International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: www.tandfonline.com/journals/tprs20

Balancing and scheduling of transfer lines for cycle time and peak power minimisation: a learning-enhanced multi-objective variable neighbourhood search

Pedram Beldar , Ehsan Mahmoodi , Mostafa Kermani , Gary Linnéusson & Olga Battaia

To cite this article: Pedram Beldar , Ehsan Mahmoodi , Mostafa Kermani , Gary Linnéusson & Olga Battaia (05 Oct 2026): Balancing and scheduling of transfer lines for cycle time and peak power minimisation: a learning-enhanced multi-objective variable neighbourhood search, International Journal of Production Research, DOI: 10.1080/00207543.2026.2741382

To link to this article: https://doi.org/10.1080/00207543.2026.2741382

© 2026 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

Published online: 05 Oct 2026.

Submit your article to this journal

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tprs20

INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
https://doi.org/10.1080/00207543.2026.2741382

Taylor & Francis
by informa

OPEN ACCESS Check for updates

Balancing and scheduling of transfer lines for cycle time and peak power minimisation: a learning-enhanced multi-objective variable neighbourhood search

Pedram Beldar[a], Ehsan Mahmoodi[b], Mostafa Kermani[a], Gary Linnéusson[c] and Olga Battaia[d]

[a]Division of Electrical Engineering and Automation, School of Engineering Science, University of Skövde, Skövde, Sweden; [b]Division of Mechanical and Industrial Engineering, School of Engineering Science, University of Skövde, Skövde, Sweden; [c]Division of Human, Technology, and Management, School of Engineering Science, University of Skövde, Skövde, Sweden; [d]Centre of Excellence in Supply Chain and Transportation (CESIT), KEDGE Business School, Bordeaux, France

ABSTRACT

As global electricity demand surges due to industrial electrification and digitalisation, peak power (PP) management has emerged as a critical challenge in energy-intensive manufacturing, affecting operational costs and grid capacity. PP is defined as the maximum instantaneous total power demand across all workstations during the production cycle, and its minimisation is essential for reducing demand charges and avoiding grid overloads. This study introduces the balancing and scheduling of machining transfer lines as a bi-objective optimisation problem that simultaneously minimises cycle time (CT) and PP under precedence, inclusion/exclusion, and technological compatibility constraints. A multi-objective constraint programming (CP) model is formulated and solved via a bidirectional ε-constraint method, generating high-quality Pareto sets for small-scale instances. To overcome the computational intractability of CP for medium- and large-scale problems, a multi-objective variable neighbourhood search (MOVNS) enhanced with deep reinforcement learning (NN_MOVNS) is proposed, in which a Deep Q-Network adaptively guides neighbourhood operator selection throughout the search. Computational experiments on 55 benchmark instances demonstrate that NN_MOVNS consistently outperforms competing algorithms across all problem scales, with improvements becoming increasingly pronounced on larger instances, as confirmed by statistical significance tests. Peak shaving experiments confirm PP reductions ranging from 33% to 73% with modest CT increases averaging 5.7%. An industrial-scale test instance of a 148-task automotive cylinder head production line achieves an 11.8% CT reduction and a 63.1% PP reduction, lowering the peak-to-average power ratio from 2.35 to 1.49.

ARTICLE HISTORY
Received 3 January 2026
Accepted 22 September 2026

KEYWORDS
Peak shaving; energy redistribution; cycle time; line balancing; deep reinforcement learning; variable neighbourhood search

SUBJECT CLASSIFICATIONS
Scheduling; production modelling; metaheuristics

1. Introduction

Global electricity demand is rising rapidly as industrial processes become increasingly electrified and digitalised. At the same time, escalating energy prices, power grid limitations, and supply chain disruptions have made efficient industrial power management a central challenge in the 21st century. This challenge is reflected in unprecedented levels of industrial energy dependency across the world's largest manufacturing economies. For example, manufacturing in China alone accounts for more than half of national final energy demand and a dominant share of energy-related CO_2 emissions Sandalow et al. 2022). Moreover, global energy consumption is projected to surge by 50% by 2050 (U.S. Energy Information Administration. 2021), with manufacturing's share of total electricity consumption expected to nearly double from 22% to 46% (Delorme and Gianessi 2024). As a result, the need for energy-efficient manufacturing systems has become more urgent than ever (L. Zhang et al. 2017).

In response, researchers have explored several strategies aimed at improving energy performance in manufacturing systems (Chen et al. 2025; Rakovitis et al. 2022), typically targeting one or more of the following objectives (Bänsch et al. 2021): (1) minimising total energy consumption, (2) minimising energy cost under pricing mechanisms such as time-of-use tariffs, and (3) minimising peak power (PP). Although minimising total energy consumption and cost are well-established objectives, PP minimisation remains comparatively underexplored. Among industrial systems, synchronised high-power operations in automated production lines represent

CONTACT Pedram Beldar pedram.beldar@his.se Division of Electrical Engineering and Automation, School of Engineering Science, University of Skövde, P.O. Box 408, Skövde SE-541 28, Sweden

© 2026 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

a particularly challenging source of recurring peak demand. Machining transfer lines in automotive and aerospace manufacturing are a prime example (Beldar et al. 2024). In machining transfer lines, operations grouped into blocks are executed by spindle heads at sequentially arranged stations. Within each station, blocks execute sequentially, while all stations operate simultaneously across the transfer line (Guschinskaya and Dolgui 2009). This simultaneous activation of spindle heads across stations repeatedly generates high-intensity PP per cycle, amplifying electricity demand charges and increasing operational risks due to grid capacity limitation. Moreover, in this setting, total energy consumption per production cycle is largely fixed because all tasks must be executed exactly once and each task has a given processing time and power requirement. By contrast, energy cost minimisation via time-of-use pricing is of limited applicability, since the transfer line operates continuously and cyclically, leaving little room to shift the overall production window to lower-cost tariff periods. Therefore, the main energy-related decision lever is not how much energy is consumed or when the production cycle occurs, but how power demand is distributed within the cycle. This makes PP a particularly relevant objective in machining transfer lines.

Existing approaches typically optimise either line balancing or scheduling separately (Ersin Telemeci and Azizoğlu 2024; Gianessi, Delorme, and Masmoudi 2019), which is shown herein to be inadequate when both cycle time (CT) and PP are critical performance metrics. Line balancing distributes tasks across workstation blocks and stations, avoiding workload imbalances that could create processing bottlenecks and increase CT. However, even if tasks are well balanced, simultaneous scheduling of high-power blocks across multiple workstations can generate recurring PP, thereby inflating costs. Scheduling aims to stagger these blocks to smooth the power profile; however, without proper task balancing, scheduling alone may result in inefficient use of workstation capacity or increased CT. Consequently, the assignment of tasks to blocks and workstations (balancing) and the timing of block execution scheduling must be optimised simultaneously. Hence, this paper refers to this problem as the balancing and scheduling of machining transfer lines (BSMTL). To illustrate the trade-off between CT and PP, Figure 1(a,b) show two alternative schedules for the same transfer line instance, which achieve opposite balances between the two objectives and underscore the need for bi-objective optimisation that jointly considers both metrics. To address BSMTL, a complex combinatorial optimisation problem, a multi-objective variable neighbourhood search (MOVNS) algorithm, enhanced with a deep reinforcement learning-based operator selection mechanism, named NN_MOVNS, is proposed. To the best of our knowledge, no existing approach simultaneously addresses CT and PP optimisation in machining transfer lines under the full set of precedence, inclusion/exclusion, and technological constraints. This gap motivates the following research questions:

RQ1: How can the BSMTL be jointly formulated as a bi-objective optimisation problem to simultaneously minimise CT and PP under precedence, inclusion/exclusion, and technological compatibility constraints?

RQ2: To what extent can the proposed NN_MOVNS algorithm outperform both exact methods and standard metaheuristics in terms of Pareto front quality across small, medium, and large-scale instances?

RQ3: What are the practical implications of the proposed approach in terms of PP reduction, energy redistribution, and peak-to-average power ratio (PAPR) improvement in industrial-scale machining transfer lines?

The remainder of this paper is organised as follows. Section 2 reviews the literature on transfer line balancing

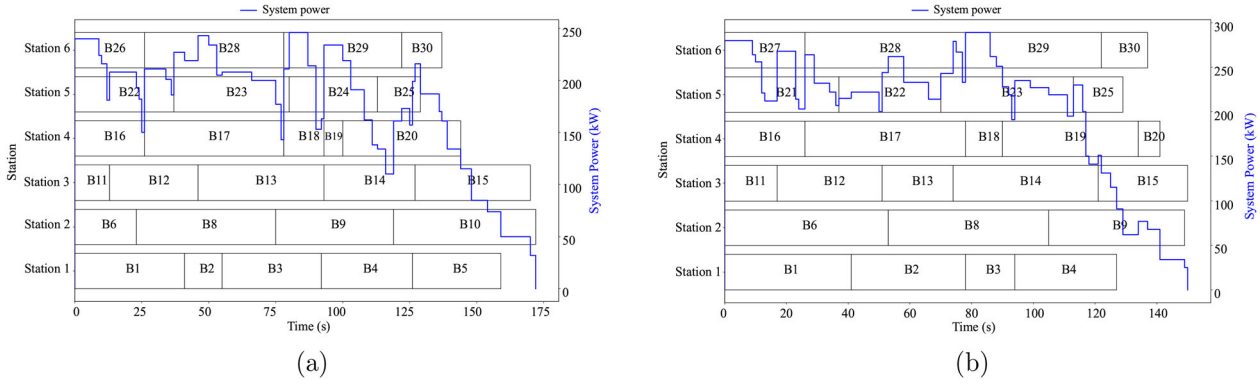

Figure 1. Examples of two solutions illustrating the trade-off between CT and PP in a transfer line. Each subfigure shows a Gantt chart, the system power demand profile, and average power level. (a) Solution A: CT = 172 s, PP = 246 kW and (b) Solution B: CT = 150 s, PP = 289 kW.

problems and PP minimisation, identifies the research gap, and summarises the main scientific contributions. Section 3 presents the problem description and the constraint programming (CP) model. Section 4 describes the solution method, including the constructive heuristic, shaking and neighbourhood operators, as well as the key components of the neural network-enhanced MOVNS framework. Section 5 presents the experimental evaluation, including the experimental design, competing algorithms, performance metrics, and parameter tuning. Section 6 reports the comparative performance analysis, including pooled-front evaluation, comparisons with the exact solver and metaheuristics, coverage analysis, statistical significance testing, Pareto front visualisation, and neural network component analysis. Section 7 presents the industrial-scale test instance together with the peak shaving and PAPR analyses, and synthesises the findings and managerial insights through the lens of the research questions. Finally, Section 8 concludes the paper and outlines future research directions.

