SHORT BIBLIOGRAPHY REPORT: OPTIMIZING JOB SHOP SCHEDULING WITH DEEP REINFORCEMENT LEARNING

[Master Applied AI for Digital Production Management

Deggendorf Institute Of Technology, Cham, Germany]

RIFSHU HUSSAIN SHAIK	[12505018]
ANKITH RAMESH BABU	[22403982]
ALWIN SHAJI	[22407660]
SANABOYINA SATYA NARASIMHA	[12503984]
GNANASUDHA PATUR	[12501826]

Table of Contents

1. Introduction and Project Context

1.1 The Job Shop Scheduling Problem (JSSP)

- **Definition and Classification:** The **Job Shop Scheduling Problem (JSSP)** is a classic combinatorial optimization problem that involves sequencing a set of N jobs across a set of M machines, subject to stringent constraints. JSSP is globally recognized as an **NP-hard** problem [1]. The definition requires finding the optimal sequence of operations on machines to satisfy production goals.
- **Computational Implication:** The NP-hard classification implies that the computational effort required to find the absolute global optimal schedule scales exponentially with the number of jobs and machines increases. Consequently, for any realistic industrial-scale problem, finding an exact mathematical solution is infeasible. This necessitates the adoption of efficient heuristic or learning-based approaches capable of finding high-quality, near-optimal solutions quickly.
- **Core Constraints:** The complexity arises from the simultaneous satisfaction of strict constraints:
    - **Precedence Constraint:** Each job consists of a sequence of operations that must be executed in a fixed order (e.g., drilling must precede painting).
    - **Resource Capacity Constraint:** Each machine is a single, non-divisible resource that can only process one operation at any given time.
    - **Non pre-emption:** Once an operation begins processing on a machine, it must run to completion without interruption. The scheduler must commit to the full processing time.
- **Primary Objectives:** The project focuses on the standard, often conflicting, multi-objective goals that define successful manufacturing operations:
    - Minimizing the **Makespan (Cmax)**: The total time elapsed until the very last job operation is completed (a measure of throughput).
    - Maximizing **Machine Utilization (U)**: The percentage of time machines are actively processing jobs (a measure of resource efficiency).
    - Minimizing **Job Tardiness**: The total time by which jobs are completed after their contractual due dates.

1.2 The Failure of Traditional Methods in Dynamic Environments

In modern industrial environments, the JSSP is exacerbated by dynamic factors—random events that render static schedules obsolete. The most basic, high-speed scheduling methods currently used, **Priority Dispatching Rules (PDRs)** like **FIFO** and **SPT**, cannot cope with this volatility.

- **Priority Dispatching Rules (PDRs) - FIFO and SPT:**
    - These rules are computationally inexpensive, making them the default choice for quick, local decision-making at a single machine's queue. They serve as essential, high-speed baselines for any scheduling system.
    - **Limitation: Local Optimality and Short-Sightedness:** PDRs are fundamentally short-sighted, focusing only on the local queue criterion (e.g., the shortest processing time on the current machine). They cannot account for downstream bottlenecks, the total schedule impact, or the critical due dates of other jobs across the entire production line [9].
    - **Limitation: Rigidity in Dynamic Environments:** Since these rules are static—they always execute the same logic regardless of the shop floor's global state—they lack the adaptability required when disruptions, such as unexpected machine failures or random job arrivals, occur. They merely respond to a sudden event by executing the same predetermined rule, often leading to cascading schedule failures.
    - **Result:** PDRs deliver schedules that are fast and feasible, but they are almost always highly sub-optimal regarding critical global metrics like Cmax and Total Weighted Tardiness (TWT). They fail to achieve the global coordination necessary for effective manufacturing.

The Project's Value Proposition: The Deep Reinforcement Learning (DRL) Advantage

The project proposes leveraging Deep Reinforcement Learning (DRL) to provide a fundamentally different and superior approach to basic PDRs by learning a rapid, adaptive policy network.

