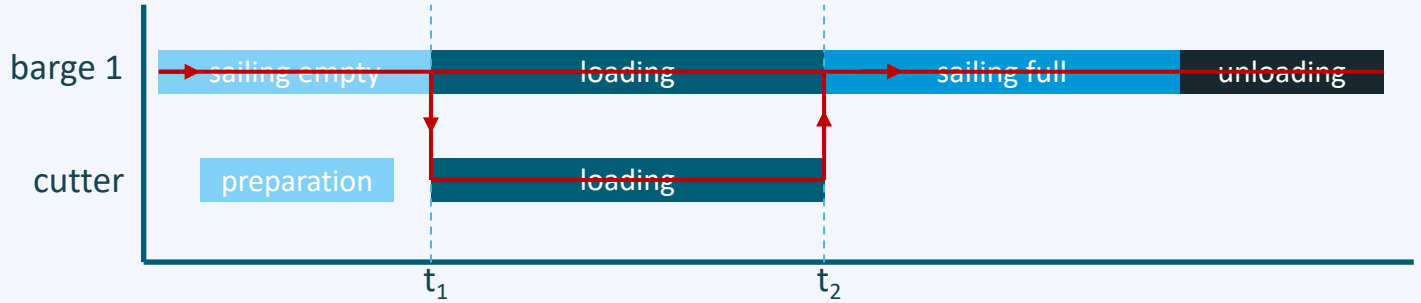
Technical approach – outline

- Example illustrates two aspects
  - which activity happened when, and which objects were involved?
  - why does something happen, specifically the relations between activities?



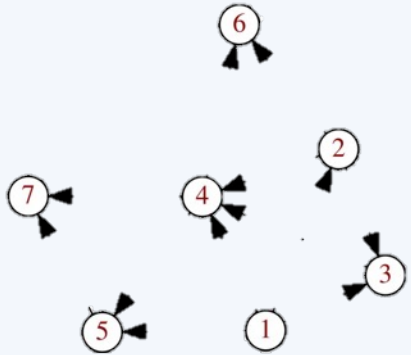
Technical approach – outline

- Technical tasks
  - (1) extract *activities* as simulated by `OpenCLSim` and their *dependencies*
  - (2) from that find the critical path

