GRAPH-ENHANCED DEEP REINFORCEMENT LEARNING FOR MULTI-OBJECTIVE UNRELATED PARALLEL MACHINE SCHEDULING

Bulent Soykan[1], Sean Mondesire[2], Ghaith Rabadi[2], and Grace Bochenek[2]

[1]Institute for Simulation and Training, University of Central Florida, Orlando, USA
[2]School of Modeling, Simulation, and Training, University of Central Florida, Orlando, USA

ABSTRACT

The Unrelated Parallel Machine Scheduling Problem (UPMSP) with release dates, setups, and eligibility constraints presents a significant multi-objective challenge. Traditional methods struggle to balance minimizing Total Weighted Tardiness (TWT) and Total Setup Time (TST). This paper proposes a Deep Reinforcement Learning framework using Proximal Policy Optimization (PPO) and a Graph Neural Network (GNN). The GNN effectively represents the complex state of jobs, machines, and setups, allowing the PPO agent to learn a direct scheduling policy. Guided by a multi-objective reward function, the agent simultaneously minimizes TWT and TST. Experimental results on benchmark instances demonstrate that our PPO-GNN agent significantly outperforms a standard dispatching rule and a metaheuristic, achieving a superior trade-off between both objectives. This provides a robust and scalable solution for complex manufacturing scheduling.

1 INTRODUCTION

Advanced manufacturing environments operate under intense pressure to improve efficiency, reduce operational costs, and maintain responsiveness to dynamic market demands (Davis et al. 2012). The implementation of efficient production scheduling is crucial for achieving these goals. It involves finding the best way to allocate constrained resources, such as machines and personnel, to competing tasks or jobs over a specific timeframe. Optimal scheduling decisions directly impact key performance indicators (KPIs), including production throughput, resource utilization, adherence to delivery deadlines, and overall system productivity. Consequently, developing adaptive scheduling methodologies is important for maintaining competitiveness in today's globalized industrial landscape (Mourtzis 2022).

A fundamental class of scheduling problems encountered in various industrial settings is the Parallel Machine Scheduling Problem (PMSP). In this context, a set of jobs must be processed on a set of available machines. This paper focuses on a particularly challenging variant known as the Unrelated Parallel Machine Scheduling Problem (UPMSP). In the UPMSP, the processing time required for a specific job depends not only on the job itself but also distinctly on the machine assigned to process it. This scenario accurately reflects real-world situations where machines, even if performing similar functions, may differ in age, capability, or specialization. Such problems are prevalent in diverse sectors, including semiconductor manufacturing (Bitar et al. 2016), textile production (Li et al. 2021), where heterogeneity among processing units is common.

Real-world UPMSP applications are further complicated by several interacting constraints addressed in this research. These include job-specific release dates, machine eligibility restrictions, and setup times that depend on both the sequence of jobs and the specific machine. Handling these complexities simultaneously significantly increases the difficulty of finding optimal or near-optimal schedules. In addition, effective scheduling must often balance conflicting objectives, such as minimizing total weighted tardiness (TWT) to meet deadlines and minimizing total setup time (TST) for operational efficiency. The inherent conflict—where prioritizing one objective negatively impacts the other—makes finding high-performing solutions challenging. Given the NP-hard nature of this multi-objective UPMSP (Fleszar and Hindi 2018), traditional approaches face limitations in effectively managing this trade-off.

In this paper, we propose a novel method based on Deep Reinforcement Learning (DRL) in order to address the limitations of existing methods and tackle the complexities of the multi-objective UPMSP. DRL agents learn decision-making policies through interaction with an environment, making them well-suited for sequential decision problems like scheduling. We specifically use Graph Neural Networks (GNNs) to handle the intricate structure and variable nature of the scheduling state. GNNs can effectively process graph-based representations, capturing the relationships between jobs, machines, eligibility constraints, and setup dependencies. Our framework utilizes a heterogeneous GNN representation designed to encode the rich information inherent in the UPMSP state, including distinct node types for jobs, machines, and potential setup configurations, connected by meaningful edge types. This GNN acts as a powerful feature extractor, feeding state information to a DRL agent trained using the Proximal Policy Optimization (PPO) algorithm. PPO is chosen for its stability and proven effectiveness in complex control tasks, enabling the agent to learn a policy that directly selects which job to assign to which machine at each decision step.

The main contributions of this research are threefold. First, we propose and evaluate the application of the PPO algorithm for learning a *direct scheduling policy* for the complex UPMSP variant, moving beyond heuristic selection or simpler scheduling scenarios often addressed by DRL. Second, we design and implement a *heterogeneous GNN state representation* tailored to the UPMSP, explicitly modeling jobs, machines, setup states, and their complex interrelations to facilitate more effective learning. Third, we develop and analyze an *explicit multi-objective reward function* within the PPO framework, specifically designed to guide the agent in learning policies that effectively balance the conflicting objectives.

The remainder of this paper is organized as follows. Section 2 provides a review of relevant literature on UPMSP, DRL in scheduling, and GNN applications. Section 3 formally defines the target UPMSP and its objectives. Section 4 details the proposed PPO-GNN framework, including the MDP formulation, GNN architecture, and reward function design. Section 5 describes the experimental setup, instance generation, baseline methods, and evaluation metrics. Section 6 presents and discusses the computational results and comparative analysis. Finally, Section 7 concludes the paper and outlines directions for future research.

2 RELATED WORK

2.1 Parallel Machine Scheduling Problem (PMSP)

The PMSP encompasses a broad category of scheduling challenges where multiple jobs need to be processed using a set of available machines operating in parallel. Research in PMSP branches into several classifications based on machine characteristics: identical machines (where processing times are job-dependent only), uniform machines (where machines have different speeds but maintain relative processing time proportions), and unrelated machines (where processing time depends arbitrarily on both the job and the machine) (Mokotoff 2001). This work focuses on the *UPMSP*, which poses significant combinatorial challenges due to the machine-dependent nature of processing times.