- **Problem The Project Solve: Local Optimality vs. Global Objectives**
    - Current Remedy: FIFO or SPT focus only on the local machine queue, ignoring global impact.
    - **The DRL Advantage:** The agent is trained to maximize a global, cumulative reward (TWT + Cmax). By receiving a comprehensive state vector, the agent learns a **global policy** that optimally balances local efficiency (like SPT) with critical global metrics (like due dates). The policy learns **when to break the rule** to anticipate future bottlenecks [4].
- **Problem The Project Solve: Lack of Adaptability to Stochasticity**
    - Current Remedy: Fixed PDRs execute the same logic even if the factory state changes unexpectedly.
    - **The DRL Advantage:** The DRL agent is trained across thousands of stochastic scenarios (simulating machine failures, random order arrivals). This process makes the resulting policy **highly adaptive and robust** to unforeseen events, as it has already learned the optimal conditional response for a wide variety of disrupted states. This gives the scheduling solution its **online and real-time** capability, which is vastly superior to any static rule [2].

1.3 Project Scope and Objectives (Initial Stage)

The primary goal is to design, build, and evaluate a Reinforcement Learning (RL)-based framework for solving the JSSP. This initial stage focuses on proving the DRL concept using stable, achievable methods within the project timeline.

- **Primary Goal:** To design, build, and evaluate an RL-based framework for solving the JSSP, proving that a learned policy can outperform traditional methods in a dynamic environment.
- **Core Objectives:**
    - **Simulation Build:** Create a virtual job shop environment using the Python library simpy. The initial model will be a small, yet complex, 3 machines by 5 jobs instance, which is computationally manageable for initial training [3].
    - **Agent Development:** Implement and train an intelligent DRL agent, strategically utilizing the stable **Proximal Policy Optimization (PPO)** algorithm and a **Multi-Layer Perceptron (MLP)** architecture. PPO is chosen for its stability and suitability for the sequential decision task [4].
    - **Performance Benchmark:** Compare the DRL agent's learned policy against two common, high-speed industry standards: **First-In-First-Out (FIFO)** and **Shortest Processing Time (SPT)** heuristics [3].
    - **Metrics Evaluation:** Measure the effectiveness of the learned policy across the three key industry metrics: **Makespan (Cmax)**, **Machine Utilization (U)**, and **Job Tardiness** [3].

2. Systematic Literature Review (SLR) and Academic Validation

This section provides the comprehensive academic foundation for the project, validating the methodological choices and setting precise performance expectations based on current research findings.

2.1 SLR Methodology and Criteria

The systematic review followed a targeted search strategy across multiple top-tier academic databases (IEEE Xplore, ScienceDirect, ResearchGate, arXiv) to ensure methodological rigor.

2.1.1 Inclusion and Exclusion Criteria

The selection process was designed to ensure the study is both academically rigorous and practically feasible within the project's time constraints.

- **Inclusion Criteria:**
    - Studies were included only if they directly applied Reinforcement Learning (RL) or Deep Reinforcement Learning (DRL) methods specifically to the Job Shop Scheduling Problem (JSSP) or its immediate variants (Flexible JSSP, Dynamic JSSP).
    - Studies must have quantified performance using key industrial metrics (Makespan, tardiness, utilization) against established optimization benchmarks (e.g., Taillard instances).
    - A direct comparative analysis against traditional heuristic rules (FIFO, SPT) or computational meta-heuristics (which are treated as general "traditional methods") was mandatory to establish the DRL method's performance baseline improvement.
- **Exclusion Criteria:**
    - Studies were strictly excluded if they focused on optimization problems unrelated to job sequencing (e.g., vehicle routing, pure flow shop scheduling).
    - The exclusion of specialized architectures, such as those relying on complex graph encoding methods, was a strategic decision to **maximize the robustness and analytical depth** of the foundational DRL policy on the core State-Action-Reward components.
    - Papers primarily covering traditional mathematical programming or purely theoretical analysis without a practical DRL policy implementation were filtered out.