2. Literature review

This section provides a structured review of the scientific literature relevant to the problem of minimising PP and CT in machining transfer lines. Section 2.1 establishes the foundational concepts of transfer line balancing. Section 2.2 examines PP considerations in general line balancing, drawing transferable insights due to the scarcity of transfer line-specific energy studies. Section 2.3 identifies the research gap, and Section 2.4 summarises the main scientific contributions of this work.

2.1. Transfer line balancing problem

The Transfer Line Balancing Problem (TLBP) is defined as the optimisation of assigning machining operations to workstations in an automated production line to minimise one or more objectives such as number of stations, line cost, or CT, while satisfying precedence constraints, technological limitations, and equipment compatibility requirement (Battaïa, Dolgui, and Guschinsky 2024; Lahrichi, Grangeon, et al. 2021). Transfer lines are characterised by their serial configuration of workstations connected by automated material handling systems, where parts move automatically from one station to the next, with each station equipped with specialised tools that perform machining operations using single or multiple spindle heads (Nourmohammadi et al. 2024). Unlike assembly line balancing, which often focuses on manual or semi-automated task allocation, TLBPs must address distinct features of automated machining systems, including high automation levels, the capacity for parallel processing via multi-spindle heads, equipment design decisions that directly influence task times, and complex constraints related to tool compatibility and task-machine accessibility (Beldar et al. 2025).

According to the comprehensive survey recently carried out by Beldar et al. (2025), most studies on the TLBP have focussed on cost-related objectives, particularly minimising line cost (Battaïa, Dolgui, and Guschinsky 2023, 2024) and the number of machines (Lahrichi, Deroussi, et al. 2021; Lahrichi, Grangeon, et al. 2021). While cost considerations dominate much of the research due to their direct impact on capital investment, there is a noticeable gap in addressing time-related objectives like CT minimisation and efficiency-related objectives like power or energy efficiency metrics.

Focussing on CT considerations, several key works provide valuable insights. Ersin Telemeci and Azizoğlu (2024) address the Type II TLBP, formulating a mixed-integer linear programming (MILP) model and a branch-and-bound algorithm to minimise CT. Nourmohammadi et al. (2024) study the balancing and scheduling of transfer lines, introducing a MILP model and a multi-objective optimisation approach to minimise CT and total energy cost. Their results highlight how varying energy modes influence sustainability, providing insights into energy-efficient production strategies for small to medium-scale problems. Beldar et al. (2024) propose a heuristic approach for balancing transfer lines, incorporating a wide range of equipment, time, and operational factors. A numerical example illustrates the method's practical applicability and its effectiveness in generating feasible solutions for real-world production settings.

Energy considerations, though less prevalent, are gaining traction in TLBP research as sustainability becomes a pressing concern. Petersen et al. (2024) address TLBP in the context of energy-efficient manufacturing by jointly considering CT and energy consumption. A mixed-integer linear formulation is presented, and a multi-objective Non-dominated Sorting Genetic Algorithm II (NSGA-II) is developed to handle the problem's complexity. Their results demonstrate how different energy modes influence sustainability, offering manufacturers actionable insights for energy-efficient production strategies.

Despite growing interest in energy-related objectives, research on incorporating PP into TLBP remains scarce. To better understand how such considerations can be incorporated, it is necessary to look beyond transfer lines and examine the broader body of line balancing research, where PP has recently emerged as a distinct objective alongside CT, cost, and energy metrics.

2.2. PP considerations in line balancing

PP in line balancing is defined as the maximum instantaneous electric power demand during the production process. Unlike traditional assembly line balancing problems that focus primarily on balancing workload and minimising CT or number of stations (Battaïa and Dolgui 2022), PP considerations introduce additional scheduling and sequencing complexities, as overlapping high-energy tasks can cause peak demand that elevate operational expenses and risk outages (Delorme et al. 2023; Delorme and Gianessi 2024). PP minimisation in line balancing problems must address the temporal distribution of energy-intensive tasks, often requiring integrated assignment and scheduling decisions to smooth power profiles while respecting precedence constraints, takt time, and workstation limits.

Focussing on PP considerations, several key works provide valuable insights. Lamy, Delorme, and Gianessi (2020) examine a variant of the simple assembly line balancing problem with PP minimisation (SALB3PM), formulating a model that integrates task sequencing without idle times to minimise PP. Their metaheuristic approach demonstrates that sequencing decisions can minimise peaks in paced lines. Delorme and Gianessi (2024) study PP minimisation in reconfigurable parallel-serial lines with crossover, introducing an integer programming model and matheuristic that consider workstation resource allocation and task scheduling to minimise PP.

Building on these foundations, Gianessi, Delorme, and Masmoudi (2019) deal with SALB3PM, presenting an integer linear program that minimises PP through integrated assignment and scheduling. Fang et al. (2020) explore multi-objective optimisation in mixed-model multi-robotic disassembly lines, minimising CT, peak workstation and total energy consumption under interval processing times. Their evolutionary simulated annealing algorithm yields uniformly distributed Pareto-optimal solutions, highlighting the trade-offs between energy metrics in robotic systems. Similarly, Z. Zheng, Cherif, and Shibasaki (2024) propose a maximum satisfiability-based model for the SALB3PM, efficiently handling large instances by encoding power constraints and achieving robust performance across benchmarks.

In disassembly contexts, Fang et al. (2019) tackle many-objective optimisation for mixed-model disassembly lines with multi-robotic workstations, balancing CT, energy consumption, and robot usage. Their problem-specific evolutionary algorithm provides insights into convergence-diversity trade-offs, with computational experiments showing superior performance over representative methods. H. Ming, Liu, and Pham (2019) address uncertain processing times in multi-robotic disassembly lines, developing an MILP model that minimises CT, PP, total energy consumption, and hazardous task costs, validated through illustrative examples. Z. Zhang et al. (2025) consider multi-robot destructive disassembly line balancing under component non-disassemblability, optimising CT, energy consumption, and profit with an improved multi-objective water cycle algorithm.

Further advancements in reconfigurable systems are evident in Delorme and Gianessi (2022), who minimise PP in parallel-serial lines with crossover, proposing an integer linear program that integrates balancing and scheduling for an energy-efficient system. Their approach demonstrates substantial peak reductions through resource reconfiguration. Delorme et al. (2023) introduce a decoder for permutation-based heuristics in SALB3PM with earliest starting dates, optimally splitting task sequences to minimise peaks, with preliminary tests confirming efficiency in large instances. Taken together, these works confirm that PP-oriented line balancing and scheduling is an active and growing research area, but one that has not yet been extended to multi-spindle block-based machining transfer lines.

2.3. Research gap

Although prior research on PP minimisation in assembly, reconfigurable, and disassembly lines offers valuable conceptual and methodological foundations, these studies focus on line structures and decision variables that differ significantly from those in machining transfer lines. Specifically, BSMTL combines characteristics that are not jointly addressed in the existing literature, including parallel multi-spindle processing within blocks, synchronised station operations through automated transfer mechanisms, and strict equipment- and compatibility-related constraints. When considered under a single objective, the problem relates to two distinct research streams: if only CT is minimised, it connects to transfer line balancing studies in multi-spindle settings (Ersin Telemeci and Azizoğlu 2024) and balancing-and-scheduling formulations for mono-spindle flexible transfer lines (Nourmohammadi et al. 2024); if only PP is minimised, it relates in spirit to PP-oriented balancing-and-scheduling models for simple assembly lines (Gianessi, Delorme, and Masmoudi 2019) and mono-spindle reconfigurable lines (Delorme and Gianessi 2022). However, none of these works simultaneously address multi-spindle block-based transfer lines, joint balancing and scheduling decisions, and PP as an

objective. In both mono-spindle and multi-spindle transfer lines, all stations operate simultaneously, contributing to system-level PP. However, in mono-spindle lines, only one task is executed at a time within each station, so power demands do not accumulate within a station. In contrast, in multi-spindle block-based lines all tasks within a block execute simultaneously, causing their power demands to sum instantaneously. Block composition therefore becomes itself a PP-determining decision, an additional source of complexity that does not arise in mono-spindle settings. Treating CT and PP independently is thus insufficient: CT-only approaches do not control how block composition generates instantaneous power peaks, while PP-only approaches may inflate CT by creating workload imbalances or scheduling bottlenecks. Since the task-to-block assignment determines which power demands accumulate simultaneously and the block execution sequence determines when those peaks occur, both decisions must be optimised jointly. This reveals a methodological gap in the literature and motivates the bi-objective models and solution approaches developed in this work.

2.4. Our main scientific contributions

In this work, we address the BSMTL problem, characterised by the joint optimisation of PP and CT under precedence, inclusion/exclusion, and technological compatibility constraints. The contributions of this study are classified under three categories:

(i) Theoretical contributions: A multi-objective CP model is formulated and solved via a bidirectional ε-constraint method (Bérubé, Gendreau, and Potvin 2009), generating high-quality Pareto sets for small-scale instances. This formulation establishes a rigorous mathematical foundation for the joint CT–PP optimisation in machining transfer lines, a problem class that has received limited formal treatment in the literature.

(ii) Methodological contributions: A MOVNS algorithm is proposed, featuring a two-phase constructive heuristic for initialisation: in the first phase, tasks are assigned to workstations with integrated constraint handling to ensure feasibility, while in the second phase, blocks are created on each workstation to establish a high-quality starting solution. The MOVNS framework is further augmented with a neural network-based operator selection mechanism (NN_MOVNS) that employs a Deep Q-Network (DQN) to dynamically adapt search operators during optimisation, significantly improving exploration capabilities over static operator selection strategies.

(iii) Managerial contributions: The practical implications of the proposed approach are assessed using two metrics, the Peak Shaving Index (PSI) and the Energy Redistribution Index (ERI), commonly used in demand response contexts to assess and mitigate demand peaks. These metrics provide operations managers with actionable tools to quantify and communicate the benefits of PP reduction in energy-intensive manufacturing systems.