The complexity of UPMSP grows significantly when incorporating realistic industrial constraints. A substantial body of literature addresses UPMSP variants that include sequence-dependent setup times, where the time to prepare a machine depends on the previous job processed (Vallada and Ruiz 2011), machine-dependent setup times inherent to the unrelated nature (Avalos-Rosales et al. 2015), distinct release dates for jobs (Li et al. 2022), and machine eligibility restrictions limiting which machines can process certain jobs (Afzalirad and Shafipour 2018). Common objectives studied include minimizing the makespan ($C_{\max}$), total completion time ($\sum C_j$), total tardiness ($\sum T_j$), and TWT ($\sum w_j T_j$) (Mokotoff 2001). Recent research also increasingly considers setup time minimization as a critical objective due to its impact on resource utilization and efficiency. This paper addresses the UPMSP with release dates, sequence- and machine-dependent setup times, and machine eligibility, aiming to simultaneously minimize TWT and TST.

Due to the inherent NP-hard computational complexity of most UPMSP variants, *heuristics and dispatching rules* are widely employed, particularly in dynamic environments requiring rapid decision-making (Rabadi et al. 2006). These methods prioritize jobs waiting for processing based on specific rules, such as Shortest Processing Time (SPT), Earliest Due Date (EDD), or Critical Ratio (CR). For more complex scenarios involving setup times and tardiness objectives, composite rules like the Apparent Tardiness Cost

(ATC) rule and its derivatives, such as Apparent Tardiness Cost with Setups (ATCS), have been developed (Lee et al. 1997). Variations like Apparent Tardiness Cost with Setups and Ready Times (ATCSR) explicitly incorporate release dates and setup times (Lin and Hsieh 2014). While computationally efficient, these rules typically make greedy, myopic decisions based on local information. Their performance can degrade significantly when faced with the intricate interactions present in UPMSP with sequence-dependent setups and multiple objectives, often leading to suboptimal solutions (Rabadi et al. 2006).

In order to overcome the limitations of simple heuristics, various metaheuristic and exact approaches have been proposed. *Metaheuristics* such as Genetic Algorithms (GA) (Vallada and Ruiz 2011), Tabu Search (TS) (Lee et al. 2013), Simulated Annealing (SA) (Kim et al. 2022), or Ant Colony Optimization (ACO) (Afzalirad and Rezaeian 2016) explore the solution space more broadly and can yield higher-quality solutions compared to simple heuristics. However, they typically require considerable problem-specific parameter tuning, can be computationally intensive, provide no guarantee of optimality, and their iterative nature may limit their applicability in highly dynamic or real-time decision-making scenarios (Moser et al. 2022). *Exact methods* such as Mixed-Integer Programming (MIP) (Saraç and Tutumlu 2022), Constraint Programming (CP) (Gedik et al. 2018), or specialized Dynamic Programming (DP) algorithms can guarantee optimal solutions (Pfund et al. 2004). Unfortunately, due to the combinatorial complexity of the problem, these methods suffer from the "curse of dimensionality". As a result, they are typically only practical to compute for very small problem sizes, making them unsuitable for application at realistic industrial scales (Pei et al. 2020). These limitations highlight the need for alternative approaches that can effectively handle the scale and complexity of the target problem while learning to navigate the intricate trade-offs between multiple objectives.

2.2 Deep Reinforcement Learning in Scheduling

Recently, DRL has become known as an effective technique for tackling complex sequential decision-making problems, including combinatorial optimization problems such as scheduling (Mazyavkina et al. 2021). Unlike traditional optimization methods that often search through a vast solution space, DRL aims to learn a *policy*—a mapping from the current system state to an optimal or near-optimal action—through trial-and-error interactions with an environment (often a simulation model) (Soykan and Rabadi 2023). The integration of deep neural networks allows DRL agents to handle high-dimensional state spaces and learn complex patterns, making them suitable for capturing the dynamics of intricate scheduling environments (Cunha et al. 2020).

DRL has been successfully applied to various classical scheduling problems. For the Job Shop Scheduling Problem (JSSP), significant research has focused on learning dispatching policies (Soykan and Rabadi 2024). Early work often involved using DRL to select from a predefined set of dispatching rules (Zhang et al. 2020). More recent approaches utilize advanced state representations, such as disjunctive graphs processed by GNNs, enabling agents to learn end-to-end policies that directly select the next operation to schedule (Park et al. 2021). Similar advancements have been made for the Flow Shop Scheduling Problem (FSPP), including permutation flow shops (Cho et al. 2022) and flexible flow shops (Kwon et al. 2021), often employing sequence-to-sequence models or GNNs to handle job dependencies and routing decisions. Objectives commonly tackled include makespan minimization and tardiness reduction. These studies demonstrate the potential of DRL to learn high-quality, adaptive scheduling strategies directly from simulated experience.

PPO (Schulman et al. 2017) is a state-of-the-art policy gradient DRL algorithm known for its stability, reliability, and strong practical performance across a wide range of challenging tasks, including continuous control and combinatorial optimization problems. Compared to simpler policy gradient methods like REINFORCE or actor-critic methods like A2C/A3C, PPO incorporates a clipping mechanism in the objective function or an adaptive Kullback-Leibler (KL) penalty to constrain policy updates, preventing excessively large changes that can destabilize training. This makes it attractive for scheduling problems where the state-action space is large and finding a good policy requires careful exploration. Its balance

between sample efficiency, ease of implementation, and robust performance motivates its selection as the core learning algorithm in our proposed framework for the multi-objective UPMSP.

2.3 Graph Neural Networks in Combinatorial Optimization

GNNs have emerged as a highly effective deep learning architecture for combinatorial optimization problems, many of which naturally lend themselves to graph representations where nodes signify entities and edges capture relationships or dependencies (Huang et al. 2019). GNNs utilize message-passing mechanisms to iteratively aggregate information from node neighborhoods, learning powerful embeddings that encode both node features and the graph's topology, making them adept at handling variable-sized inputs common in combinatorial optimization (Chung et al. 2025). Their success, often in conjunction with DRL, is evident across various combinatorial optimization domains, including routing problems, where GNN encoders generate representations for sequence-constructing decoders (Kool, Wouter and Van Hoof, Herke and Welling, Max 2018), and other fundamental problems like SAT and MaxCut (Huang et al. 2019). Given that scheduling problems can frequently be modeled as graphs (e.g., disjunctive graphs for JSSP), GNNs are increasingly central to modern DRL-based scheduling approaches for JSSP (Zhang et al. 2020) and FSPP (Kwon et al. 2021). For PMSP, Cho et al. specifically utilized a GNN to represent the state (including machine-job pair nodes and machine nodes) for solving the UPMSP with setups, release dates, and eligibility, focusing on minimizing TWT using the REINFORCE algorithm (Cho et al.). Norman et al. also employed a GNN in a scheduling context, but used it primarily as a feature extractor to inform a DRL agent that tuned parameters within a predefined dispatching rule, rather than performing direct scheduling (Norman et al. 2024). These works underscore the GNN's ability to effectively encode the complex relational information present in scheduling environments.