2.2 Core Findings: Performance, Stability, and Feasibility (Detailed Analysis)

The literature confirms that the planned PPO/MLP approach offers the optimal balance of stability and performance for a Master's project scope, validating both the algorithm choice and the expected results.

I. Superior Performance and Real-World Application

- **DRL Outperforms Static Planning in TWT (van Zijl, 2023):**
    - van Zijl (2023) provided a critical, high-impact validation of DRL in a real-world manufacturing context. The core finding was a remarkable **46% improvement** in Total Weighted Tardiness (TWT) when the PPO-based DRL agent was compared to the **static scheduling practices** used by the manufacturer. This statistically confirms the DRL policy's superior ability to manage complex due-date constraints and high-priority jobs compared to non-adaptive methods.
    - Additionally, the DRL agent demonstrated significant reduction in makespan (Cmax) compared to **traditional meta-heuristics** generally, proving its efficiency and solution quality on large, complex problem instances [5].
    - Source: van Zijl, Master's thesis, Eindhoven University of Technology, 2023. Available: https://research.tue.nl/files/340291333/Master_Thesis_Luc_van_Zijl.pdf
- **Operational Resilience to Stochastic Events (Zhang et al., 2022):**
    - The value of DRL lies in its ability to maintain performance in dynamic and stochastic environments, fulfilling the promise of adaptive scheduling for Industry 4.0.
    - Zhang et al. (2022) validated this approach by deploying a smart DRL scheduler in a **physical smart factory testbed**, specifically focusing on autonomous adaptation to unexpected events like machine failures and urgent job insertions.
    - The study showed the DRL agent successfully coped with these disruptions by immediately selecting a new optimal action based on the changed state, demonstrating an operational resilience impossible for brittle, fixed schedules or simple PDRs to match [2].
    - Source: Zhang et al., "Reinforcement learning for online optimization of job-shop scheduling...," Journal of Manufacturing Systems, 2022. Available: https://www.google.com/search?q=https://www.researchgate.net/publication/359190644_Reinforcement_learning_for_online_optimization_of_job-shop_scheduling_in_a-smart_manufacturing_factory

- **The Power of Coordinated Multi-Agent Systems (MARL) (da Cunha and de Madureira, 2023):**
  - While the project starts with a single-agent approach, Multi-Agent RL (MARL) provides clear validation for future scalability by proving that learned policies are superior to heuristics in high-contention environments.
  - da Cunha and de Madureira (2023) demonstrated that MARL systems significantly overperformed both single-agent methods and simple dispatching rules (**FIFO, SPT**) across multiple criteria (like flow time and tardiness), especially when the job shop was under a **severe workload** [6].
  - Source: da Cunha and de Madureira, "A Multi-Agent Reinforcement Learning Approach...," Sustainability, 2023. Available: https://www.mdpi.com/2071-1050/15/10/8262

II. Algorithm Stability and Architecture Choice

- **PPO for Superior Stability and Lower Variance (Wang et al., 2024):**
    - The strategic choice of **Proximal Policy Optimization (PPO)** is key for project stability, as opposed to the Deep Q-Network (DQN).
    - Comparative studies establish that PPO, a policy gradient method, is generally superior to DQN, a value-based method, for complex optimization tasks due to its inherent stability.
    - PPO is highly valued for its stability and lower variance during training, which directly mitigates the risk of catastrophic policy divergence and requires less intensive hyperparameter search—a critical factor for a time-constrained Master's project [7].
    - Source: Wang et al., "Comparative Study of Reinforcement Learning Performance Based on PPO and DQN Algorithms," ResearchGate, 2024. Available: https://www.google.com/search?q=https://www.researchgate.net/publication/382022513_Comparative_Study_of_Reinforcement_Learning_Performance_Based_on_PPO_and_DQN_Algorithms
- **MLP Feasibility and Performance (Tassel et al., 2021):**
    - The reliance on a simple **Multi-Layer Perceptron (MLP)** network is justified because it achieves high performance without overcomplicating the architecture.
    - Tassel et al. (2021) demonstrated that an MLP architecture, when correctly paired with strategic reward engineering, can achieve high performance on JSSP benchmarks comparable to much more complex, specialized models.
    - This confirms that focusing on robust MDP design and accurate feature engineering is the right strategy for securing project feasibility within the aggressive two-month timeline [4].
    - Source: Tassel et al., "A Reinforcement Learning Environment For Job-Shop Scheduling," arXiv preprint, 2021. Available: https://arxiv.org/abs/2104.03760