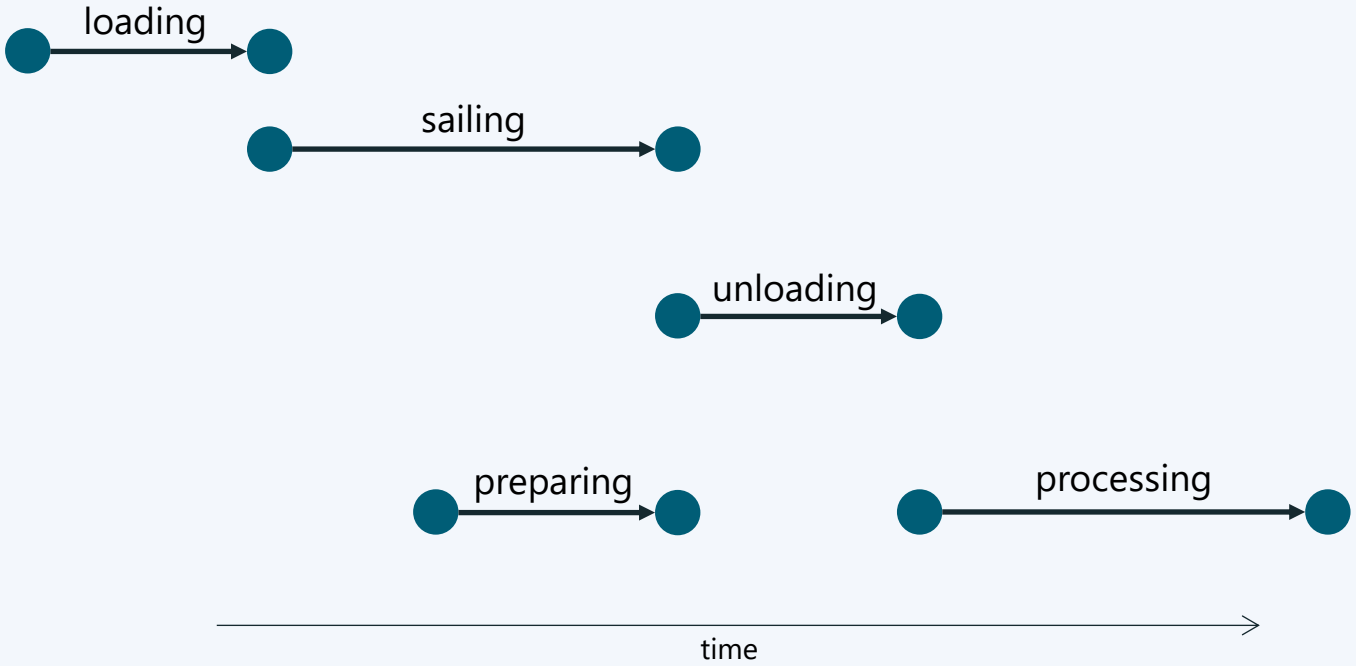


Technical approach – (1) finding the critical path

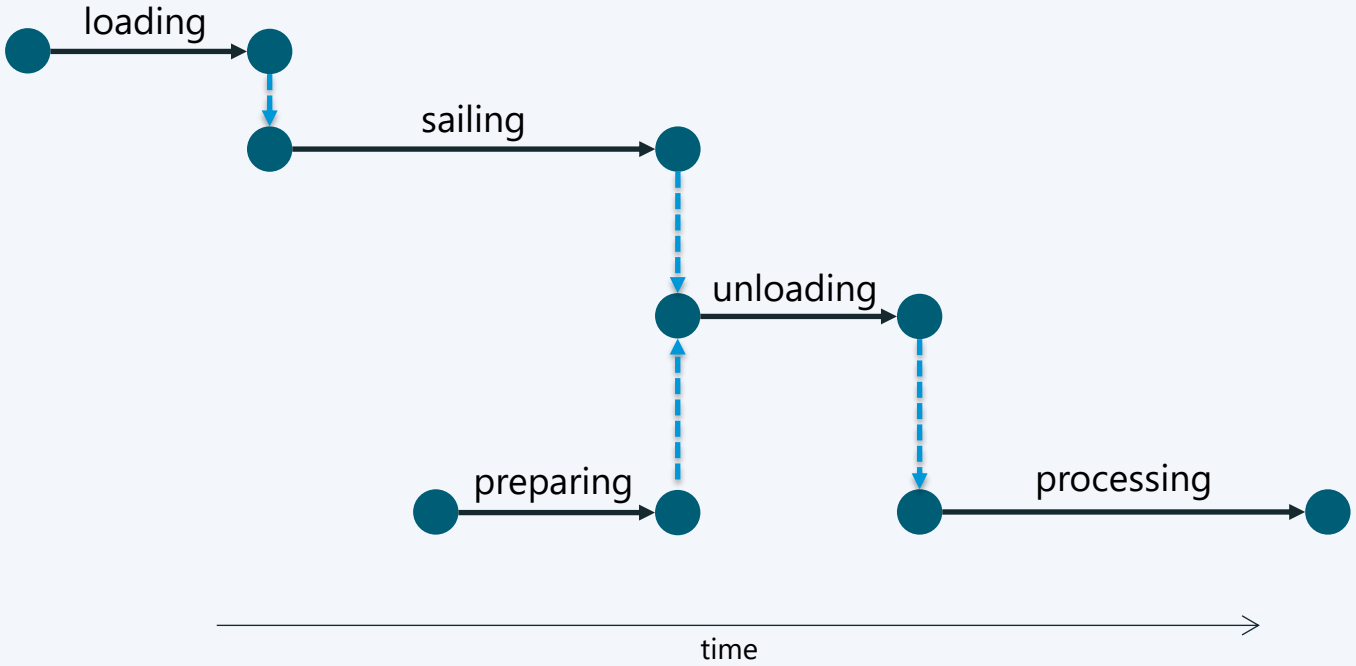
- Interdependent activities resemble *directed graph*
  - activities represent a *node-edge-node* combination from a *start* node to an *end* node
  - activity duration as weight of the *activity-edge*
  - *dependencies* are additional directed edges from the *end* of one activity to the *start* of another
- *Critical path → longest path through the graph*



