Deep Reinforcement Learning Guided Improvement Heuristic for Job Shop Scheduling

Cong Zhang[1], Zhiguang Cao[2], Wen Song[3], Yaoxin Wu[4], and Jie Zhang[1]
[1]Nanyang Technological University, [2]Singapore Management University,
[3]Shandong University, [4]Eindhoven University of Technology

Paper & Code

ICLR-2024

Contributions

- ❑ A novel learning-based improvement heuristic for job-shop scheduling (JSSP).
- ❑ The proposed methd has linear computational complexity.

Preliminaries

Job-Shop Scheduling Problem (JSSP):

- ❑ A set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$, where J_i has $O_i = \{O_{i1} \ldots O_{in_i}\}$ operations $\forall 1 \le i \le n$.
- ❑ Machines set $\mathcal{M} = \{M_1, \ldots, M_m\}$.
- ❑ A predefined processing order $\mathrm{Ord}: O_i \to M^{n_i}, \forall 1 \le i \le n$.
- ❑ Objective: $min_{S_{ij}, \forall O_{ij}} C_{max} = max_{i,j}\{C_{ij} = S_{ij} + p_{ij}\}$, and S_{ij} is starting time of O_{ij}.

Disjunctive Graph (DG): $G = \{\mathcal{O}, \mathcal{C}, \mathcal{D}\}$, an example is Figure 1.

- ❑ Node set $\mathcal{O} = \{O_{ij} | \forall i, j\} \cup \{S, T\}$: S, T, $p_S, p_T = 0$, denoting the start and terminal.
- ❑ Conjunction set $\mathcal{C}$: directed arcs representing precedent constrains.
- ❑ Disjunction set $\mathcal{D}$: undirected arcs connecting operations on the same machine.
- ❑ Solution of JSSP: fixing the direction of each disjunction $\to G \in \{DAG | no\ cycle\}$.