3. Problem description

A machining transfer line composed of a finite set of workstations $M = \{1, \ldots, NM\}$ is considered, where each workstation $k \in M$ contains a limited number of blocks. A set of tasks $O = \{1, \ldots, NT\}$ must be executed on the line. Each task $i \in O$ is characterised by a deterministic processing time Ta_i and a power demand Tp_i. The tasks are also subject to technological and organisational constraints. Precedence relations ($\text{pre}_{i,j}$) specify that task i must be assigned no later than task j in the line sequence. Workstation eligibility restrictions ($\text{tec}_{i,k}$) indicate whether task i can be processed on workstation k. Compatibility rules further restrict how tasks can be grouped: inclusion constraints ($\text{inc}_{i,j}$) require two tasks to be assigned to the same workstation; workstation-level exclusion constraints ($\text{exc}_{i,j}$) prevent two tasks from being assigned to the same workstation; and block-level exclusion constraints ($\text{exc_b}_{i,j}$) prevent two tasks from being assigned to the same block.

The assignment of tasks is performed in two stages. First, tasks are grouped into blocks, where the total number of blocks across the line is $|B| = Nb \cdot NM$, with Nb blocks allocated to each workstation. The set of block indices belonging to workstation k is formally defined as:

$$R(k) = \{(k-1) \cdot Nb + 1,\ (k-1) \cdot Nb + 2,\ \ldots,\ k \cdot Nb\}$$

This partitioning ensures that every block $q \in B$ belongs to exactly one workstation, and that the Nb blocks on each workstation are indexed consecutively without overlap. For instance, with $NM = 3$ workstations and $Nb = 2$ blocks per workstation, the total block set is $B = \{1, 2, 3, 4, 5, 6\}$, partitioned as $R(1) = \{1, 2\}$, $R(2) = \{3, 4\}$, and $R(3) = \{5, 6\}$, so that workstation 1 operates blocks 1 and 2, workstation 2 operates blocks 3 and 4, and workstation 3 operates blocks 5 and 6. Within a block, all tasks are executed simultaneously; thus, the processing time of a block is equal to the maximum processing time among its assigned tasks. The number of tasks per block is limited by a technological upper bound Mtb. Second, the blocks of each workstation are scheduled sequentially, and the workstation processing time is equal to the sum of its active block durations. Since all workstations operate in parallel, the CT of the transfer line is determined by the

bottleneck workstation, i.e. the one with the maximum cumulative block time.

At each time point $t \in \mathcal{T}$, the instantaneous power demand of the system is computed: from task activity variables $\text{act}_{t,q,i}$, which determine the power load of each block $\text{blk}_{q,t}$, then the power demand of each workstation $\text{mach}_{k,t}$, and finally the total system power demand sys_t. Accordingly, the *PP* is defined as the maximum instantaneous total system power demand over the planning horizon:

$$PP = \max_{t \in \mathcal{T}} \text{sys}_t = \max_{t \in \mathcal{T}} \sum_{k \in M} \text{mach}_{k,t}$$

The problem is a bi-objective optimisation problem aiming to (i) minimise the CT of the transfer line and (ii) minimise the peak system power. This problem is denoted as $S1 \mid C^{prec}, C^{exc}, C^{exc_b}, C^{inc}, SH^{multi}, Sch \mid CT, PP$ in the standard TLBP classification introduced by Beldar et al. (2025).

The following modelling assumptions underlie the proposed formulation:

- Processing times Ta_i and power demand values Tp_i are deterministic. Stochastic variability in task durations or power demand is not considered.
- The number of workstations *NM*, the maximum number of blocks per workstation *Nb*, and the maximum number of tasks per block *Mtb* are fixed input parameters.
- All inclusion, exclusion, block-level exclusion, and technological eligibility constraints are binary and predetermined, reflecting fixed engineering and organisational requirements.
- Each task is assigned to exactly one block on exactly one workstation. Task splitting or replication across blocks is not permitted.
- Tasks are non-preemptive: once a block begins execution, all its tasks run to completion without interruption.
- Blocks on each workstation are processed sequentially and used consecutively, with no idle gaps between active blocks.
- Inactive blocks and workstations contribute zero power at any time point $t \in \mathcal{T}$. The standby and idle power draw are not modelled.
- The planning horizon $\mathcal{T} = \{0, \ldots, H\}$ is discretised into unit time steps, where $H = \sum_{i \in O} Ta_i$ provides a conservative upper bound on CT.
- The model optimises a single production cycle for a single part type.

3.1. CP model

To generate high-quality Pareto sets for small-scale instances of the BSMTL, a multi-objective CP model is developed. CP is selected as the modelling framework because the BSMTL is inherently a disjunctive scheduling problem: it combines task-to-block assignment, block formation, and block execution scheduling under precedence, inclusion/exclusion, and technological compatibility constraints. CP handles this class of problems through global constraints and constraint propagation, without the auxiliary binary variables and big-M linearizations that MILP formulations of disjunctive problems typically require, and which tend to weaken LP relaxations and slow convergence. The scheduling component is represented compactly through interval decision variables and global constraints, which have dedicated propagation algorithms. An equivalent MILP formulation would require binary sequencing variables for every pair of blocks, leading to quadratic growth in model size. Recent works on integrated line balancing and scheduling problems confirm that CP models outperform state-of-the-art MILP formulations in objective value and solution time for this class of problems (Kheirabadi et al. 2025; Nourmohammadi et al. 2025; C. Zheng et al. 2025; C. Zheng, Li, and Ponnambalam 2025). The CP model is solved using IBM CP Optimiser and its solutions serve as reference Pareto sets for validating the proposed NN_MOVNS metaheuristic. For completeness, an equivalent MILP formulation of the BSMTL is provided in Appendix 1.

Parameters:

NT	Number of tasks
NM	Number of workstations
Nb	Number of blocks per workstation
Ta_i	Processing time of task i
Tp_i	Power demand of task i
$pre_{i,j}$	1 if task i must precede task j
$inc_{i,j}$	1 if tasks i and j must be on the same workstation
$exc_{i,j}$	1 if tasks i and j cannot be on the same workstation
$exc_b_{i,j}$	1 if tasks i and j cannot be in the same block
$tec_{i,k}$	1 if task i must be performed on workstation k
Mtb	Maximum number of tasks per block
H	Conservative upper bound on the discretised time horizon, $H = \sum_{i \in O} Ta_i$

Sets:

O	Set of tasks indexed by i ($O = \{1, \ldots, NT\}$)

M	Set of workstations indexed by k ($M = \{1, \ldots, NM\}$)
B	Set of blocks indexed by q ($B = \{1, \ldots, Nb \cdot NM\}$)
$R(k)$	Set of blocks on workstation k, $R(k) = \{(k-1) \cdot Nb + 1, \ldots, k \cdot Nb\}$
$\mathcal{T}$	Set of time points, $\mathcal{T} = \{0, \ldots, H\}$

Decision Variables:

X_i	Interval variable for scheduling task i, with fixed duration Ta_i, $\forall i \in O$
$Y_{i,q}$	Optional interval variable for assigning task i to block q, with fixed duration Ta_i, $\forall i \in O$, $\forall q \in B$
BS_q	Start time of block q, $\forall q \in B$
BE_q	End time of block q, $\forall q \in B$
a_q	Binary variable: 1 if block q is active, 0 otherwise, $\forall q \in B$
$\mathrm{act}_{t,q,i}$	Power demand by task i in block q at time t, $\forall t \in \mathcal{T}, q \in B, i \in O$
$\mathrm{blk}_{q,t}$	Total power in block q at time t, $\forall q \in B, t \in \mathcal{T}$
$\mathrm{mach}_{k,t}$	Total power in workstation k at time t, $\forall k \in M$, $t \in \mathcal{T}$
sys_t	System power at time t, $\forall t \in \mathcal{T}$
PP	Peak power, i.e. maximum instantaneous system power demand
CT	Cycle time

Model Formulation:

$$\text{Minimize} \quad \text{CT} \tag{1}$$

$$\text{Minimize} \quad \text{PP} \tag{2}$$

subject to:

$$\text{alternative}(X_i, \{Y_{i,q} \mid q \in B\}), \quad \forall\, i \in O \tag{3}$$

$$\sum_{q \in B} (q) \cdot \text{presenceOf}(Y_{i,q}) \leq \sum_{q \in B} (q) \cdot \text{presenceOf}(Y_{j,q}), \quad \forall\, i < j, \text{ if } \text{pre}_{i,j} = 1 \tag{4}$$

$$\text{presenceOf}(Y_{i,q}) + \text{presenceOf}(Y_{j,q}) \leq 1, \quad \forall\, i < j,\ \forall\, q \in B, \text{ if } \text{exc_b}_{i,j} = 1 \tag{5}$$

$$\sum_{q \in R(k)} \text{presenceOf}(Y_{i,q}) = \sum_{q \in R(k)} \text{presenceOf}(Y_{j,q}), \quad \forall\, i < j,\ \forall\, k \in M, \text{ if } \text{inc}_{i,j} = 1 \tag{6}$$

$$\sum_{q \in R(k)} \left(\text{presenceOf}(Y_{i,q}) + \text{presenceOf}(Y_{j,q})\right) \leq 1, \quad \forall\, i < j,\ \forall\, k \in M, \text{ if } \text{exc}_{i,j} = 1 \tag{7}$$

$$\sum_{q \in R(k)} \text{presenceOf}(Y_{i,q}) = 1, \quad \forall\, i \in O,\ \forall\, k \in M, \text{ if } \text{tec}_{i,k} = 1 \tag{8}$$

$$\sum_{i \in O} \text{presenceOf}(Y_{i,q}) \leq \text{Mtb}, \quad \forall\, q \in B \tag{9}$$

$$\text{If } q = (k-1) \cdot Nb + 1 : \text{presenceOf}(Y_{i,q}) = 1 \implies \text{startOf}(Y_{i,q}) = 0, \quad \forall\, i \in O, k \in M \tag{10}$$

$$\text{If presenceOf}(Y_{i,q}) = 1 \implies \mathrm{BS}_q = \text{startOf}(Y_{i,q}), \quad \forall\, i \in O,\ \forall\, q \in B \tag{11}$$