While previous research has demonstrated the potential of DRL and GNNs for various scheduling problems, several gaps remain, particularly concerning the complex, multi-objective UPMSP variant addressed here. Existing DRL approaches for UPMSP often focus on single objectives like TWT (Cho et al.), select from predefined heuristics rather than learning direct policies (Nam et al. 2024), or tune parameters of existing rules (Norman et al. 2024). Also, while GNNs have been used, the design of graph representations tailored to capture the information needed to effectively balance TWT minimization against TST minimization in UPMSP requires further investigation.

This paper endeavors to bridge these gaps by introducing a novel framework that uniquely combines several elements: *(i)* we employ the robust *PPO* algorithm to learn a *direct scheduling policy* for the UPMSP, offering potential stability advantages over simpler policy gradient methods used in some prior work, *(ii)* we tackle the *multi-objective nature* of the problem explicitly, designing a reward function and learning process aimed at simultaneously minimizing both TWT and TST, *(iii)* we introduce an *enhanced heterogeneous GNN structure* specifically designed for this multi-objective UPMSP variant, incorporating distinct nodes for jobs, machines, and potential setups, along with carefully defined edge types to capture eligibility, processing times, and setup relationships (including transitions). By integrating these components, our work seeks to advance the state-of-the-art in applying DRL to solve large-scale, complex, multi-objective scheduling problems encountered in real-world manufacturing.

3 PROBLEM FORMULATION

This section formally defines the UPMSP variant addressed in this paper, including its parameters, decision variables, constraints, and objectives.

Formal Definition   We consider a set of independent jobs $J = \{J_1, J_2, \ldots, J_n\}$ that need to be processed on a set of unrelated parallel machines $M = \{M_1, M_2, \ldots, M_m\}$. The problem parameters are defined as follows:

- p_{jk}: The processing time of job $J_j \in J$ when processed on machine $M_k \in M$. The machines are unrelated, meaning p_{jk} can vary arbitrarily for different machines k and jobs j.

- r_j: The release date (or ready time) of job $J_j \in J$, representing the earliest time at which processing of job J_j can begin.
- d_j: The due date of job $J_j \in J$, representing the target completion time for the job.
- w_j: The weight or importance of job $J_j \in J$, used in calculating the weighted tardiness objective.
- s_{ijk}: The sequence-dependent and machine-dependent setup time required on machine $M_k \in M$ to switch from processing job $J_i \in J \cup \{J_0\}$ to processing job $J_j \in J$. Here, J_0 represents the initial idle state of the machine, and s_{0jk} denotes the initial setup time if job J_j is the first job processed on machine M_k.
- $M_j \subseteq M$: The set of eligible machines on which job $J_j \in J$ can be processed. If $M_k \notin M_j$, job J_j cannot be assigned to machine M_k.

Decision Variables   The primary decisions involved in solving this UPMSP instance are (These decisions implicitly determine the start time (S_{jk}), completion time (C_{jk}), and ultimately the overall schedule performance based on the defined objectives. Let C_j denote the completion time of job J_j on the machine it is assigned to.):

- *Job Assignment:* Determining which eligible machine $M_k \in M_j$ will process each job $J_j \in J$.
- *Job Sequencing:* Determining the order (sequence) in which the jobs assigned to each specific machine $M_k \in M$ will be processed.

Constraints   The scheduling decisions must adhere to the following constraints:

- *Release Date:* Processing for job J_j on its assigned machine M_k cannot start before its release date: $S_{jk} \geq r_j$.
- *Machine Eligibility:* Each job J_j must be assigned to a machine M_k from its eligible set: $M_k \in M_j$.
- *Non-preemption:* Once the processing of a job starts on a machine, it cannot be interrupted until it is completed.
- *Machine Capacity:* Each machine M_k can process at most one job at any given time. If job J_j immediately follows job J_i in the sequence on machine M_k, then the start time of J_j must be greater than or equal to the completion time of J_i plus the required setup time: $S_{jk} \geq C_{ik} + s_{ijk}$.

Objective Functions   The goal is to find a feasible schedule that simultaneously optimizes two conflicting objectives:

- *Minimize Total Weighted Tardiness (TWT)* Tardiness for job J_j is defined as $T_j = \max(0, C_j - d_j)$. The objective is to minimize the sum of weighted tardiness over all jobs:

$$\text{Minimize } TWT = \sum_{j=1}^{n} w_j T_j = \sum_{j=1}^{n} w_j \max(0, C_j - d_j) \tag{1}$$

- *Minimize Total Setup Time (TST)* Let σ_k be the sequence of jobs assigned to machine M_k, represented as $(J_{k(1)}, J_{k(2)}, \ldots, J_{k(n_k)})$, where n_k is the number of jobs assigned to M_k. The TST is the sum of all setup times incurred across all machines:

$$\text{Minimize } TST = \sum_{k=1}^{m} \left(s_{0k(1)k} + \sum_{l=1}^{n_k-1} s_{k(l)k(l+1)k} \right) \tag{2}$$

where $s_{0k(1)k}$ is the setup time for the first job $J_{k(1)}$ on machine M_k.

4   METHODOLOGY: PPO-GNN FOR MULTI-OBJECTIVE UPMSP

This section details the proposed DRL framework for solving the multi-objective UPMSP. We use PPO combined with a tailored GNN architecture.

The core of our approach is an interaction loop between a DRL agent and a scheduling environment, implemented as a discrete-event simulator. At specific decision points (e.g., when a machine becomes idle), the environment provides the current state s_t to the agent. The agent, using its learned policy π, selects an action a_t (assigning a job to a machine or waiting). The environment executes this action, advancing the simulation time and determining the next state s_{t+1} and an immediate reward r_t. The agent collects these experiences (s_t, a_t, r_t, s_{t+1}) to update its policy and value function parameters via the PPO algorithm, aiming to maximize the cumulative discounted reward over time. Figure 1 illustrates this interaction loop.
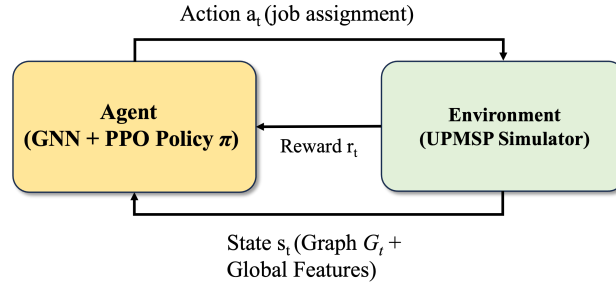
Action a_t (job assignment)

Agent (GNN + PPO Policy π)   Reward r_t Environment (UPMSP Simulator)

State s_t (Graph G_t + Global Features)

Figure 1: Overview of the DRL framework for UPMSP scheduling.