III. The Criticality of Composite Reward Engineering

- **Overcoming Sparse Reward with Dense Cmax Penalty (Tassel et al., 2021):**
    - The major challenge of **sparse reward** (feedback only at the end) is overcome by using a **dense reward function**.
    - Tassel et al. (2021) showed the solution is a dense reward that instantly penalizes the agent based on the increase in the partial makespan (Change in Cmax) at every decision step.
    - This technique is indispensable because it effectively guides the policy towards the global goal of minimizing total schedule length by enforcing necessary local discipline [4].
    - Source: Tassel et al., "A Reinforcement Learning Environment For Job-Shop Scheduling," arXiv preprint, 2021. Available: https://arxiv.org/abs/2104.03760
- **Immediate Penalty for Tardiness Optimization (Fathipoor et al., 2023):**
    - To effectively minimize due-date violations, the agent must receive an immediate signal concerning tardiness.
    - Fathipoor et al. (2023) validated that applying a **negative reward proportional to the tardiness** incurred upon job completion is the most effective way to train a due-date-focused policy [8].
    - The project will integrate this as a weighted penalty for Total Weighted Tardiness (TWT), reflecting job priorities [8].
    - Source: Fathipoor et al., "Deep Reinforcement Learning-Based Scheduler...," MDPI, 2023. Available: https://www.google.com/search?q=https://www.mdpi.com/2071-1050/15/4/760
- **Long-Term Potential with Advanced Metrics (Estimated Tardiness Study, 2025):**
    - Research shows that even more sophisticated metrics, like the **Estimated Tardiness (ETD)** metric, can be successfully integrated into the reward function, yielding significant performance gains (up to 27% over conventional approaches).
    - This confirms that the project's research path—focusing on optimizing the reward signal—is the correct avenue for maximizing performance, offering potential avenues for future work [10].
    - Source: Estimated Tardiness-Based Reinforcement Learning Solution to Repeatable Job-Shop Scheduling Problems, MDPI, 2025. Available: https://www.mdpi.com/2227-9717/13/1/62

3. DRL Framework Definition and Viability

The project design ensures that the **Markov Decision Process (MDP)** components are simple, powerful, and achievable for the pilot 3 x 5 JSSP within the given timeline.

3.1 State Space (S): Multi-Layer Perceptron (MLP) Input

- The state S must be a **fixed-length, feature-engineered vector** suitable for input into the Multi-Layer Perceptron (MLP) network. This design simplifies the processing phase while providing necessary environmental context.
- **Machine Status (Feasibility: High)**:
  - Includes the binary status (Busy or Idle) of all 3 machines.
  - Includes the time remaining until the current operation on each busy machine is completed.
  - This directly enforces resource constraints and allows the agent to gauge immediate resource availability and commit to non-preemptive operations.
- **Resource Load (Feasibility: High)**:
  - Includes the current queue length and the total remaining work waiting for the 3 machines.
  - Crucially, includes the explicit **Bottleneck utilization ratio** [6].
  - This set of features guides the agent toward a necessary global perspective on congestion, preventing locally optimal, globally disastrous decisions.
- **Job Progress (Feasibility: High)**:
  - Includes the index of the next operation for each of the 5 jobs.
  - Includes the required processing time and the priority weight (wj) for that next operation.
  - This defines the set of ready operations and provides the necessary input for the agent to emulate and select scheduling rules.
- **Global Time (Feasibility: High)**:
  - Includes the current simulation time (t) and the current makespan (Cmax) of the partial schedule.
  - Includes the estimated tardiness status for jobs nearing their due date.
  - This provides critical temporal context for the reward calculation and decision-making under time constraints.