$$\mathrm{BE}_q = \max_{i \in O} \left(\text{endOf}(Y_{i,q}) \cdot \text{presenceOf}(Y_{i,q})\right), \quad \forall\, q \in B \tag{12}$$

$$\mathrm{a}_q = \min\left(1, \sum_{i \in O} \text{presenceOf}(Y_{i,q})\right), \quad \forall\, q \in B \tag{13}$$

$$a_{q_1} = 1 \ \wedge\ a_{q_2} = 1 \ \Rightarrow\ \mathrm{BE}_{q_1} = \mathrm{BS}_{q_2}, \quad \forall\, k \in M,\ \forall\, q_1, q_2 \in R(k),\ q_2 = q_1 + 1 \tag{14}$$

$$a_{q-1} \geq a_q, \quad \forall\, k \in M, \quad \forall\, q \in R(k),\ q > (k-1) \cdot Nb + 1 \tag{15}$$

$$\text{If } \mathrm{a}_q = 0, \text{ then } \mathrm{BE}_q = \mathrm{BS}_q, \quad \forall\, q \in B \tag{16}$$

$$\mathrm{act}_{t,q,i} = Tp_i \cdot \mathbb{I}\big\{\text{presenceOf}(Y_{i,q}) = 1 \wedge \text{startOf}(Y_{i,q}) \leq t < \text{endOf}(Y_{i,q})\big\}, \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{17}$$

$$\mathrm{blk}_{q,t} = \sum_{i \in O} \mathrm{act}_{t,q,i}, \quad \forall\, q \in B,\ t \in \mathcal{T} \tag{18}$$

$$\mathrm{mach}_{k,t} = \sum_{q \in R(k)} \mathrm{blk}_{q,t}, \quad \forall\, k \in M,\ t \in \mathcal{T} \tag{19}$$

$$\mathrm{sys}_t = \sum_{k \in M} \mathrm{mach}_{k,t}, \quad \forall\, t \in \mathcal{T} \tag{20}$$

$$\text{PP} \geq \mathrm{sys}_t, \quad \forall\, t \in \mathcal{T} \tag{21}$$

$$\text{CT} = \max_{i \in O, q \in B} \left(\text{endOf}(Y_{i,q}) \cdot \text{presenceOf}(Y_{i,q})\right) \tag{22}$$

The objective functions (1) and (2) minimise the CT and PP, respectively. Constraint (3) guarantees each task is assigned to exactly one block. Constraint (4) enforces precedence relationships between tasks. Constraint (5) prevents incompatible tasks from being assigned to the same block. Constraints (6) and (7) enforce workstation inclusion and exclusion rules for task assignments, respectively. Constraint (8) ensures tasks are assigned to compatible workstations. Constraint (9) limits the number of tasks per block. Constraint (10) sets the start time of the first block on each workstation to zero if active. Constraints (11) and (12) define the start and end

Table 1. Active constraint pairs for the illustrative example.

Constraint	Active pairs / entries
Precedence ($pre_{i,j} = 1$)	(1, 2), (1, 3), (1, 4), (1, 5), (2, 6), (3, 7), (4, 7), (5, 7), (6, 8), (7, 9), (8, 10), (9, 11), (10, 11)
Workstation inclusion ($inc_{i,j} = 1$)	(1, 6), (7, 11)
Workstation exclusion ($exc_{i,j} = 1$)	(3, 9), (4, 6)
Block exclusion ($exc_b_{i,j} = 1$)	(3, 11)
Workstation eligibility ($tec_{i,k} = 1$)	Task 1 must be performed on workstation 1.

Table 2. Feasible task assignment for the illustrative example.

Task	Block	Workstation
1	1	1
3	1	1
2	2	1
6	3	1
8	4	2
10	4	2
5	5	2
4	7	3
7	8	3
9	8	3
11	8	3

times of blocks based on assigned tasks. Constraint (13) determines if a block is active. Constraint (14) ensures sequential scheduling of blocks on the same workstation. Constraint (15) prevents gaps in block usage on workstations. Constraint (16) binds block start and end times if inactive, making the end time equal to the start time, effectively resulting in zero duration. Constraints (17), (18), (19), (20), and (21) calculate the power demand at different levels (task, block, workstation, system) and determine the PP. Constraint (22) computes the CT as the maximum completion time of any task.

To illustrate the constraint semantics, consider a machining transfer line with $NM = 3$ workstations, $Nb = 3$ blocks per workstation, and $NT = 11$ tasks, yielding a total of $|B| = Nb \cdot NM = 9$ blocks. The block-to-workstation mapping is:

$$R(1) = \{1, 2, 3\}, \quad R(2) = \{4, 5, 6\}, \quad R(3) = \{7, 8, 9\}$$

The maximum number of tasks per block is $Mtb = 3$. The constraints active in this instance are summarised in Table 1.

A feasible assignment satisfying all constraints is presented in Table 2.

The feasibility of this assignment is verified as follows:

- *Precedence*: Task 1 (block 1) precedes tasks 2 (block 2), 3 (block 1), 4 (block 7), and 5 (block 5), since $1 \leq 2$, $1 \leq 1$, $1 \leq 7$, and $1 \leq 5$, respectively. Task 2 (block 2) precedes task 6 (block 3): $2 \leq 3$ ✓. Tasks 3 (block 1), 4 (block 7), and 5 (block 5) all precede task 7 (block 8): $1 \leq 8, 7 \leq 8, 5 \leq 8$ ✓. Task 6 (block 3) precedes task 8 (block 4): $3 \leq 4$ ✓. Task 7 (block 8) precedes task 9 (block 8): $8 \leq 8$ ✓. Task 8 (block 4) precedes task 10 (block 4): $4 \leq 4$ ✓. Tasks 9 and 10 (blocks 8 and 4) precede task 11 (block 8): $8 \leq 8$ and $4 \leq 8$ ✓. Note that when two tasks share the same block, the precedence constraint is satisfied with equality, as the model enforces block-level ordering rather than strict task-level sequencing within a block.
- *Workstation inclusion*: Task 1 (block $1 \in R(1)$) and task 6 (block $3 \in R(1)$) are both assigned to workstation 1 ✓. Task 7 (block $8 \in R(3)$) and task 11 (block $8 \in R(3)$) are both assigned to workstation 3 ✓.
- *Workstation exclusion*: Task 3 (block $1 \in R(1)$, workstation 1) and task 9 (block $8 \in R(3)$, workstation 3) are on different workstations ✓. Task 4 (block $7 \in R(3)$, workstation 3) and task 6 (block $3 \in R(1)$, workstation 1) are on different workstations ✓.
- *Block exclusion*: Task 3 is assigned to block 1 and task 11 to block 8; since $1 \neq 8$, they do not share the same block ✓.
- *Workstation eligibility*: Task 1 is restricted to workstation 1 only ($tec_{1,1} = 1$, $tec_{1,2} = tec_{1,3} = 0$). In the assignment, task 1 is placed in block $1 \in R(1)$, i.e. workstation 1 ✓. All other tasks carry no eligibility restriction and may be assigned to any workstation freely ✓.
- *Block capacity*: Block 1 contains tasks 1 and 3 (2 tasks $\leq Mtb = 3$); block 4 contains tasks 8 and 10 (2 tasks ≤ 3); block 8 contains tasks 7, 9, and 11 (3 tasks $= Mtb = 3$). All remaining active blocks contain one task each ✓.

Despite its effectiveness on small-scale instances, the CP model becomes computationally intractable as problem size grows. The main source of complexity is the coupled assignment-and-scheduling structure of the BSMTL. Each task can potentially be assigned to multiple blocks, represented by the optional interval variables $Y_{i,q}$, while satisfying precedence, inclusion/exclusion, technological eligibility, and block-capacity constraints. As the numbers of tasks, workstations, and blocks increase, the number of feasible task-to-block assignments grows rapidly. Moreover, each feasible assignment induces additional timing decisions for block start and end times, as well as time-indexed power calculations over the planning horizon through $\text{act}_{t,q,i}$, $\text{blk}_{q,t}$, $\text{mach}_{k,t}$, and sys_t. Since the CP model is solved repeatedly within the ε-constraint framework to generate reference Pareto sets, the overall computational effort increases substantially with instance size. As a result, the exact CP approach becomes impractical for medium- and large-scale instances within the imposed time limit, which motivates the development

of the MOVNS-based solution method described in Section 4.

4. Solution method

The proposed NN_MOVNS algorithm builds on the variable neighbourhood search (VNS) framework, which has demonstrated strong and consistent performance in constrained combinatorial optimisation, notably in line balancing (Meng, Tang, and Zhang 2022; Zhao, Tang, and Zhang 2022; Zhao et al. 2024) and scheduling (J.-K. Li et al. 2026; Zhou et al. 2026), outperforming population-based methods on highly structured problem instances. The block-based structure of the BSMTL, combined with its dense constraint landscape covering precedence, inclusion/exclusion, and technological compatibility constraints, makes it particularly well-suited to neighbourhood search, where problem-specific operators can directly exploit the feasibility structure.

The algorithm, described in Algorithm 1, comprises an initialisation stage followed by a main iterative loop that integrates MOVNS-based shaking and local search, DQN-guided operator selection, and a learning mechanism. The initialisation stage generates an initial feasible solution using a constructive heuristic, and concurrently prepares the Pareto archive for storing non-dominated solutions and the DQN agent with its experience replay buffer. The main iterative loop operates until a time-based termination criterion is met. At the start of each iteration, a diversification step periodically reseeds the working solution from the Pareto archive to prevent premature convergence. A shaking operator is then applied to perturb the current solution before local search proceeds. The selection of a local search operator is randomised during an initial warm-up period and is subsequently guided by the DQN agent via an ε-greedy policy based on a five-dimensional state vector. The application of the selected operator yields a new solution, which is then verified for feasibility and evaluated against the two objectives. A multi-component reward function assessing Pareto expansion, diversity improvement, and per-objective gain is computed and, together with the observed state transition, stored in the experience replay buffer (Liu et al. 2025). The Pareto archive is updated accordingly, and the DQN network is periodically retrained via mini-batch sampling from the buffer, allowing the agent to improve its operator selection policy throughout the search. Upon reaching the termination criterion, the algorithm outputs the final Pareto archive. The subsequent sections provide detailed formulations for each of these components.

