Deep Reinforcement Learning for an NP-Hard Scheduling Problem

Elisabeth Bankl, BSc

FH Technikum Wien

April 7, 2025

Supervisor: Dipl.-Ing. Dipl.-Ing. Dr. techn. Christoph Redl, BSc

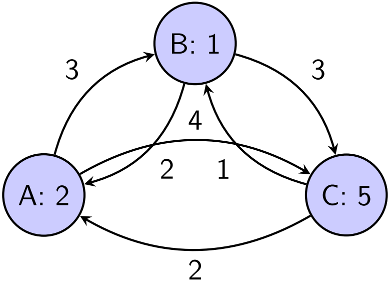
Research Questions

- How effectively can a scheduling problem be solved using Deep Reinforcement Learning?
- How close to optimal are the solutions the RL agent discovers, in percentage, depending on problem size?
- Can the RL agent compete with state-of-the-art solutions in terms of execution time and achieved optimality?

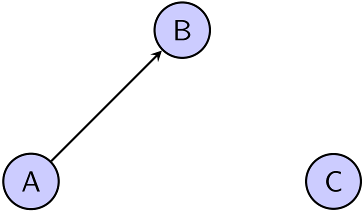
Problem Statement

- Tasks need to be completed sequentially
- Some tasks must be finished before starting others (precedence constraints)
- The execution time of a task depends on the task before it (transition time)
- Goal: Order the tasks in a way that minimizes the total execution time and follows all precedence constraints
- Sequential Ordering Problem (SOP) / Asymmetric Travelling Salesperson Problem with Precedence Constraints

Graph Representation



(a) Distances



(b) Precedence constraints

Formal Problem Definition (1/2)

Adapted from Schoen (2024).
Let $V = \{1, 2, \ldots, n\}$ be the set of nodes.
c_i: cost for visiting node i first
c_{ij}: cost for moving from node i to node j
Precedence constraints: if $p_{ij} = 1$, node i must be visited before node j.
Binary variable y_i for $i \in V$ as:

$$y_i = \begin{cases} 1 & \text{if node } i \text{ is the first node in the tour} \\ 0 & \text{otherwise} \end{cases}$$

The sum of these variables should be 1:

$$\sum_{i \in V} y_i = 1$$

Formal Problem Definition (2/2)

Binary variable y_{ij} for $i \in V$ and $j \in V \setminus \{i\}$ as:

$$y_{ij} = \begin{cases} 1 & \text{if the tour includes the edge}(i,j) \\ 0 & \text{otherwise} \end{cases}$$

$$0 \leq \sum_{j \in V \setminus \{i\}} y_{ij} \leq 1$$

$$\sum_{i \in V \setminus \{j\}} y_{ij} + y_j = 1$$

Objective: Minimize the total cost of the tour:

$$\min \sum_{i \in V} c_i y_i + \sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{ij} y_{ij}$$

Existing algorithms

- **Greedy / Nearest Neighbor Algorithm**
  Iteratively go to the nearest unvisited node Cirasella et al. (2001)
- **LKH-3**
  A variable-depth local search heuristic Helsgaun (2017).
- **Branch and Bound**
  Uses the dynamic Hungarian algorithm to find lower bounds and a local-search domination technique to prune suboptimal branches Jamal et al. (2017)
- **Ant Colony Optimization**
  A metaheuristic where artificial ants deposit pheromones on selected edges to guide optimization Skinderowicz (2013)

Markov Decision Process

- **State Space**:
  - **Distance Matrix (D)**: Distances between nodes
  - **Precedence Matrix (P)**: Which nodes must be visited before others
  - **Cost Matrix (C)**: Cost of choosing a node as the first node in the tour, later cost for choosing a node next
  - **Visited Nodes Matrix (V)**: A binary matrix indicating which nodes have been visited.
- **Action Space**: The actions are the nodes that the agent can visit next.
  - Nodes with unmet precedence constraints are not allowed.
  - Previously visited nodes are not allowed.