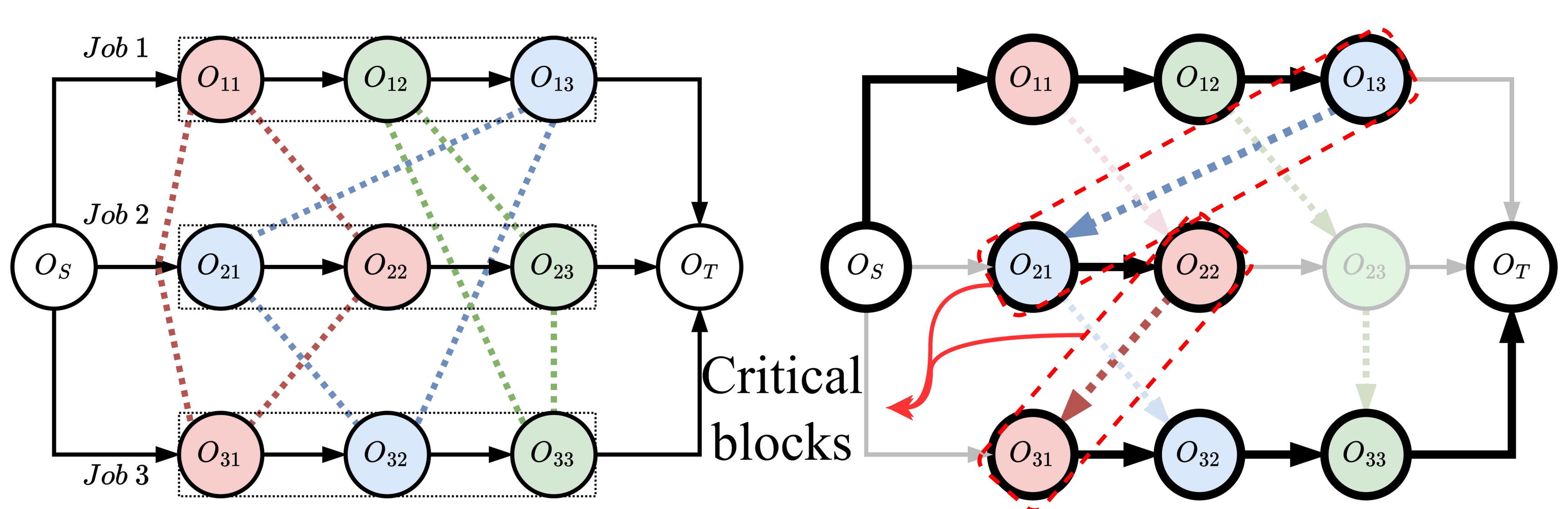


Figure 1:**Disjunctive graph representation.**

N_5 **Neighbourhood Structure**: A widely-used local operator for JSSP.

Motivation

- ❑ The performance of learning-based constructive heuristics is far from optimality.
- ❑ Neural improvement operator has linear computational complexity.

Markov Decision Process (MDP) Formulation

- ❑ **State**: Any complete solutions represented as a disjunctive graph
- ❑ **Action**: Any eligible operation paris defined by the N_5 neighbourhood structure.
- ❑ **Reward**: The difference on the objective (e.g., the makespan) between the solution at current step and the incumbent.

- ❑ **State Transition**: The new state is obtained by swapping the pair of operations in current state.

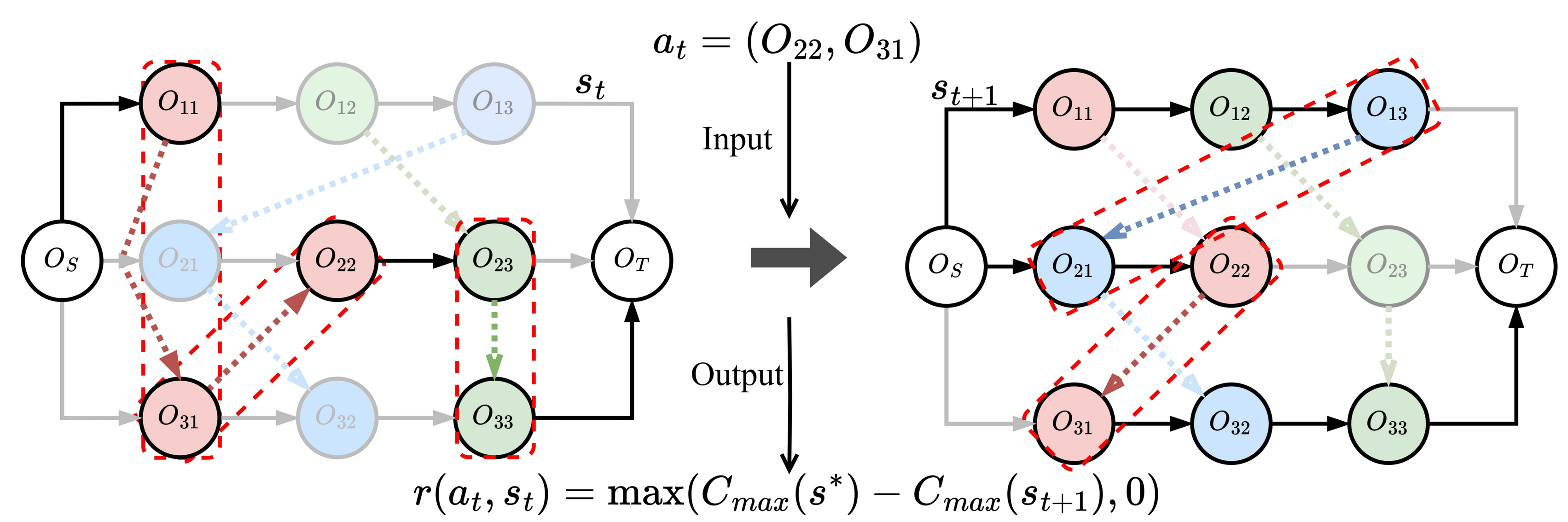


Figure 2: **Example of state transition.**

- ❑ **The improvement process**: Improving the initial solution iteratively.