Algorithm 1: NN_MOVNS

Input: Task blocks $\mathcal{TB}$; constraint matrices pre_{ij}, inc_{ij}, exc_{ij}, exc_b_{ij}, tec_{ik}; time limit $t_{\max}$; Mtb; task times τ; task powers ρ
Output: Pareto archive $\mathcal{P}$
Parameters: w_1=0.8, w_2=0.9, w_3=0.3; $\varepsilon_{\text{init}}$=1.0, $\varepsilon_{\text{post}}$=0.3, $\varepsilon_{\min}$=0.1, δ=0.995; γ =0.9; lr = 0.001; B_s=32; ϕ=0.2; f=5; τ_d=10; A_s=20; A_l=20; ε=10^{-12}

// Initialization
Generate S_0 via RPW constructive heuristic; $(f_1^0, f_2^0) \leftarrow$ Evaluate(S_0)
$\mathcal{P} \leftarrow \{S_0\}$; $S \leftarrow S_0$; $\mathcal{B} \leftarrow \emptyset$; Initialize DQN: multilayer perceptron(5→64→64→8), weights $\boldsymbol{\theta}$, $\varepsilon \leftarrow \varepsilon_{\text{init}}$
iter ← 0; stag ← 0; imp ← 0; warmup ← **true**
while $t_{\text{elapsed}} < t_{\max}$ **do**
  if *warmup* ***and*** $t_{\text{elapsed}} > \phi \cdot t_{\max}$ **then** warmup ← **false**; $\varepsilon \leftarrow \varepsilon_{\text{post}}$
  // Diversification
  if iter mod $\tau_d = 0$ ***and*** $|\mathcal{P}| > 1$ **then** $S \leftarrow$ uniform random sample from $\mathcal{P}$
  // Shaking - SO_1 first, then SO_2 if SO_1 fails; retain S if both fail
  $S_{\text{sh}} \leftarrow$ **null**
  for $SO \in \{SO_1, SO_2\}$ **do**
    for $i \leftarrow 1$ **to** A_s **do if** $SO(S) \neq$ **null then** $S_{\text{sh}} \leftarrow SO(S)$;
    break
    if $S_{\text{sh}} \neq$ **null then break**
  end
  if $S_{\text{sh}} =$ **null then** $S_{\text{sh}} \leftarrow S$
  // State construction & ε-greedy operator selection over $\mathcal{A} = \{N_1, \ldots, N_8\}$
  $D_t \leftarrow$ Diversity($\mathcal{P}$)
  $\mathbf{s}_t \leftarrow [\text{stag}/10,\ \text{imp}/5,\ |\mathcal{P}|/100,\ D_t/1000,\ t_{\text{elapsed}}/t_{\max}]$
  if ¬*warmup* ***and*** iter mod $f = 0$ **then** $\varepsilon \leftarrow \max(\varepsilon_{\min},\ \varepsilon \cdot \delta)$
  if *warmup* ***or*** $\xi \sim \mathcal{U}[0,1] < \varepsilon_t$ **then** $a_t \leftarrow$ uniform random from $\mathcal{A}$
  else $a_t \leftarrow \arg\max_{a \in \mathcal{A}} Q(\mathbf{s}_t, a; \boldsymbol{\theta})$
  // Local search on shaken solution
  $S^* \leftarrow$ **null**; $r_t \leftarrow -0.1$
  for $i \leftarrow 1$ **to** A_l **do if** $N_{a_t}(S_{\text{sh}}) \neq$ **null then** $S^* \leftarrow N_{a_t}(S_{\text{sh}})$;
  break
  // Evaluation, reward & Pareto update
  if $S^* \neq$ **null then**
    $(f_1', f_2') \leftarrow$ Evaluate(S^*); $|\mathcal{P}|_{\text{old}} \leftarrow |\mathcal{P}|$; $D_{\text{old}} \leftarrow D_t$; Compute $f_k^{\min}, f_k^{\max}$ over $\mathcal{P} \cup \{S, S^*\}$ for $k \in \{\text{CT}, \text{PP}\}$
    $\mathcal{P} \leftarrow$ UpdatePareto($\mathcal{P}, S^*$); $D_{t+1} \leftarrow$ Diversity($\mathcal{P}$)
    $r_{\text{p}} \leftarrow \dfrac{|\mathcal{P}| - |\mathcal{P}|_{\text{old}}}{|\mathcal{P}|_{\text{old}} + 1}$; $r_{\text{d}} \leftarrow \dfrac{D_{t+1} - D_{\text{old}}}{D_{\text{old}} + 1}$;
    $r_{\text{o}} \leftarrow \sum_{k \in \{\text{CT},\text{PP}\}} \dfrac{f_k^{\min}}{\max(f_k(S), \varepsilon)} \cdot \dfrac{f_k(S) - f_k(S^*)}{\max(f_k^{\max} - f_k^{\min}, \varepsilon)}$
    $r_t \leftarrow \text{clip}(w_1 r_{\text{p}} + w_2 r_{\text{d}} + w_3 r_{\text{o}},\ -1,\ 1)$
    if $|\mathcal{P}| > |\mathcal{P}|_{\text{old}}$ **then**
      stag ← 0; imp ← imp + 1; $S \leftarrow S^*$
    else
      stag ← stag + 1; imp ← max(0, imp − 1)
    end
  end
  stag ← stag + 1; imp ← max(0, imp − 1)
  // Experience replay & network training
  $D_{t+1} \leftarrow$ Diversity($\mathcal{P}$)
  $\mathbf{s}_{t+1} \leftarrow$ GetState(stag, imp, $|\mathcal{P}|, D_{t+1}, t_{\text{elapsed}}, t_{\max}$); $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})\}$
  if iter mod $f = 0$ ***and*** $|\mathcal{B}| \geq B$ **then**
    Sample $\{(\mathbf{s}_i, a_i, r_i, \mathbf{s}_{i+1})\}_{i=1}^{B_s}$ from $\mathcal{B}$; $y_i \leftarrow r_i + \gamma \cdot \max_{a'} Q(\mathbf{s}_{i+1}, a'; \boldsymbol{\theta})$ $\forall\, i$
    $\mathcal{L}(\boldsymbol{\theta}) \leftarrow \dfrac{1}{B_s} \sum_{i=1}^{B_s} (Q(\mathbf{s}_i, a_i; \boldsymbol{\theta}) - y_i)^2$; Update $\boldsymbol{\theta}$ via Adam on $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
  end
  iter ← iter + 1
end
return $\mathcal{P}$

4.1. Feasible solution generation using constructive heuristic

The constructive heuristic generates an initial feasible solution through a multi-step process that ensures tasks are assigned to blocks on different workstations while respecting all problem constraints. The steps are as follows:

(1) *Task ordering:* In this step, the initial sequence of tasks is determined using the Ranked Positional Weight method (Helgeson and Birnie 1961), which prioritises tasks based on their cumulative positional weights (the sum of a task's processing time and all its successor task times) to establish an efficient preliminary order.
(2) *Initial task assignment:* Distribute tasks to all workstations following the order determined in the previous step. Initially, each workstation is treated as a candidate location for each task, forming a workstation–task eligibility matrix. Subsequent constraint-filtering steps progressively remove infeasible task–workstation combinations before the final unique assignment is constructed.
(3) *Inclusion constraint implementation:* For task pairs requiring co-location, identified by $inc_{ij} > 0$, the heuristic enforces that tasks i and j can only remain feasible for workstations where both tasks can be jointly assigned. If either task is infeasible or unavailable on a workstation, the paired task is also marked infeasible for that workstation.
(4) *Exclusion constraint implementation:* For task pairs prohibited from co-location (identified via exclusion matrix $exc_{ij} > 0$), if both tasks are assigned to the same workstation, retain one task based on prioritisation and mark the other as infeasible (-1) for that workstation. Additionally, mark the retained task as incompatible (-1) on any workstation that initially designates both tasks together, maintaining exclusivity constraints.
(5) *Technological constraint implementation:* For tasks requiring specific workstations (identified via technological matrix $tec_{ik} = 1$), ensure assignment only to technologically capable workstations by removing the task from any incompatible workstation, maintaining technological compatibility requirements.
(6) *Final assignment:* Assigns tasks to workstations to balance workload while satisfying all constraints. A threshold is calculated as total task time divided by workstation count multiplied by the coefficient $\beta = 2.7$. For each workstation (except the last), tasks are assigned from the prioritised list only if prerequisites are completed on the same or earlier workstations. Assignment continues until the threshold is reached. When exceeded, assigned tasks are marked infeasible (-1) on remaining workstations, and unassignable tasks are marked infeasible for the current workstation. Exception: tasks requiring specific workstations are assigned regardless of threshold violations.
(7) *Block creation and task-to-block assignment:* This step groups each workstation's tasks into discrete blocks while satisfying exclusion constraints. The algorithm first generates a set number of empty blocks per workstation (Nb). Then, for each workstation, it iterates through its tasks and tries to assign each one to an existing block. Before placement, it checks the block exclusion matrix (exc_b_{ij}) to ensure no conflicts with tasks already exist in that block. If there is a conflict ($exc_b_{ij} = 1$), the task is skipped for that block. The search continues until a suitable block is found or a new one is created if needed. Each block can contain up to a maximum number of tasks (Mtb), ensuring balanced workload distribution.
(8) *CT calculation:* This step computes the execution times for each workstation and determines the overall CT of the system. For each workstation, the algorithm processes the blocks sequentially, calculating the execution time of each block as the maximum task duration among all tasks within that block (since tasks within a block execute in parallel). The workstation's total CT is computed as the sum of all its block execution times, representing the sequential execution of blocks on that workstation. The algorithm tracks the start and end times for each block to provide detailed timing information. The final system CT is determined as the maximum CT across all workstations, representing the bottleneck workstation that determines the overall system performance. This step provides the foundation for evaluating one of the primary optimisation objectives.
(9) *Power demand calculation:* This step determines the power demand profile for each block across all workstations to support the second optimisation objective. For each workstation, the algorithm processes each block and calculates the total power demand as the sum of power requirements of all tasks within that block. Since tasks within a block execute simultaneously, their power demand are additive. The algorithm handles various edge cases, including empty blocks (assigned zero power demand) and validates the integrity of the task assignments.