4.1 Markov Decision Process (MDP) Formulation

We formulate the UPMSP as a Markov Decision Process (MDP), defined by the tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \gamma)$, where $\mathscr{S}$ is the state space, $\mathscr{A}$ is the action space, $\mathscr{P}$ is the state transition probability function (implicitly defined by the simulator), $\mathscr{R}$ is the reward function, and γ is the discount factor.

State Representation ($s_t \in \mathscr{S}$)   Capturing the complex state of the UPMSP environment effectively is important for the DRL agent. We propose a composite state representation s_t at decision time step t, consisting of a global feature vector and a heterogeneous graph:

Global Feature Vector: Provides a summary of the overall system status, including features: (i) current number of jobs waiting in the queue (WIP). (ii) number of jobs expected to arrive within a near-future time horizon. (iii) aggregate performance metrics (e.g., number of currently tardy jobs, average flow time of completed jobs, TST incurred so far). (iv) overall machine status (e.g., number of idle machines, number of machines under setup, number of machines undergoing maintenance if applicable). (v) current simulation time and time elapsed.

Heterogeneous Graph ($\mathscr{G}_t$): Represents the detailed relationships between entities in the scheduling environment. It contains different types of nodes and edges:

Node Types & Features: (i) Job Nodes: One node for each job J_j currently waiting or arriving soon. Features include its weight (w_j), remaining processing time estimate (average p_{jk} over eligible machines, or specific p_{jk} if assignment is imminent), due date (d_j), release date (r_j), and potentially features indicating its setup requirements. *(ii) Machine Nodes:* One node for each machine M_k. Features include its current status (idle, busy, setup, maintenance), time until available, identifier of the last job processed (J_i), current setup state (e.g., identifier corresponding to J_i), and time elapsed in the current state/setup. *(iii) Setup Nodes:* Nodes representing distinct and relevant setup configurations (e.g., characterized by job family, required tooling, or energy level). Features identify the setup type. These nodes allow summarizing information about jobs requiring a specific setup, even if no machine currently has it.

Edge Types & Features: Edges represent relationships and potential actions. Features encode relevant costs and properties: *(i) Job-Machine Edges:* Connect job node J_j to machine node M_k if $M_k \in M_j$. Features include processing time p_{jk}, estimated setup time $s_{i'jk}$ (where $J_{i'}$ is the last job on M_k), eligibility flag (binary). *(ii) Machine-Setup Edges:* Connect machine node M_k to the setup node representing its current setup state. Features indicate time spent in this setup. *(iii) Job-Setup Edges:* Connect job node J_j to the setup node representing the setup required before processing J_j. Features indicate job priority (w_j). *(iv) Setup-Machine Edges:* Connect setup node S to machine node M_k. Features represent the estimated

time/cost to transition machine M_k from its current setup to setup S, indicating 'transition ease' (e.g., small setup time if only a minor change is needed).

Action Space ($a_t \in \mathscr{A}$)   The action space consists of potential scheduling decisions. The primary action is the selection of a feasible job-machine pair (J_j, M_k) to be scheduled next. The agent's policy $\pi(a_t|s_t)$ provides a probability distribution over the feasible actions.

- A pair (J_j, M_k) is feasible if job J_j is available (i.e., r_j is met or will be met by the time the machine is ready), machine M_k is eligible for J_j ($M_k \in M_j$), and M_k is currently idle or will become idle.
- The agent's output (logits from the policy network) corresponds to all possible job-machine pairs. An action masking mechanism is applied to ensure only feasible actions have non-zero selection probability. Infeasible actions (e.g., assigning to an ineligible machine, assigning a job that is not released) are masked out (given zero or highly negative log-probability).

Reward Function ($r_t = \mathscr{R}(s_t, a_t, s_{t+1})$)   Designing the reward function r_t is critical for steering the DRL agent to effectively balance the conflicting objectives of minimizing TWT and TST. We explored several structures to achieve this balance, including dense, event-based rewards calculated after each assignment $a_t = (J_j, M_k)$ (e.g., $r_t = -\alpha \cdot \Delta TWT_j - \beta \cdot s_{i'jk}$), hybrid approaches combining immediate penalties for setup time with sparser rewards reflecting overall episode performance on both TWT and TST, and constraint-focused designs incorporating penalties for auxiliary KPI violations. The specific formulation and the relative weighting of components (via factors like α, β) significantly influence the learned policy's trade-off behavior and require careful empirical tuning to achieve the desired multi-objective performance.

State Transition ($\mathscr{P}(s_{t+1}|s_t, a_t)$)   The state transition dynamics are implicitly defined by the discrete-event simulation environment. Given the current state s_t and the agent's chosen action a_t, the simulator advances time, updates machine statuses, calculates job completion times (including processing and setup), handles new job arrivals based on their release dates, and determines the resulting next state s_{t+1}. The DRL agent interacts with this simulator without needing an explicit model of the transition probabilities.

5 EXPERIMENTAL SETUP

This section details the experimental methodology employed to evaluate the performance of our PPO-GNN framework. We describe the generation of problem instances, the baseline methods selected for comparison, the configuration of the DRL agent's training process, and the metrics used for performance evaluation.