3.2 Action Space (A): Rule Selection (Meta-Rule)

- The action space is defined using the **Meta-Rule Approach**, which is the simplest and most stable method for this scope.
- This strategy prevents the action space from exploding exponentially, which is essential for a short timeline [9].
- The agent's policy will select one priority rule from the discrete set of actions at every scheduling decision point.
- The set of actions a sub t includes:
    - **FIFO** (First-In-First-Out)
    - **SPT** (Shortest Processing Time)
    - **EDD** (Earliest Due Date)
    - **LPT** (Longest Processing Time)

3.3 Composite Reward Function (R): The Core Focus

- The success of the project hinges on iterating and tuning the **Composite Reward Function**, which balances the conflicting objectives of Makespan, Utilization, and Tardiness.
- The overall instantaneous reward r sub t is calculated as the sum of three weighted components:
  - The composite reward rt is calculated as R Completion multiplied by lambda C plus R Tardiness multiplied by lambda T plus R Utilization multiplied by lambda U.
- **R Completion (The Dense Penalty):**
  - This is the negative penalty applied instantly when the schedule decision causes the Change in Cmax to be greater than zero.
  - It guides the agent away from locally greedy decisions that delay the overall project completion time.
- **R Tardiness (The Weighted Penalty):**
  - This is a strong negative weighted penalty applied only at the moment a job is completed, if its completion time (Cj) is greater than its due date (dj).
  - The weighting (based on job priority) ensures the agent prioritizes critical or urgent jobs.
- **R Utilization (The Positive Incentive):**
  - This is a small positive reward given to encourage the agent to keep machines busy and reduce idle time. This ensures resource capital is maximized.
- **lambdaC, lambdaT, lambdaU (Hyperparameters):**
  - These weighting factors are the primary hyperparameters to be tuned during the experimentation phase.
  - They control the balance of the policy, allowing the project to prioritize Cmax (low lambdaT) or TWT (high lambdaT) based on the client's needs.

4. Implementation and Evaluation Strategy

4.1 Technology Stack and Simulation Modeling

- **Language & Framework:** The project will utilize Python 3.9+ with the discrete-event simulation library simpy to model the shop floor environment. The DRL agent will be implemented using the Stable-Baselines3 library, which provides a robust PPO implementation [3].
- **Simulation Environment:** The simpy model will handle the scheduling queue and resource management for the 3 x 5 JSSP. Crucially, it will utilize discrete events to pause the clock and request an action from the PPO policy at every decision point (e.g., machine completion).

4.2 Training Strategy: Progressive Curriculum

To achieve stable and generalizable results quickly, the agent will follow a **Progressive Curriculum** training strategy [5], starting simple and incrementally adding complexity.

1. **Phase 1 (Basic Training):**
   - Focus: Train on small, static JSSP instances (e.g., 3x3, 5x5) where due dates are irrelevant.
   - Reward: Use only the R Completion component to establish a baseline for maximizing throughput.
2. **Phase 2 (Complexity Integration):**
   - Focus: Introduce the full Composite Reward Function (including R Tardiness and R Utilization).
   - Goal: Teach the agent the complex trade-offs required to balance due dates with makespan, moving from local efficiency to global optimization.
3. **Phase 3 (Generalization/Robustness):**
   - Focus: Introduce stochasticity (random job arrivals, variable processing times) into the environment.
   - Goal: Prepare the agent's policy to be robust and adaptive for a dynamic environment, testing its resilience.

4.3 Evaluation Methodology

The DRL agent will be evaluated on a held-out test set of larger problem instances (e.g., 10 x 10) to rigorously test generalization beyond the training set.