For each block, it retrieves the power demand values from the task power matrix and computes the aggregate power requirement. This information is essential for determining the PP across the entire system, which serves as the second objective function in the multi-objective optimisation problem. The power calculation ensures that the solution evaluation considers both temporal and resource constraints effectively.

4.2. Shaking operators

The proposed MOVNS uses two distinct shaking operators (SO), SO_1 and SO_2, to perturb the current solution and help avoid local optima. In the shaking phase, SO_1 is applied first; if it fails to produce a feasible solution within $A_s = 20$ attempts, SO_2 is applied instead with the same attempt limit. If neither operator yields a feasible solution, the algorithm retains the current solution, thereby ensuring that a feasible solution is always maintained.

SO_1 *(Block shift neighbourhood)*. A shifting process takes place wherein a block is randomly selected from a randomly chosen workstation and is then moved to a new position either earlier or later within the same workstation's block sequence. The feasibility of this move depends on ensuring that all constraints, including precedence relationships, are satisfied after the shift.

SO_2 *(Intra-workstation block split)*. A splitting process occurs wherein a block with at least two tasks is randomly selected from a chosen workstation, and a subset of tasks from this block is moved to another block on the same workstation. The target block may be an existing one or an empty block. The move is only feasible if the target block has enough capacity to accommodate the additional tasks and all relevant constraints are satisfied.

4.3. Neighbourhood operators

This section briefly explains the structure of the neighbourhood operators used in the local search phase of NN_MOVNS.

N_1 *(Inter-workstation task swap)*. A swapping process takes place wherein two different workstations are randomly selected, and from each workstation, a non-empty block is chosen at random. Subsequently, a task from each selected block is picked randomly and exchanged between the blocks. The feasibility of this move depends on ensuring that all constraints are satisfied after the swap.

N_2 *(Intra-workstation adjacent block task swap)*. A swapping process occurs wherein a single workstation is randomly selected, and from this workstation, two adjacent non-empty blocks are chosen at random. Subsequently, a task from each of these blocks is picked randomly and exchanged between them. The feasibility of this move depends on ensuring that all constraints are satisfied after the swap.

N_3 *(Task relocation across workstations)*. A moving process takes place wherein two distinct workstations are randomly selected from the set of available workstations. From each workstation, a block is chosen at random, with the source block on the first workstation having at least one task and the target block on the second workstation having available capacity. Subsequently, a task from the source block is randomly picked and moved to the target block on the second workstation. The feasibility of this move depends on satisfying all problem constraints after the task is moved.

N_4 *(Task relocation within a workstation)*. A moving process occurs wherein a single workstation is randomly selected from the set of available workstations. On this workstation, two distinct blocks (not necessarily adjacent) are chosen at random, with the source block having at least one task and the target block having available capacity. Subsequently, a task from the source block is randomly picked and moved to the target block on the same workstation. The feasibility of this move depends on satisfying all problem constraints after the task is moved.

N_5 *(Task insertion across blocks)*. A relocation process takes place wherein a task is randomly selected from a non-empty block on a workstation, and an attempt is made to insert it into blocks on other workstations that have available capacity. The target blocks are considered in a randomised order to promote exploration diversity. The move is accepted only if all problem constraints are satisfied, with the first feasible insertion being accepted.

N_6 *(Task relocation from higher CT to adjacent lower CT workstation)*. A shifting process takes place wherein a pair of adjacent workstations is randomly selected from the set of available workstations, and the workstation with the higher CT is identified as the source, while the other is the target. From the source workstation, the last task of the last non-empty block is chosen, and it is moved to the start of the first block of the target workstation if capacity allows. The feasibility of this move depends on satisfying all the constraints of the problem after the task is shifted.

N_7 *(Task move between adjacent blocks on single workstation)*. A moving process occurs wherein a single workstation is randomly selected from the set of available workstations, and from this workstation, two adjacent blocks are chosen at random, with the source block having at least one task and the target block having available capacity. Subsequently, a task from the source block is randomly picked and moved to the target block on the

same workstation. The feasibility of this move depends on satisfying all problem constraints after the task is moved.

N_8 (Task swap between adjacent workstations). A swapping process takes place wherein two adjacent workstations are randomly selected from the set of available workstations, and from each workstation, a non-empty block is chosen at random. Subsequently, a task from each selected block is picked randomly and exchanged between the blocks. The feasibility of this move depends on ensuring that all constraints are satisfied after the swap.

4.4. Key components of the NN_MOVNS framework

The proposed NN_MOVNS algorithm builds on traditional MOVNS by incorporating a DQN to guide the adaptive selection of neighbourhood operators. Unlike conventional MOVNS frameworks, which apply operators in a fixed or random order, this static approach can limit the algorithm's ability to escape local optima and explore the solution space effectively (Kalatzantonakis, Sifaleras, and Samaras 2025). By using deep reinforcement learning, the proposed method continuously learns which operators perform best, allowing it to adjust dynamically between exploration and exploitation based on accumulated search experience rather than relying on preset rules. This integration of learning mechanisms into neighbourhood search is supported by a growing body of evidence in the scheduling and manufacturing optimisation literature, where reinforcement learning has been shown to improve operator selection in VNS (Kalatzantonakis, Sifaleras, and Samaras 2025; P. Li et al. 2023), graph neural networks have been combined with neighbourhood search for combinatorial scheduling (T. Guo et al. 2025), and deep reinforcement learning has been applied to resilient production control in complex manufacturing environments (Leng et al. 2024; Liu et al. 2025). The key components of the NN_MOVNS framework, as outlined above, are detailed in the following subsections

4.4.1. DQN architecture and state representation

The DQN integrated into NN_MOVNS employs a multilayer perceptron architecture to approximate the action-value function that maps search states to the expected cumulative rewards of each neighbourhood operator (F. Ming et al. 2024). The architecture consists of an input layer that receives five-dimensional state feature vectors, two hidden layers with sixty-four neurons each using rectified linear unit activations, and an output layer with eight neurons representing the Q-values of the eight local search operators. The DQN training configuration was determined through the parameter tuning procedure described in Section 5.2, and the final settings are reported in Table 4. These include the optimiser, learning rate, discount factor, batch size, exploration parameters, and training frequency. Experience replay is used to reduce the correlation between successive samples, while reward clipping bounds the magnitude of temporal-difference updates and improves numerical stability. To examine whether a more advanced deep reinforcement learning architecture was required, preliminary tests were also conducted with a Double DQN variant (Y. Zhang et al. 2023), which decouples action selection from target-value estimation to reduce overestimation bias. Across a representative subset of instances, Double DQN did not produce consistently better Pareto fronts, while increasing the training overhead due to the additional target-network evaluation. Given the compact state representation with five features and the limited action space of eight neighbourhood operators, the overestimation bias addressed by Double DQN was not observed to be a dominant performance-limiting factor. Therefore, the standard DQN architecture was retained as the most parsimonious and computationally efficient alternative.

The state representation encodes five key features that describe the current search status. The stagnation counter, normalised by division by ten, measures the number of consecutive iterations without improvement in the Pareto set, indicating when exploration-oriented operators may be beneficial. The recent improvement metric, normalised by division by five, quantifies the frequency of Pareto set expansions within a recent time window, suggesting periods where exploitation-oriented operators should be prioritised. The Pareto set cardinality, normalised by division by one hundred, reflects the size of the solution archive and the degree of convergence achieved. The diversity measure, normalised by division by one thousand, captures the spread of solutions in objective space through the average pairwise Euclidean distance. Lastly, the search progress, computed as the ratio of elapsed time to the maximum allowed time, enables the network to adjust its behaviour according to the current optimisation phase. The resulting state feature vector is defined as:

$$\mathbf{s}_t = \left[\frac{\text{stagnation}_t}{10}, \frac{\text{improvement}_t}{5}, \frac{|\mathcal{P}_t|}{100}, \frac{D_t}{1000}, \frac{t_{\text{elapsed}}}{t_{\max}} \right]$$

where $\mathcal{P}_t$ denotes the Pareto set at iteration t and D_t represents the diversity metric, computed as the mean pairwise distance between solutions in objective space. This compact representation is designed for the static BSMTL setting considered in this study, where the task-precedence structure and workstation configuration

remain fixed throughout optimisation. In dynamic settings involving real-time disruptions such as machine failures or demand shifts, a graph convolutional network-based state encoding (Leng et al. 2024) could provide a richer representation by capturing the topological structure of the task-precedence graph, enabling the agent to perceive and respond to structural changes during execution.

4.4.2. Action space and neighbourhood operators

The action space in the reinforcement learning framework comprises the eight local search operators over which the DQN agent learns a selection policy. These operators, $\{N_1, N_2, N_3, N_4, N_5, N_6, N_7, N_8\}$, perform small-scale structural adjustments aimed at solution intensification, preserving the integrity of solutions while exploring promising regions near the current search position. The formal action space is defined as:

$$\mathcal{A} = \{N_1,\ N_2,\ N_3,\ N_4,\ N_5,\ N_6,\ N_7,\ N_8\}$$

The two shaking operators, SO_1 and SO_2, are not part of the learned action space. Instead, they are applied as a mandatory perturbation step at every iteration prior to local search, independently of the DQN's decision. This design reflects the classical VNS principle of separating diversification (shaking) from intensification (local search): the DQN governs the intensification phase only, adaptively selecting which local structure to exploit given the current search state, while the shaking phase ensures structural diversity unconditionally.