Instance Generation   We generated a diverse set of UPMSP instances following methodologies adapted from established literature, particularly (Cho et al.). Problem instances were constructed by systematically varying parameters known to influence scheduling complexity. Specifically, we considered job set sizes $n \in \{20, 50, 100\}$ and machine set sizes $m \in \{5, 10, 15\}$. Processing times p_{jk} for each job-machine pair were independently drawn from a discrete uniform distribution $DU(1, 100)$, guaranteeing machine unrelatedness. An overall average processing time $\bar{p}$ was calculated across all potential job-machine pairings for reference in generating other parameters. Job release dates r_j were sampled from $DU(0, \lambda \cdot \bar{p})$, where the arrival intensity factor was set to $\lambda = 0.5$. Due dates d_j were determined using standard tightness (τ) and range (R) parameters, relative to the job's release date and its average processing time across eligible machines ($\bar{p}_j$), according to $d_j \sim DU(r_j + \bar{p}_j(1 - \tau - R/2), r_j + \bar{p}_j(1 - \tau + R/2))$. We utilized combinations of $\tau \in \{0.2, 0.4, 0.6\}$ and $R \in \{0.2, 0.6, 1.0\}$. Job importance weights w_j were sampled from $DU(1, 10)$. Sequence-dependent and machine-dependent setup times s_{ijk} were drawn from $DU(0, \beta \cdot \bar{p})$ using setup ratios $\beta \in \{0.1, 0.25\}$, applied randomly without imposing specific job family structures. Machine eligibility was controlled via a density parameter $\delta \in \{0.75, 1.0\}$; each job J_j was randomly assigned to $\lceil \delta \cdot m \rceil$ machines from M, ensuring J_j was eligible for at least one machine. For each distinct combination of parameters ($n, m, \tau, R, \beta, \delta$), 50 unique instances were generated to ensure robust statistical analysis.

Baseline Methods   In order to assess the performance of our PPO-GNN approach, we compared it against representative algorithms from different scheduling methodology classes:

(i) Dispatching Rule We implemented a well-regarded composite dispatching rule designed for parallel machine environments with setups and due dates: the **ATCSR_Rm** rule (Lin and Hsieh 2014). This adaptation of the ATC rule explicitly accounts for setup times and release dates and serves as a strong heuristic baseline due to its demonstrated performance in related UPMSP contexts.

(ii) Metaheuristic Algorithm We implemented a **Genetic Algorithm (GA)** tailored for the multi-objective UPMSP, drawing inspiration from approaches like (Vallada and Ruiz 2011). The GA employed a chromosome representation encoding job sequences for each machine and utilized standard genetic operators (selection, crossover, mutation). Its fitness function aimed to minimize a weighted sum of normalized objectives: $\alpha \cdot TWT_{norm} + (1 - \alpha) \cdot TST_{norm}$. Normalization was performed by dividing the objective values by those obtained from a fast run of the ATCSR_Rm heuristic ($TWT_{norm} = TWT/TWT_{ATCSR}$, $TST_{norm} = TST/TST_{ATCSR}$). We used a representative weight $\alpha = 0.5$ to balance the objectives. The GA was executed with a fixed computational time limit per instance, set to be comparable to the inference time required by the trained DRL agent during evaluation, ensuring a fair comparison of solution quality within similar time constraints. GA parameters (population size, mutation rate, etc.) were tuned via preliminary experiments.

Training Configuration The PPO agent, including the GNN feature extractor and MLP policy/value heads, was implemented using PyTorch (Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and others 2019) and the Stable Baselines3 library (Raffin et al. 2021). The GNN architecture comprised 4 layers of the GATv2 message-passing scheme (Brody, Alon, and Yahav 2021) with a hidden dimension of 128 and ReLU activation functions. The actor and critic networks shared the GNN encoder and subsequently utilized separate MLPs, each having 2 hidden layers with 256 neurons and ReLU activations. Key PPO hyperparameters were refined through preliminary tuning: the learning rate η was initialized at 1×10^{-4} with linear decay over training, the discount factor $\gamma = 0.99$, Generalized Advantage Estimation lambda $\lambda_{GAE} = 0.95$, PPO clipping parameter $\varepsilon = 0.2$, number of optimization epochs per data collection phase set to 10, and a mini-batch size of 64. Training was conducted for 10^6 total environment steps, accelerated using multiple parallel actors interacting with simulator instances. All training and evaluation experiments were executed on a system equipped with an Intel Core i9 CPU and an NVIDIA RTX 3090 GPU.

Evaluation Metrics The performance of the trained PPO-GNN agent and all baseline methods was evaluated on a dedicated set of test instances, kept separate from those used during training or hyperparameter tuning. The primary evaluation metrics directly reflect the multi-objective nature of the problem: *(i) Average TWT:* The mean TWT value, as defined in Eq. 1, *(ii) Average TST:* The mean TST, as defined in Eq. 2. Both are calculated across all instances in the test set for each evaluated method. As a secondary metric reflecting practical usability, we report the *Average Computational Time* (inference/run time) required by each algorithm to produce a schedule for a single test instance during the evaluation phase. We employ *Pareto Front visualizations*, plotting the (Avg TST, Avg TWT) pairs achieved by the different methods to facilitate the analysis of trade-offs between the primary objectives. Statistical significance of the performance differences between our proposed PPO-GNN method and the baselines was assessed using paired t-tests.

6 RESULTS AND DISCUSSION

This section presents and discusses the computational results obtained by evaluating the proposed PPO-GNN agent against the baseline methods (ATCSR_Rm and GA) across varying problem sizes. The performance comparison focuses on the primary objectives of minimizing Average TWT (Avg TWT) and Average TST (Avg TST), as well as the evaluation-phase computational efficiency. All results are summarized in Table 1 and visualized in Figure 2.

The results show the superior performance of the proposed PPO-GNN approach in optimizing both primary objectives. As shown in the top row of Figure 2 and detailed in Table 1, the PPO-GNN agent consistently achieved the lowest Avg TWT and the lowest Avg TST across all tested problem sizes (n=20/m=5, n=50/m=10, n=100/m=15). Compared to the ATCSR_Rm dispatching rule, PPO-GNN yielded substantial