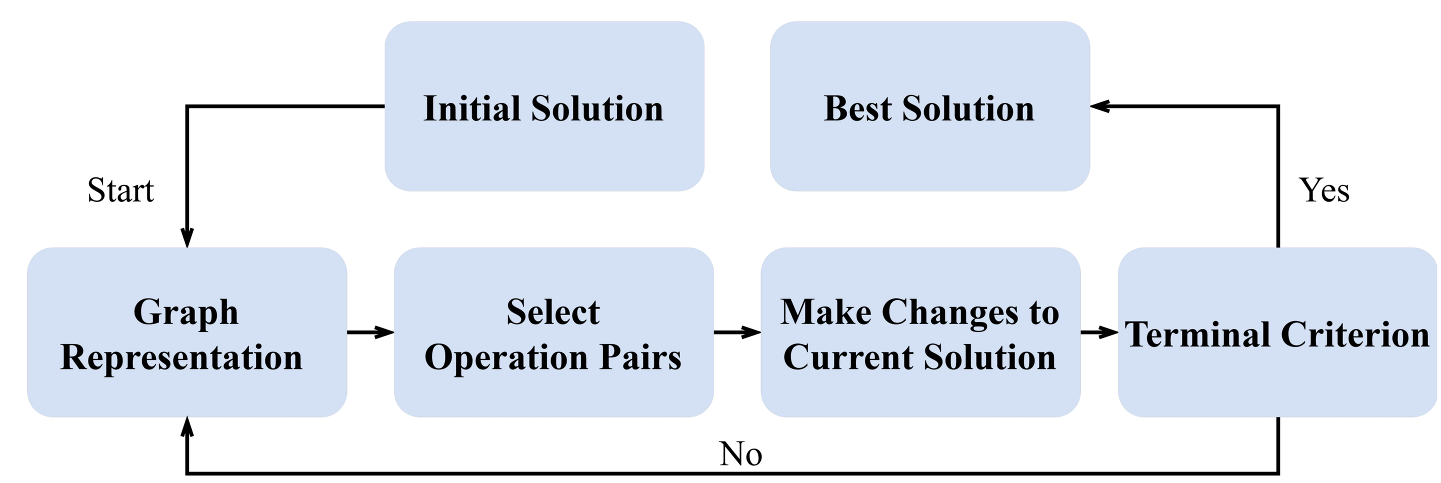


Figure 3: **The overall improvement process.**