4.4.3. Multi-component reward function

The reward function in NN_MOVNS integrates multiple performance indicators to comprehensively evaluate operator effectiveness beyond simple measures of solution improvement. The multi-component reward formulation is defined as:

$$r_t = w_1 \cdot r_{\text{Pareto}} + w_2 \cdot r_{\text{diversity}} + w_3 \cdot r_{\text{objective}}$$

The Pareto expansion component assesses an operator's contribution to enlarging the solution archive and is computed as the normalised change in Pareto set cardinality:

$$r_{\text{Pareto}} = \frac{|\mathcal{P}_{t+1}| - |\mathcal{P}_t|}{|\mathcal{P}_t| + 1}$$

The diversity enhancement component promotes operators that increase the spread of solutions along the Pareto front, measured by the normalised change in average pairwise distances:

$$r_{\text{diversity}} = \frac{D_{t+1} - D_t}{D_t + 1}$$

where diversity D_t is calculated as:

$$D_t = \frac{1}{|\mathcal{P}_t|(|\mathcal{P}_t| - 1)/2} \sum_{i=1}^{|\mathcal{P}_t|-1} \sum_{j=i+1}^{|\mathcal{P}_t|} \|\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j)\|_2$$

The objective improvement component rewards operators that enhance solution quality for either objective, expressed as:

$$r_{\text{objective}} = \sum_{k \in \{\text{CT,PP}\}} \frac{f_k^{\min}}{\max\left(f_k(\mathbf{x}_t),\ \varepsilon\right)} \cdot \frac{f_k(\mathbf{x}_t) - f_k(\mathbf{x}_{t+1})}{\max\left(f_k^{\max} - f_k^{\min},\ \varepsilon\right)}$$

where $f_k^{\max}$ and $f_k^{\min}$ denote the extrema computed over $\mathcal{P} \cup \{S, S^*\}$ for objective k, $\varepsilon = 10^{-12}$ prevents division by zero, and the leading scale factor $f_k^{\min} / \max(f_k(\mathbf{x}_t), \varepsilon)$ normalises the gain relative to the current solution's objective value. If no feasible solution is produced by the selected operator, a fixed penalty reward of -0.1 is assigned. The weighting coefficients were set to $w_1 = 0.8$, $w_2 = 0.9$, and $w_3 = 0.3$ based on preliminary computational experiments conducted on a representative subset of instances. Candidate configurations were assessed according to their ability to generate Pareto sets with good convergence and broad coverage of the CT–PP trade-off space. The selected configuration assigns higher weights to r_{Pareto} and $r_{\text{diversity}}$, thereby prioritising archive expansion and solution spread. This reflects the multi-objective purpose of the algorithm: to generate a rich and well-distributed set of non-dominated solutions rather than concentrate the search on a narrow region of the Pareto front.

The lower weight assigned to $r_{\text{objective}}$ retains a direct incentive to improve CT and PP while ensuring that immediate improvement in one objective does not outweigh contributions to Pareto-set expansion and solution diversity. Thus, the selected weights balance archive expansion, front diversity, and direct improvements in the two objectives. Details of the preliminary experiments, including a sensitivity analysis of five alternative reward-weight configurations across five representative instances, are provided in Appendix 2. The results support the selected configuration by showing its consistently strong convergence and Pareto-front coverage across the tested instances. The final reward value is clipped to the interval $[-1, 1]$ to maintain numerical stability and facilitate convergence during DQN training.

4.4.4. Experience replay and network training

The NN_MOVNS algorithm utilises experience replay to improve learning stability and sample efficiency.

A replay buffer with a given capacity stores recent state–action–reward–next state tuples, allowing the decorrelation of training samples and reuse of past experiences (Liu et al. 2025). At periodic intervals, determined by a training frequency parameter set to five iterations, the DQN performs a training step following the procedure outlined below.

A random batch of thirty-two experiences is sampled from the replay buffer, and for each tuple $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$, the temporal-difference target is computed as:

$$y_t = r_t + \gamma \cdot \max_{a' \in \mathcal{A}} Q(\mathbf{s}_{t+1}, a'; \boldsymbol{\theta})$$

The network parameters $\boldsymbol{\theta}$ are updated via gradient descent on the mean squared error loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{B_s} \sum_{i=1}^{B_s} \left(Q(\mathbf{s}_i, a_i; \boldsymbol{\theta}) - y_i \right)^2$$

where B_s denotes the batch size. Gradients are computed through backpropagation, and parameter updates are executed using the Adam optimiser. This periodic training strategy achieves a balance between learning effectiveness and computational efficiency, minimising training overhead while ensuring sufficient network adaptation to capture patterns of operator performance.

4.4.5. Adaptive exploration through warm-up phase and epsilon decay

The NN_MOVNS algorithm incorporates a two-phase exploration strategy comprising an initial warm-up phase followed by adaptive ε-greedy selection. During the warm-up phase, encompassing the first twenty percent of the total time budget, operators are selected uniformly at random to ensure comprehensive exploration of the operator space and accumulation of diverse experiences in the replay buffer. This initialisation strategy prevents premature convergence to suboptimal operator selection policies and establishes a foundation for effective Q-value learning.

Following warm-up completion, the exploration rate is reset to $\varepsilon_{\text{post}} = 0.3$ to initiate exploitation-dominant behaviour, and then undergoes multiplicative decay according to:

$$\varepsilon \leftarrow \max(\varepsilon_{\min}, \varepsilon \cdot \delta), \tag{23}$$

where $\varepsilon_{\min} = 0.1$ and $\delta = 0.995$ is the per-training-step decay factor. Since the network is trained every $f = 5$ iterations, the effective per-iteration decay rate is $\delta^{1/f} \approx 0.999$. This two-step transition, which first resets to $\varepsilon_{\text{post}}$ rather than continuing from the warm-up's uniform selection, ensures a controlled handover from the exploration regime to the learned policy without abrupt discontinuities. The action selection mechanism is formalised as:

$$a_t = \begin{cases} \text{uniform random from } \mathcal{A}, & \text{if } \xi < \varepsilon_t \\ \arg\max_{a \in \mathcal{A}} Q(\mathbf{s}_t, a; \boldsymbol{\theta}), & \text{otherwise} \end{cases} \tag{24}$$

where $\xi \sim \mathcal{U}[0, 1]$ and $\mathcal{A} = \{N_1, \ldots, N_8\}$.

4.4.6. Pareto-based diversification mechanism

Complementing the reinforcement learning-guided operator selection, NN_MOVNS incorporates explicit diversification through periodic Pareto set sampling. At regular intervals of ten iterations, the current working solution is replaced by a randomly selected solution from the Pareto archive, formally expressed as:

$$\begin{aligned} &\text{if } t \bmod \tau_d = 0 \quad \text{and} \quad |\mathcal{P}_t| > 1 : \\ &\quad \mathbf{x}_t \leftarrow \text{uniform random from } \mathcal{P}_t \end{aligned} \tag{25}$$

where $\tau_d = 10$ denotes the diversification interval. This mechanism serves multiple purposes: first, it ensures exploration of diverse regions along the Pareto front rather than concentration in limited areas; second, it provides quality-preserving diversification by maintaining focus on non-dominated solutions; third, it offers a complementary exploration channel independent of the learned operator preferences, thereby mitigating potential biases in the Q-network predictions. This explicit diversification strategy has demonstrated effectiveness in maintaining population diversity and preventing premature convergence in multi-objective optimisation contexts.

5. Experimental evaluation

This section presents the experimental evaluation of the proposed NN_MOVNS algorithm, covering the computational environment, experimental design, parameter tuning, and the final parameter configuration. All experiments were executed on a workstation equipped with a 12th Generation Intel Core i9-12900 processor operating at 2.40 GHz, 128 GB of RAM, and a 64-bit Windows operating system with an x64-based processor architecture. All algorithms were implemented in Python 3.12. Numerical computations utilised NumPy 2.3.0 for array operations and Pandas 2.3.3 for data handling and statistical analysis. The neural network component of NN_MOVNS was developed in PyTorch 2.8.0, leveraging its automatic differentiation and efficient tensor operations for the DQN implementation. Performance indicators, including hypervolume and inverted generational distance plus (IGD^+), were computed using the pymoo 0.6.1 multi-objective optimisation framework. Additional scientific computations employed SciPy

Table 3. Summary characteristics of test instance families.

Group	Family	Instances	Tasks	Workstations	Blocks
Small	JACKSON	T1–T5	11	3	3–4
	MITCHELL	T6–T10	21	4	3–4
	BUXEY	T11–T15	29	5	3–6
	GUNTHER	T16–T20	35	5	4–6
Medium	KILBRID	T21–T25	45	5	4–6
	WARNECKE	T26–T30	58	6	5–6
	TONGE70	T31–T35	70	7	5–6
	WEE-MAG	T36–T40	75	7	6–7
Large	LUTZ3	T41–T45	89	8	7–8
	MUKHERJEE	T46–T50	94	9	7–8
	BARTHOLD	T51–T55	148	10	8–10

1.14.1 for spatial calculations and Matplotlib 3.9.2 for visualisation. The mathematical model was solved using CPLEX 12.10.

5.1. Experimental design

To evaluate the performance of the proposed algorithms, a diverse set of test instances derived from 11 established line balancing studies is used. The precedence relationships (pre_{ij}) and task times (Ta_i) were sourced from established online datasets (Scholl et al. 1995), ensuring the authenticity and relevance of the input data. Since the original datasets do not report power demand for individual tasks, these values were estimated based on technical specifications from manufacturers' catalogs for industrial multi-spindle machines. The power demand of individual tasks was assumed to lie within the range of 8–35 kW, reflecting typical industrial values. The dataset includes 55 instances varying in size, structure, and computational difficulty, grouped into small (4 families), medium (4 families), and large (3 families) categories based on problem dimensions. Table 3 summarises the characteristics of each problem family, including instance ranges, the number of tasks, workstations, and blocks.

The small-scale group (JACKSON, MITCHELL, BUXEY, GUNTHER) includes 20 instances with 11–35 tasks and 3–5 workstations. These compact problems can be feasibly solved by CP, serving as a baseline for validating metaheuristic performance. The medium-scale group (KILBRID, WARNECKE, TONGE70, WEE-MAG) contains 20 instances with 45–75 tasks and 5–7 workstations. Only KILBRID remains solvable by CP, while the others exceed its computational limits, emphasising the growing challenge for exact methods. The large-scale group (LUTZ3, MUKHERJEE, BARTHOLD) includes 15 instances with 89–148 tasks and 8–10 machines, representing industrial-scale problems where CP becomes intractable. These are solved exclusively by metaheuristic approaches, with MOVNS and NN_MOVNS demonstrating the strongest performance.

The detailed characteristics of all 55 test instances, including the number of tasks, workstations, blocks, maximum tasks per block, inclusion rate, exclusion rate, block exclusion rate, and technological ratio, are publicly available at the following link: https://figshare.com/s/9db5b6bfd7ba7212473b.