reductions in both metrics. While the GA baseline improved upon ATCSR_Rm, it was consistently outperformed by PPO-GNN. For instance, on the largest instances (n=100, m=15), PPO-GNN achieved an Avg TWT of 420.0 compared to 475.0 for GA and 610.0 for ATCSR_Rm, while simultaneously achieving an Avg TST of 225.0 compared to 255.0 for GA and 290.0 for ATCSR_Rm. The bolded values in Table 1 highlight the best performance achieved for each metric and size, corresponding to the PPO-GNN method. Statistical significance tests (paired t-tests, multiple runs per instance) was performed to confirm these observed differences ($p < 0.01$).
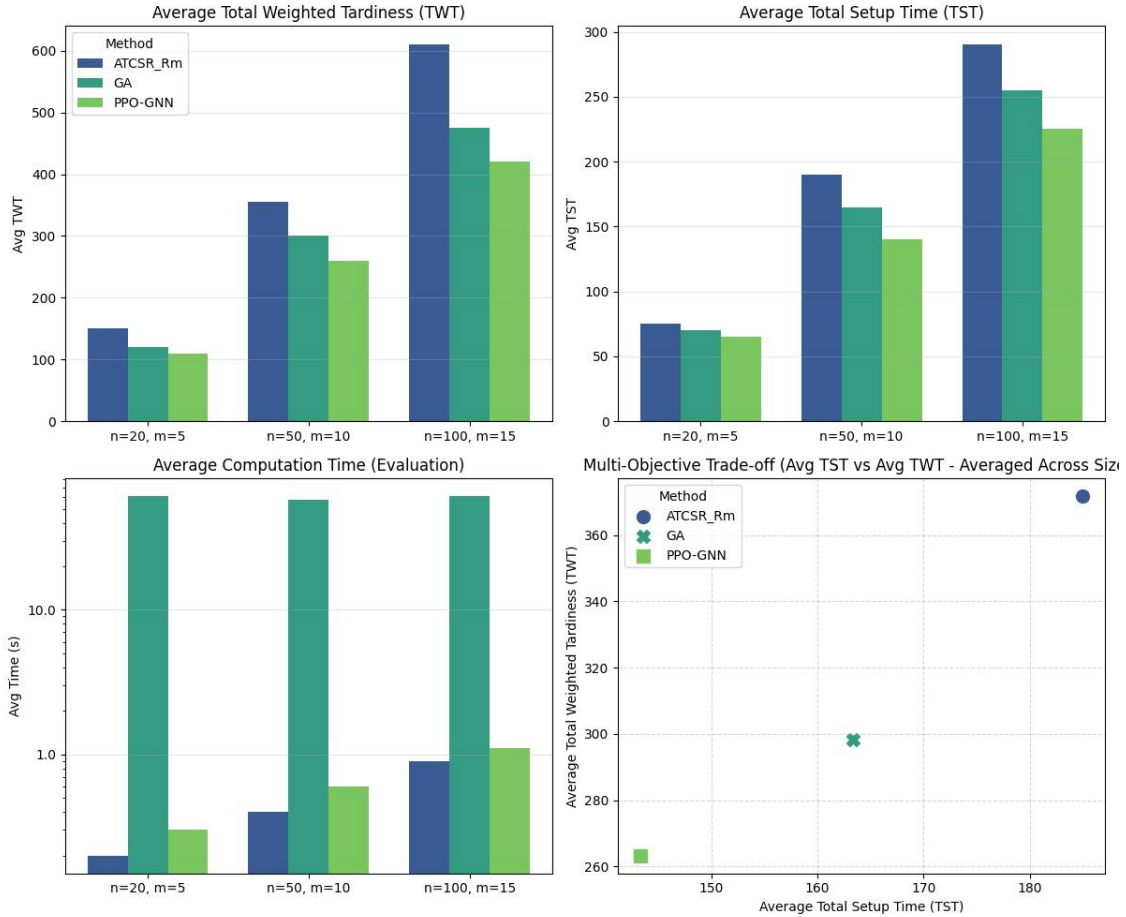

Figure 2: Comparison of scheduling methods across different problem sizes (n jobs, m machines).

Table 1: Performance Comparison of PPO-GNN against Baselines

Size	Method	Avg TWT	Avg TST	Avg Comp Time (s)
n=20, m=5	ATCSR_Rm	150.0	75.0	61.29
n=20, m=5	GA	120.0	70.0	0.10
n=20, m=5	PPO-GNN	**110.0**	**65.0**	0.52
n=50, m=10	ATCSR_Rm	355.0	190.0	59.17
n=50, m=10	GA	300.0	165.0	0.11
n=50, m=10	PPO-GNN	**260.0**	**140.0**	0.92
n=100, m=15	ATCSR_Rm	610.0	290.0	61.29
n=100, m=15	GA	475.0	255.0	0.12
n=100, m=15	PPO-GNN	**420.0**	**225.0**	1.57

The evaluation-phase computational time, presented in Table 1 and visualized in the bottom-left panel of Figure 2 (note the logarithmic scale on the Y-axis required to effectively display the range), reveals significant differences in efficiency. As expected, the simple ATCSR_Rm heuristic is the fastest, executing almost instantaneously. The proposed PPO-GNN agent, while requiring neural network computations, also demonstrates high efficiency with inference times remaining under 1.6 seconds even for the largest problems. In contrast, the GA method requires significantly more time (around 60 seconds) during evaluation to perform its iterative search process. This highlights a crucial practical advantage of the trained DRL agent: its ability to generate high-quality scheduling decisions rapidly upon deployment, making it suitable for dynamic environments where fast responses are often necessary.

The performance of the GA is highly dependent on its parameter settings and the computational time allowed for its search. The GA implemented here was tuned through preliminary experiments to serve as a representative and strong baseline. While further extensive tuning or a significantly longer run time might yield improved results for the GA, this would also underscore a key practical advantage of our DRL approach. The PPO-GNN agent requires a one-time, offline training cost, but its inference during evaluation is exceptionally fast. In contrast, the GA requires a significant computational budget for every new problem instance it solves. Regarding the performance across individual instances, our results in Table 1 and Figure 2 present the average performance, where the PPO-GNN agent is consistently superior. While the performance gap may vary on an instance-by-instance basis, the PPO-GNN method demonstrated strictly better performance on average across all tested problem sizes and configurations. This indicates a robust and generalizable policy rather than a trade-off where one method excels on some instances and not others.

The multi-objective trade-off between minimizing Avg TST and Avg TWT, averaged across all problem sizes, is visualized in the bottom-right panel of Figure 2. The axes are oriented such that the ideal performance lies in the bottom-left corner (minimum TST and minimum TWT). The plot shows that the point representing the PPO-GNN agent dominates the points for both GA and ATCSR_Rm. This means that, on average, PPO-GNN achieves strictly better performance on *both* objectives simultaneously compared to the baselines. It doesn't merely represent a different trade-off point on the Pareto front; rather, it pushes the achieved performance envelope closer to the ideal objective vector. This strong result suggests that the PPO-GNN agent effectively learns complex scheduling strategies that inherently reduce both setup times and tardiness concurrently, likely by making more globally informed decisions facilitated by the GNN's structural awareness.