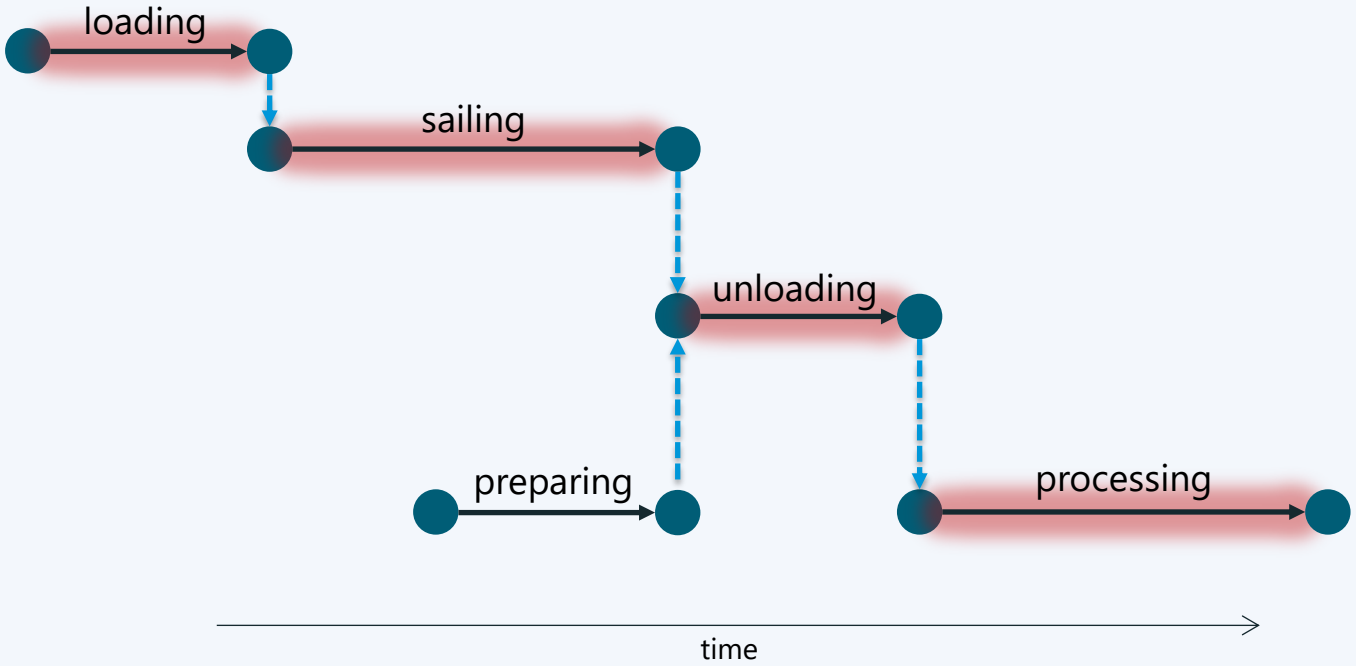
Technical approach – (2) finding the critical path



Technical approach – (2) finding the critical path



Technical approach – (1) finding the critical path



Technical approach – (1) finding the critical path

- Usage of `Python` package: `networkx`
- Pre-existing function to find *a* longest path
  - longest path is *not unique (e.g. independent parallel activities)*
  - custom-built functionality that iteratively checks all longest path
  - specific interest: *which activities are on a longest path (i.e. critical path)*