1. **Makespan (Cmax):** Must significantly outperform the baseline FIFO rule and be competitive with or better than SPT.
2. **Total Weighted Tardiness (TWT):** Must significantly outperform both the SPT and FIFO heuristics, as they fail to manage due dates effectively.
3. **Machine Utilization (U):** Must achieve utilization rates comparable to or better than a greedy SPT rule, especially for bottleneck machines.
4. **Computational Speed:** Action time must be near-instantaneous (e.g., less than 1 millisecond) to qualify as a real-time solution.

5. Conclusion and Project Viability

This initial bibliography report validates that **Deep Reinforcement Learning (DRL)** provides the most viable, effective, and modern method for solving the JSSP in dynamic environments. The project's structure, which relies on the stability of the **Proximal Policy Optimization (PPO)** algorithm and a carefully designed **Multi-Layer Perceptron (MLP)** architecture, is strategically positioned to deliver quantifiable, state-of-the-art results [5]. The ultimate success hinges on the rigorous implementation and tuning of the **Composite Reward Function** and **State Feature Engineering** to enable the DRL agent to learn complex, non-obvious policies that overcome the severe limitations of conventional scheduling heuristics.

The project is structured to succeed by prioritizing robustness and fundamental DRL principles, confirming that the goals are manageable and the expected outcomes are highly impactful for the field of intelligent manufacturing.

6. References

1. Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Complexity of vehicle routing and scheduling problems. *Networks*, 9(2), 117-124. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230090204

2. Zhang et al. (2022). Reinforcement learning for online optimization of job-shop scheduling in a smart manufacturing factory. *Journal of Manufacturing Systems*, 62, 792-803. URL: https://www.google.com/search?q=https://www.researchgate.net/publication/359190644_Reinforcement_learning_for_online_optimization_of_job-shop_scheduling_in_a-smart_manufacturing_factory

3. Project Slides (Internal Document). Citing the core goals and methodology (3x5 instance, FIFO/SPT). (Project Documentation/Slides)

4. Tassel, P., Gebser, M., & Schekotihin, K. (2021). A Reinforcement Learning Environment for Job-Shop Scheduling. *arXiv preprint arXiv:2104.03760*. URL: https://arxiv.org/abs/2104.03760

5. van Zijl, L. (2023). Solving a Job-Shop Scheduling Problem through Deep Reinforcement Learning (Master's thesis). Eindhoven University of Technology. URL: https://research.tue.nl/files/340291333/Master_Thesis_Luc_van_Zijl.pdf

6. da Cunha, A. S. C., & de Madureira, J. A. P. P. (2023). A Multi-Agent Reinforcement Learning Approach to the Dynamic Job Shop Scheduling Problem. *Sustainability*, 15(10), 8262. URL: https://www.mdpi.com/2071-1050/15/10/8262

7. Wang et al. (2024). Comparative Study of Reinforcement Learning Performance Based on PPO and DQN Algorithms. *ResearchGate*, 2024. URL: https://www.google.com/search?q=https://www.researchgate.net/publication/382022513_Comparative_Study_of_Reinforcement_Learning_Performance_Based_on_PPO_and_DQN_Algorithms

8. Fathipoor et al. (2023). Deep Reinforcement Learning-Based Scheduler on Parallel Dedicated Machine Scheduling Problem towards Minimizing Total Tardiness. *MDPI*, 15(4), 760. URL: https://www.google.com/search?q=https://www.mdpi.com/2071-1050/15/4/760

9. Zhou, X., Ji, G., & Sun, Y. (2018). Review of dynamic job shop scheduling problems. *Journal of Intelligent Manufacturing*, 29(1), 163-181. URL: https://www.google.com/search?q=https://link.springer.com/article/10.1007/s10845-017-1339-1

10. Gacem, M., Bouzouia, B., & Benamar, A. (2025). Estimated Tardiness-Based Reinforcement Learning Solution to Repeatable Job-Shop Scheduling Problems. *MDPI*, 13(1), 62. URL: https://www.mdpi.com/2227-9717/13/1/62