Parameterizing the Policy

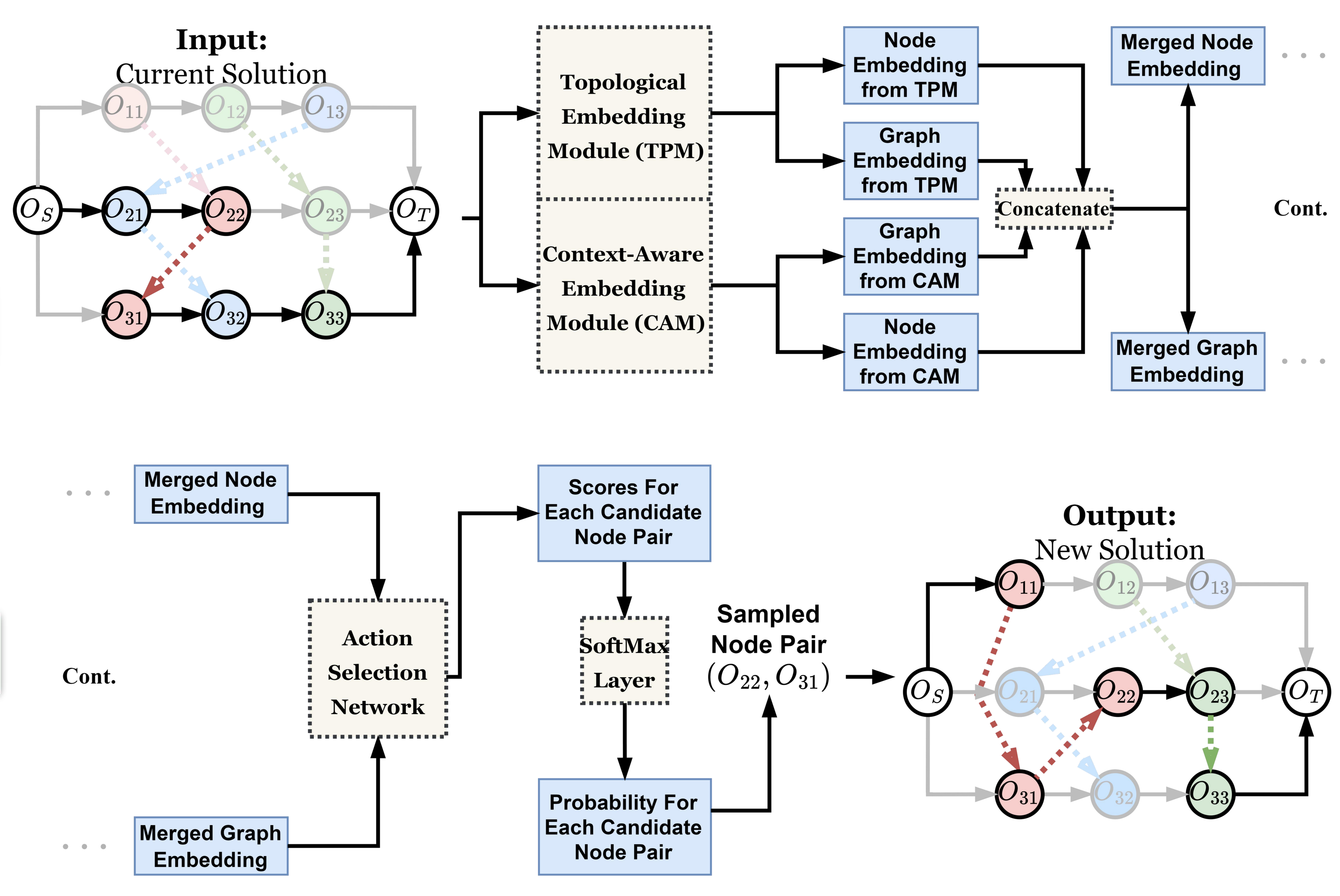
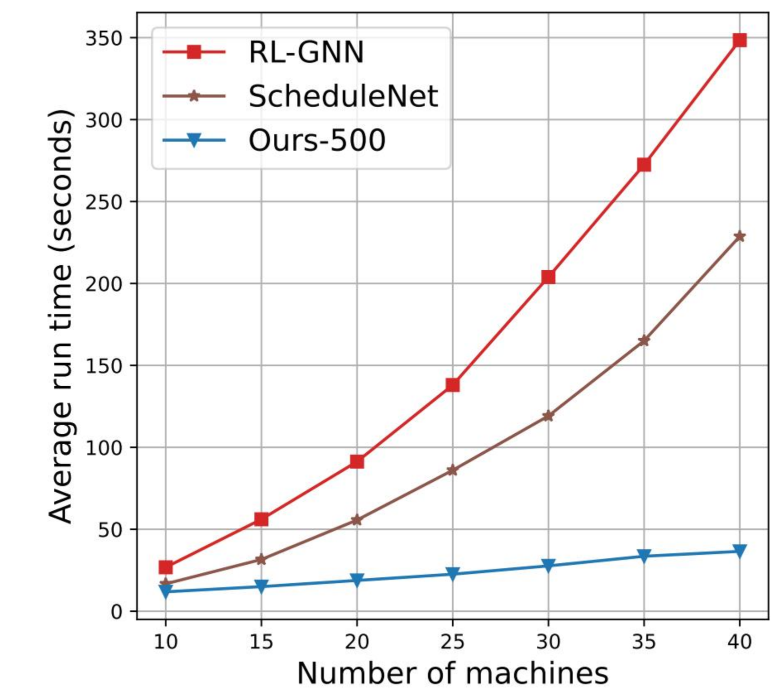