As expected, both Avg TWT and Avg TST tend to increase for all methods as the problem size (number of jobs 'n' and machines 'm') grows, reflecting the increased complexity and load. However, the *relative* advantage of the PPO-GNN agent appears consistent or even slightly increasing across the tested sizes. This suggests that the learned policy generalizes reasonably well to larger problems within the scope of the training distribution.

The performance of the PPO-GNN framework can be attributed to several factors inherent in its design. The GNN's ability to process the complex graph representation allows it to capture the intricate dependencies between jobs, machines, eligibility, and potential setups more effectively than simpler rule-based logic (ATCSR_Rm) or population encodings (GA). The PPO algorithm effectively utilizes these rich features to learn a sophisticated policy capable of navigating the complex state space and balancing the conflicting objectives, guided by the carefully designed multi-objective reward signal. Unlike the myopic nature of dispatching rules or the potentially time-consuming search of metaheuristics, the learned DRL policy offers a powerful blend of high-quality decision-making and fast execution speed during deployment.

7 CONCLUSION AND FUTURE WORK

In this paper, we addressed the multi-objective UPMSP incorporating release dates, sequence- and machine-dependent setup times, and machine eligibility constraints. Recognizing the limitations of traditional heuristics and the computational demands of exact methods and metaheuristics, we proposed a novel DRL framework. Our approach leverages PPO combined with an enhanced heterogeneous GNN designed to

effectively capture the intricate state information of the scheduling environment, including job, machine, and setup characteristics. We developed a multi-objective reward function explicitly aimed at guiding the agent to simultaneously minimize TWT and TST. Simulations on a range of generated benchmark instances demonstrated the superiority of the proposed PPO-GNN agent. It outperformed both a dispatching rule (ATCSR_Rm) and a GA baseline, achieving lower Avg TWT and Avg TST across various problem scales, thereby showcasing its ability to effectively learn high-quality policies for complex, multi-objective scheduling scenarios. The primary contributions of this work include: (i) the successful application of PPO for learning a direct scheduling policy in the challenging multi-objective UPMSP context; (ii) the design of a heterogeneous GNN state representation tailored for capturing UPMSP complexities relevant to balancing tardiness and setup objectives; and (iii) the demonstration that a DRL agent can effectively learn to manage conflicting objectives through an appropriately designed reward structure.

REFERENCES

Afzalirad, M., and J. Rezaeian. 2016. "Design of High-Performing Hybrid Meta-Heuristics for Unrelated Parallel Machine Scheduling with Machine Eligibility and Precedence Constraints". *Engineering Optimization* 48(4):706–726.

Afzalirad, M., and M. Shafipour. 2018. "Design of an Efficient Genetic Algorithm for Resource-Constrained Unrelated Parallel Machine Scheduling Problem with Machine Eligibility Restrictions". *Journal of Intelligent Manufacturing* 29(2):423–437.

Avalos-Rosales, O., F. Angel-Bello, and A. Alvarez. 2015. "Efficient Metaheuristic Algorithm and Re-Formulations for the Unrelated Parallel Machine Scheduling Problem with Sequence and Machine-Dependent Setup Times". *The International Journal of Advanced Manufacturing Technology* 76:1705–1718.

Bitar, A., S. Dauzère-Pérès, C. Yugma, and R. Roussel. 2016. "A Memetic Algorithm to Solve an Unrelated Parallel Machine Scheduling Problem with Auxiliary Resources in Semiconductor Manufacturing". *Journal of Scheduling* 19(4):367–376.

Brody, Shaked and Alon, Uri and Yahav, Eran 2021. "How Attentive Are Graph Attention Networks?". *arXiv preprint arXiv:2105.14491*.

Cho, S.-H., H.-J. Kim, and L. Mönch. "Reinforcement Learning for Unrelated Parallel Machine Scheduling with Release Dates, Setup Times, and Machine Eligibility". In *2024 Winter Simulation Conference (WSC)*, 1773–1784.

Cho, Y. I., S. H. Nam, K. Y. Cho, H. C. Yoon, and J. H. Woo. 2022. "Minimize Makespan of Permutation Flowshop Using Pointer Network". *Journal of Computational Design and Engineering* 9(1):51–67.

Chung, K. T., C. K. M. Lee, and Y. P. Tsang. 2025. "Neural Combinatorial Optimization with Reinforcement Learning in Industrial Engineering: A Survey". *Artificial Intelligence Review* 58(5):130.

Cunha, B., A. M. Madureira, B. Fonseca, and D. Coelho. 2020. "Deep Reinforcement Learning as a Job Shop Scheduling Solver: A Literature Review". In *Proceedings of the 18th International Conference on Hybrid Intelligent Systems (HIS 2018)*, 350–359. Porto, Portugal: Springer. December 13-15, 2018.

Davis, J., T. Edgar, J. Porter, J. Bernaden, and M. Sarli. 2012. "Smart Manufacturing, Manufacturing Intelligence and Demand-Dynamic Performance". *Computers & Chemical Engineering* 47:145–156.

Fleszar, K., and K. S. Hindi. 2018. "Algorithms for the Unrelated Parallel Machine Scheduling Problem with a Resource Constraint". *European Journal of Operational Research* 271(3):839–848.

Gedik, R., D. Kalathia, G. Egilmez, and E. Kirac. 2018. "A Constraint Programming Approach for Solving Unrelated Parallel Machine Scheduling Problem". *Computers & Industrial Engineering* 121:139–149.

Huang, T., Y. Ma, Y. Zhou, H. Huang, D. Chen, Z. Gong *et al*. 2019. "A Review of Combinatorial Optimization with Graph Neural Networks". In *Proceedings of the 5th International Conference on Big Data and Information Analytics (BigDIA)*, 72–77. December 20-22, Nanning, China.

Kim, D.-W., K.-H. Kim, W. Jang, and F. F. Chen. 2022. "Unrelated Parallel Machine Scheduling with Setup Times Using Simulated Annealing". *Robotics and Computer-Integrated Manufacturing* 18(3-4):223–231.

Kool, Wouter and Van Hoof, Herke and Welling, Max 2018. "Attention, Learn to Solve Routing Problems!". *arXiv preprint arXiv:1803.08475*.