Technical approach – (1) finding th al path

Solved

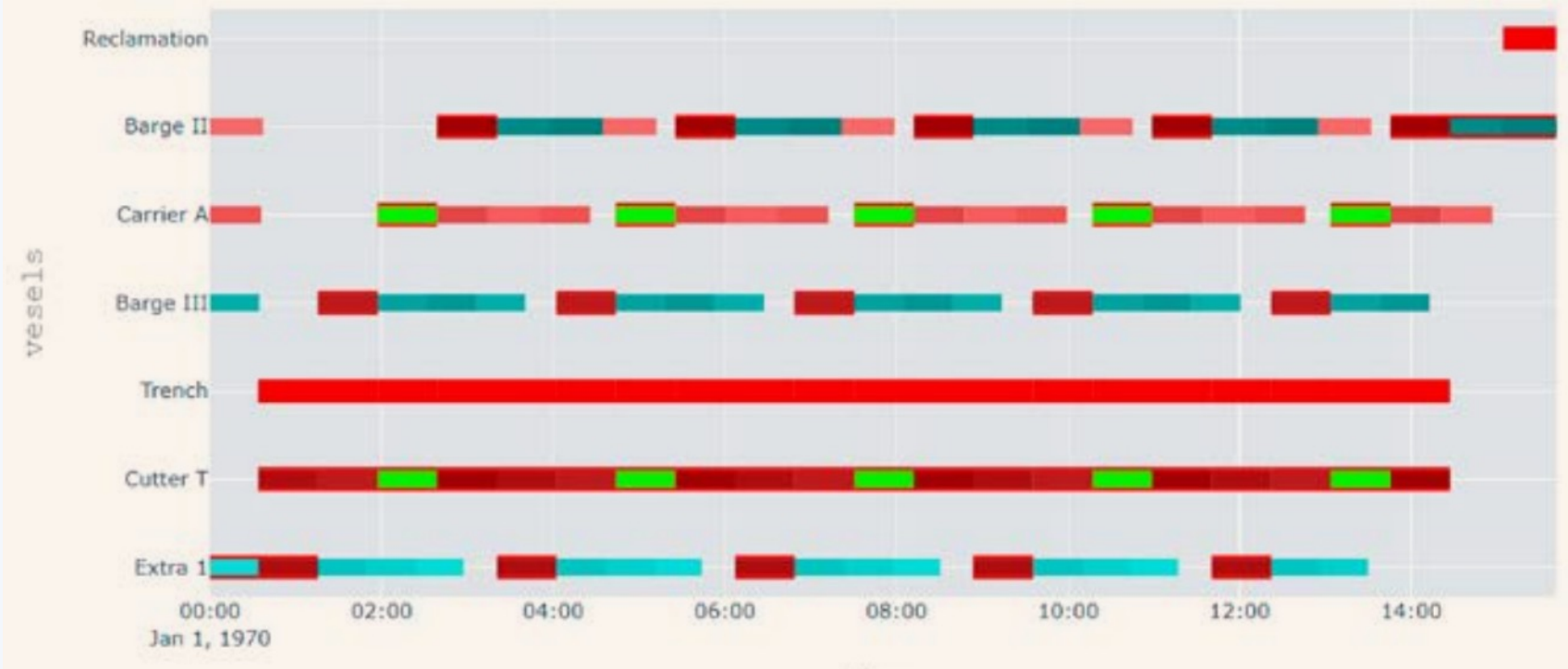
- Usage of `Python` package: `networkx`
- Pre-existing function to find *a* longest path
  - longest path is *not unique (e.g. independent parallel activities)*
  - custom-built functionality that iteratively checks all longest path
  - specific interest: *which activities are on a longest path (i.e. critical path)*

Technical approach – (2) activities and dependencies

- Needed: relevant info to build the graph
  - activities as simulated → existing `OpenCLSim` logging
  - dependencies → *not straightforward*

Technical approach – (2) activities and dependencies

- Finding dependencies
  - activity logs no info on ‘who triggered me’ or ‘who do I trigger’
  - assumptions on *start* and *end times* and *shared objects* not sufficient
- Solution
  - core functionalities of parent package `SimPy`
  - inspect the queue of activities to detect triggers of new activities
  - resulting in explicit logging of dependencies *while simulating*



Gannt chart for cutter and barges. The critical path is added as red line under and over activities.

Results

- Build-in functionality for critical path extraction from `OpenCLSim` simulations
- Visualization of the critical path
- Jupyter Notebook example
- Active member of `OpenCLSim` community, application in other projects

See also: de Boer G.J. et al., *Simulation for sustainability: alternative operating strategies for energy efficiency*, Terra et Aqua #170 - SUMMER 2023, pp. 6-17

Repository: https://github.com/TUDelft-CITG/OpenCLSim


www.witteveenbos.com