Reinforcement Learning Algorithm

- **Reward**:
  - Negative cost of selecting the first node in the path
  - Negative distance between nodes
- **Algorithm**: Maskable Proximal Policy Optimization (PPO) Schulman et al. (2017)
- **Network Architecture**: MLP

Dataset

- Available datasets for SOP:
  - Not enough instances for training
  - Used for benchmarking
- Randomly generate problem instances during training.
  - Distances between nodes are sampled from a uniform distribution
  - Precedence constraints are sampled from binary distribution with probability p
  - Node costs from a uniform distribution
- Distances between the nodes follow the triangle inequality $c_{ij} + c_{jk} \geq c_{ik}$
- Size: 25 nodes

Performance Evaluation

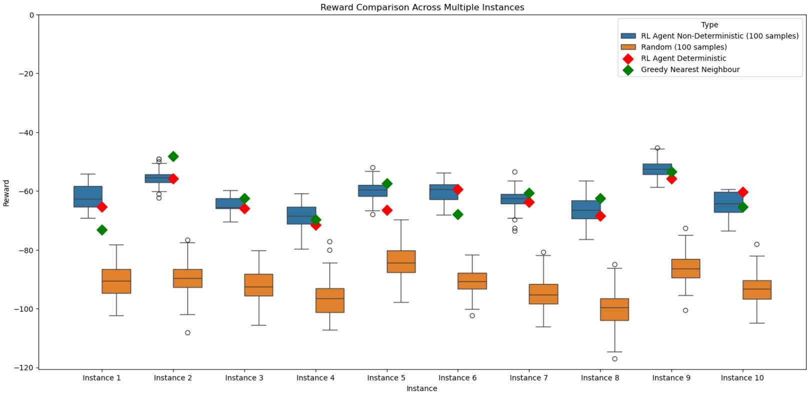


Figure: Comparing of the average reward of the agent with the simple greedy heuristic and random paths ($p = 0.1$).

Performance Evaluation depending on the precedence constraints

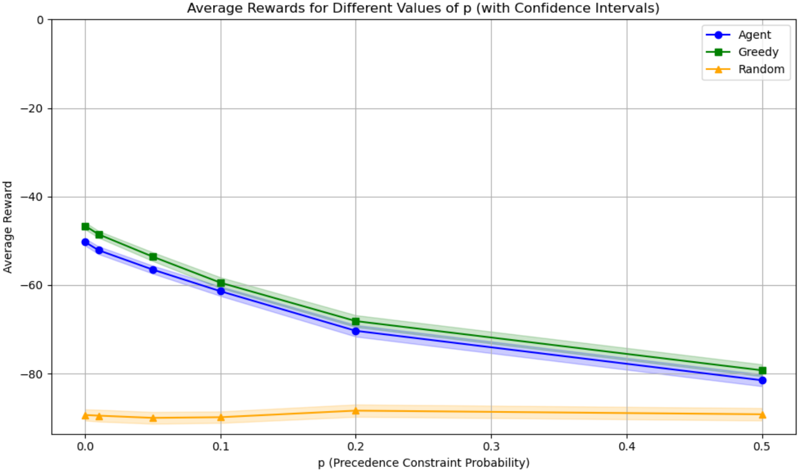


Figure: Comparing the average reward achieved by the agent, the simple greedy heuristic and random paths.

Current Limitations & Next Steps

- Maximum problem size: 25 nodes
- The RL agent's performance is still worse, on average, than the greedy heuristic.
- Change the problem instances
  - Make instance generator more deterministic
  - Clustered instances
  - Use existing instance generators
- Experiment with different RL algorithms
  - Deep Q-Learning and its variants
- Reward Shaping: use $\frac{1}{r}$ or $-r^2$ instead of $-r$
- Modify network Architecture:
  - GNN
  - Attention

References I

Cirasella, J., Johnson, D. S., McGeoch, L. A., and Zhang, W. (2001). The asymmetric traveling salesman problem: Algorithms, instance generators, and tests. In Buchsbaum, A. L. and Snoeyink, J., editors, *Algorithm Engineering and Experimentation*, pages 32–59, Berlin, Heidelberg. Springer Berlin Heidelberg.

Helsgaun, K. (2017). An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University.

Jamal, J., Shobaki, G., Papapanagiotou, V., Gambardella, L., and Montemanni, R. (2017). Solving the sequential ordering problem using branch and bound. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–9.

Schoen, F. (2024). *Optimization Models*. edito dall'autore.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

References II

Skinderowicz, R. (2013). Ant colony system with selective pheromone memory for sop. In Bădică, C., Nguyen, N. T., and Brezovan, M., editors, *Computational Collective Intelligence. Technologies and Applications*, pages 711–720, Berlin, Heidelberg. Springer Berlin Heidelberg.