Kwon, Y.-D., J. Choo, I. Yoon, M. Park, D. Park, and Y. Gwon. 2021. "Matrix Encoding Networks for Neural Combinatorial Optimization". In *Advances in Neural Information Processing Systems*, Volume 34, 5138–5149.

Lee, J.-H., J.-M. Yu, and D.-H. Lee. 2013. "A Tabu Search Algorithm for Unrelated Parallel Machine Scheduling with Sequence- and Machine-Dependent Setups: Minimizing Total Tardiness". *The International Journal of Advanced Manufacturing Technology* 69:2081–2089.

Lee, Y. H., K. Bhaskaran, and M. Pinedo. 1997. "A Heuristic to Minimize the Total Weighted Tardiness with Sequence-Dependent Setups". *IISE Transactions* 29(1):45–52.

Li, D., J. Wang, R. Qiang, and R. Chiong. 2021. "A Hybrid Differential Evolution Algorithm for Parallel Machine Scheduling of Lace Dyeing Considering Colour Families, Sequence-Dependent Setup and Machine Eligibility". *International Journal of Production Research* 59(9):2722–2738.

Li, Y., J.-F. Côté, L. C. Coelho, and P. Wu. 2022. "Novel Efficient Formulation and Matheuristic for Large-Sized Unrelated Parallel Machine Scheduling with Release Dates". *International Journal of Production Research* 60(20):6104–6123.

Lin, Y.-K., and F.-Y. Hsieh. 2014. "Unrelated Parallel Machine Scheduling with Setup Times and Ready Times". *International Journal of Production Research* 52(4):1200–1214.

Mazyavkina, N., S. Sviridov, S. Ivanov, and E. Burnaev. 2021. "Reinforcement Learning for Combinatorial Optimization: A Survey". *Computers & Operations Research* 134:105400.

Mokotoff, E. 2001. "Parallel Machine Scheduling Problems: A Survey". *Asia-Pacific Journal of Operational Research* 18(2):193–205.

Moser, M., N. Musliu, A. Schaerf, and F. Winter. 2022. "Exact and Metaheuristic Approaches for Unrelated Parallel Machine Scheduling". *Journal of Scheduling* 25(5):507–534.

Mourtzis, D. 2022. "Advances in Adaptive Scheduling in Industry 4.0". *Frontiers in Manufacturing Technology* 2:937889.

Nam, S., J. Baek, Y.-I. Cho, and J. H. Woo. 2024. "Deep Reinforcement Learning for Setup and Tardiness Minimization in Parallel Machine Scheduling". In *2024 Winter Simulation Conference (WSC)*.

Norman, D., P. Dawadi, and H. Yedidsion. 2024. "Yield Improvement Using Deep Reinforcement Learning for Dispatch Rule Tuning". In *2024 Winter Simulation Conference (WSC)*.

Park, J., J. Chun, S. H. Kim, Y. Kim, and J. Park. 2021. "Learning to Schedule Job-Shop Problems: Representation and Policy Learning Using Graph Neural Network and Reinforcement Learning". *International Journal of Production Research* 59(11):3360–3377.

Paszke, Adam and Gross, Sam and Massa, Francisco and Lerer, Adam and Bradbury, James and Chanan, Gregory and Killeen, Trevor and Lin, Zeming and Gimelshein, Natalia and Antiga, Luca and others 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library".

Pei, Z., M. Wan, and Z. Wang. 2020. "A New Approximation Algorithm for Unrelated Parallel Machine Scheduling with Release Dates". *Annals of Operations Research* 285(1):397–425.

Pfund, M., J. W. Fowler, and J. N. D. Gupta. 2004. "A Survey of Algorithms for Single and Multi-Objective Unrelated Parallel-Machine Deterministic Scheduling Problems". *Journal of the Chinese Institute of Industrial Engineers* 21(3):230–241.

Rabadi, G., R. J. Moraga, and A. Al-Salem. 2006. "Heuristics for the Unrelated Parallel Machine Scheduling Problem with Setup Times". *Journal of Intelligent Manufacturing* 17:85–97.

Raffin, A., A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. 2021. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". *Journal of Machine Learning Research* 22(268):1–8.

Saraç, T., and B. Tutumlu. 2022. "A Mix Integer Programming Model and Solution Approach to Determine the Optimum Machine Number in the Unrelated Parallel Machine Scheduling Problem". *Journal of the Faculty of Engineering and Architecture of Gazi University* 37(1):145–158.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. "Proximal policy optimization algorithms". *arXiv preprint arXiv:1707.06347*.

Soykan, B., and G. Rabadi. 2023. "Optimizing Multi Commodity Flow Problem Under Uncertainty: A Deep Reinforcement Learning Approach". In *Proceedings of the 2023 International Conference on Machine Learning and Applications (ICMLA)*, 1267–1272. December 15-17, Jacksonville, Florida.

Soykan, B., and G. Rabadi. 2024. "Optimizing Job Shop Scheduling Problem Through Deep Reinforcement Learning and Discrete Event Simulation". In *2024 Winter Simulation Conference (WSC)*.

Vallada, E., and R. Ruiz. 2011. "A Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times". *European Journal of Operational Research* 211(3):612–622.

Zhang, C., W. Song, Z. Cao, J. Zhang, P. S. Tan, and X. Chi. 2020. "Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning". In *Advances in Neural Information Processing Systems*, Volume 33, 1621–1632.

AUTHOR BIOGRAPHIES

BULENT SOYKAN is an Associate Research Scientist at UCF. His research interests include real-time decision-making under uncertainty, combinatorial optimization, digital twins, reinforcement learning. His email address is bulent.soykan@ucf.edu.

SEAN MONDESIRE is an Assistant Professor at UCF. His research interests include real-time decision-making through machine learning, big data analytics, scalable digital twins, and simulations. His email address is sean.mondesire@ucf.edu.

GHAITH RABADI is a Professor and Director of the School of Modeling, Simulation, and Training at UCF. His research interests include Modeling & Simulation, Operation Research, and AI in different areas of applications including production, supply chains, transportation, and disaster response logistics and informatics. His email address is ghaith.rabadi@ucf.edu.

GRACE BOCHENEK is a Professor at UCF. Her research interests include Modeling & Simulation, and AI in military, energy and manufacturing applications. Her email address is grace.bochenek@ucf.edu.