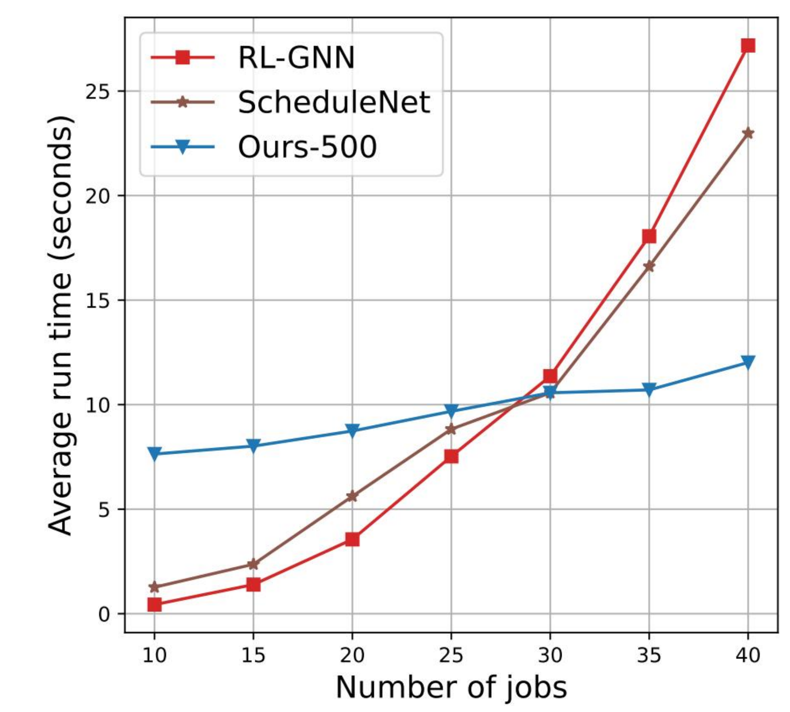
Figure 4. **Policy Network.**

Theoretical Findings

- ❑ **Theorem 4.1**: The proposed policy network has linear time complexity with respect to both |J | and |M|, where |J| and |M| are the number of jobs and machines, respectively.



(a) Fixed $|\mathcal{J}| = 40$



(b) Fixed $|\mathcal{M}| = 10$

Theorem 4.2: We propose a neural operator based on message passing mechanism that can calculate the quality of a batch of JSSP solutions with diverse size (emperically has linear computational complexity), which significantly improve the GPU utilization for learning-based methods for JSSP.

Batch size	1	32	64	128	256	512
MP (CPU)	0.051s	0.674s	1.216s	2.569s	5.219s	10.258s
MP (GPU)	0.058s	0.094s	0.264s	0.325s	0.393s	0.453s
CPM (CPU)	0.009s	0.320s	0.634s	1.269s	2.515s	5.183s
Speedup	0.16×	3.40×	2.40×	3.90×	6.42×	11.4×

Experiment Results

- ❑ **Benhmarks**: Taillard, ABZ, FT, LA, SWV, ORB, YN, and Synthetic data.
- ❑ **Baselines**:L2D (NeurIPS20), RL-GNN (IJPR21), ScheduleNet (23), conventional improvement operators (e.g., greedy), OR-Tools (Google).
- ❑ **Results**:

1) Average optimality gap: **~5%**.
2) Outperforms the exsisting learning-based methods by a large margin.