5.1.1. Competing algorithms and implementation

To evaluate the proposed NN_MOVNS, we compare it with four competing approaches: the CP model described in Section 3.1, the baseline MOVNS, and two representative population-based multi-objective metaheuristics, namely NSGA-II (Fang et al. 2020; Petersen et al. 2024) and multi-objective particle swarm optimisation (MOPSO) (Destouet et al. 2024; L. Guo et al. 2025). These two algorithms are included to benchmark NN_MOVNS against established population-based search paradigms that are commonly used in multi-objective manufacturing and energy-aware scheduling problems. NSGA-II represents an evolutionary search framework based on non-dominated sorting, elitist selection, crossover, and mutation, whereas MOPSO represents a swarm-intelligence framework in which solutions are updated through individual and collective learning mechanisms. Their inclusion therefore provides a meaningful comparison with algorithms that differ substantially from VNS in both population management and search dynamics, allowing the effectiveness of the proposed learning-guided neighbourhood search to be assessed against recognised multi-objective optimisation baselines.

All four metaheuristics are implemented using the same block-based solution representation, objective evaluation procedures, constraint-checking mechanisms, and computational-time stopping criterion. This common implementation framework ensures that the comparison focuses on the search behaviour of the algorithms rather than differences in representation, evaluation, or feasibility handling.

The time limit for each metaheuristic run is calculated as follows:

$$T = \min\left(\max\left(T_{\text{cap}} \cdot \left(\frac{NT}{NT_{\max}}\right)^{a} \cdot \left(\frac{NM}{N_{\max}}\right)^{b}, T_{\min}\right), T_{\text{cap}}\right) \tag{26}$$

where $a = 2.0$, $b = 1.0$, $T_{\min} = 120\,\text{s}$, $T_{\text{cap}} = 1200\,\text{s}$, $NT_{\max} = 148$, and $N_{\max} = 10$. The formula assigns longer time limits to larger and more difficult instances, while keeping the runtime within a fixed lower and upper bound. All metaheuristics terminate once the elapsed time reaches T.

Table 4. Consolidated parameter settings for all metaheuristics.

Algorithm	Parameter	Level 1	Level 2	Level 3	Tuned Value	Source
NN_MOVNS	Learning rate (Adam)	0.0005	0.001	0.002	0.001	Taguchi
	$\varepsilon_{\text{init}}$	0.5	0.75	1.0	1.0	Taguchi
	$\varepsilon_{\min}$	0.05	0.1	0.15	0.1	Taguchi
	Decay factor δ	0.985	0.995	1.0	0.995	Taguchi
	Discount factor γ	0.8	0.9	0.95	0.9	Taguchi
	Replay buffer capacity	5 000	10 000	15 000	10 000	Taguchi
	Batch size B_s	22	32	42	32	Taguchi
MOVNS	Shaking neighbourhood size $k_{\max}$	3	5	7	5	Taguchi
	Local search iterations n_{ls}	10	20	30	20	Taguchi
	Perturbation strength ρ	0.1	0.2	0.3	0.2	Taguchi
NSGA-II	Population size	50	100	150	100	Taguchi
	Crossover probability p_c	0.85	0.90	0.95	0.95	Taguchi
	Mutation probability p_m	0.85	0.90	0.95	0.95	Taguchi
MOPSO	Swarm size	60	80	100	80	Taguchi
	Cognitive coefficient c_1	1.0	1.5	2.0	1.5	Taguchi
	Social coefficient c_2	1.5	2.0	2.5	2.0	Taguchi
	Turbulence mutation rate	0.20	0.30	0.40	0.30	Taguchi
Fixed parameters						
Shaking attempts A_s			–		20	Fixed
Local search attempts A_l			–		20	Fixed
Diversification interval τ_d			–		10	Fixed
Time limit T			–		*Equation* (26)	Instance-dependent

NSGA-II additional fixed parameters: repair attempts per offspring = 5.
MOPSO additional fixed parameters: inertia weight 0.9→0.4 linearly; stagnation injection after 8 iterations.

Each metaheuristic is executed five times independently for each instance using different random seeds. The non-dominated solutions obtained from the five runs are then pooled, and the resulting pooled front is used for the performance indicator calculations.

The implementation details of the four metaheuristics are summarised below. All tuned and fixed parameter values are consolidated in Table 4.

5.1.1.1. NN_MOVNS (proposed). NN_MOVNS is the learning-guided VNS algorithm described in Section 4. At each iteration a shaking operator (SO_1 or SO_2) diversifies the current solution, after which one of the eight local search operators N_1–N_8 is applied. The operator is selected by a DQN-guided ε-greedy policy. A warm-up phase covering 20% of the runtime precedes the learning-guided phase, during which operators are selected uniformly at random.

5.1.1.2. MOVNS. MOVNS serves as the baseline VNS method. It employs the same shaking and neighbourhood operators as NN_MOVNS but does not use the DQN-guided operator selection mechanism. This comparison therefore isolates the contribution of the learning component.

5.1.1.3. NSGA-II. The NSGA-II implementation uses non-dominated sorting and crowding-distance selection (Petersen et al. 2024). Offspring are generated via two crossover operators (workstation-level and task-level) and five mutation move types (task relocation across workstations, task swap between blocks, within-workstation block reordering, block movement between workstations, and multi-move perturbation). Because the BSMTL is highly constrained, infeasible offspring are passed through a repair procedure. If repair fails, the parent solution is retained.

5.1.1.4. MOPSO. The MOPSO implementation maintains an external archive of non-dominated solutions. At each update, a guiding solution is selected from this archive using crowding distance (L. Guo et al. 2025). Since the BSMTL is a discrete problem, the standard continuous velocity update is replaced by a discrete mechanism: the inertia component applies a random mutation, the cognitive component moves the particle toward its personal best solution, and the social component moves it toward the selected archive solution. Infeasible solutions produced by these updates undergo the same repair procedure as in NSGA-II. To counteract stagnation, turbulence mutation and periodic swarm reseeding are applied.

5.1.1.5. Computational effort. Although all four metaheuristics share the same instance-dependent time limit, the single solution-based and population-based methods allocate this budget differently. NN_MOVNS and MOVNS follow a sequential iterate-and-improve scheme: each iteration perturbs the incumbent solution, applies a single neighbourhood operator, checks feasibility, and updates the approximation set. NSGA-II and MOPSO instead process entire populations at each generation, applying crossover, mutation, or velocity updates

to multiple individuals before selection or archive update. Because the two paradigms convert the same time budget into search effort through fundamentally different mechanisms, comparing raw operation counts across them is neither meaningful nor informative. The experimental comparison is therefore based on the quality of the Pareto fronts obtained within the common time limit, providing a fair basis for assessing algorithms with different search structures.

5.1.2. Performance metrics

A rigorous and multi-faceted evaluation of algorithmic performance is conducted using three complementary quality indicators (Beldar et al. 2026): the relative hypervolume (HV_{rel}), the inverted generational distance plus (IGD^+), and the coverage metric. This set of indicators enables a holistic assessment of convergence quality, solution diversity, and pairwise dominance relationships within the generated Pareto fronts.

The HV_{rel} indicator quantifies the volume of the objective space dominated by a solution set relative to a reference point, thereby capturing both convergence and spread. The IGD^+ metric calculates the average distance from points on the reference Pareto front to their nearest solution in the obtained approximation front, measuring how well the generated front covers and conforms to the best-known trade-off surface. Finally, the coverage metric provides a direct pairwise comparison between two solution sets by quantifying the fraction of solutions in one set that are dominated by solutions in the other. Together, these three indicators provide a comprehensive evaluation of each algorithm's performance.

5.2. Parameter tuning

The performance of metaheuristic algorithms is sensitive to their control parameters. Therefore, the Taguchi method (Mahmoodi et al. 2024) was employed to configure the parameters of all four metaheuristics. An orthogonal array design was used to explore each algorithm's parameter space efficiently, reducing the number of experimental runs while capturing the main effects of individual factors. For all algorithms, a representative subset of 15 instances covering the three size categories (small, medium, and large) was used as the tuning testbed, with the mean IGD^+ value as the response variable and the signal-to-noise (SN) ratio under the 'smaller is better' criterion used to identify robust settings.

For NN_MOVNS, an L_{18} orthogonal array with seven factors at three levels was employed. The tuned factors are the learning rate, the initial and minimum ε-greedy values, the decay factor, the discount factor, the replay buffer capacity, and the batch size. Figure 2 shows the corresponding main effects plots. In these plots, red circles indicate the selected tuned values, while dashed lines indicate the overall mean. For most factors, both the mean response and SN ratio led to the same preferred level. The same Taguchi procedure was applied to MOVNS, NSGA-II, and MOPSO. For brevity, only the NN_MOVNS plots are reported. Table 4 consolidates all tuned and fixed parameter values.

In addition to the Taguchi-tuned parameters, several parameters were fixed based on preliminary experimentation or kept constant across all algorithms to ensure a fair comparison.

6. Comparative performance analysis

This section provides a rigorous comparative evaluation of algorithm performance based on pooled Pareto fronts. The analysis is organised in three stages. First, Section 6.2 compares NN_MOVNS against the exact CP solver on the instances where CP produces feasible solutions. Second, Section 6.3 extends the comparison to include all four metaheuristics across all 55 instances. Third, Section 6.4 presents statistical significance tests and an overall algorithm ranking. Throughout this section, the instance grouping follows Table 3: Small (T1–T20, families JACKSON through GUNTHER, 20 instances), Medium (T21–T40, families KILBRID through WEE-MAG, 20 instances), and Large (T41–T55, families LUTZ3 through BARTHOLD, 15 instances).

6.1. Pooled-front evaluation strategy

For each test instance, each metaheuristic was executed five times independently with different random seeds. The results were aggregated into a single pooled approximation set per algorithm: $A_{\text{pool}} = ND(A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5)$ denotes the nondominated subset. CP produced a single approximation set per instance via the bidirectional ε-constraint method. An empirical reference front was then constructed for each instance as: $\mathcal{P}^* = ND(\bigcup_{\text{all algorithms}} A_{\text{pool}})$ including CP solutions where available. All indicators, HV_{rel} (higher is better), IGD^+ (lower is better), and the pairwise coverage metric C, are computed with respect to this common reference front. For the coverage metric, $C(A, B)$ measures the fraction of solutions in B that are weakly dominated by at least one solution in A. An algorithm is considered superior on a given instance in terms of coverage when $C(A, B) > C(B, A)$.

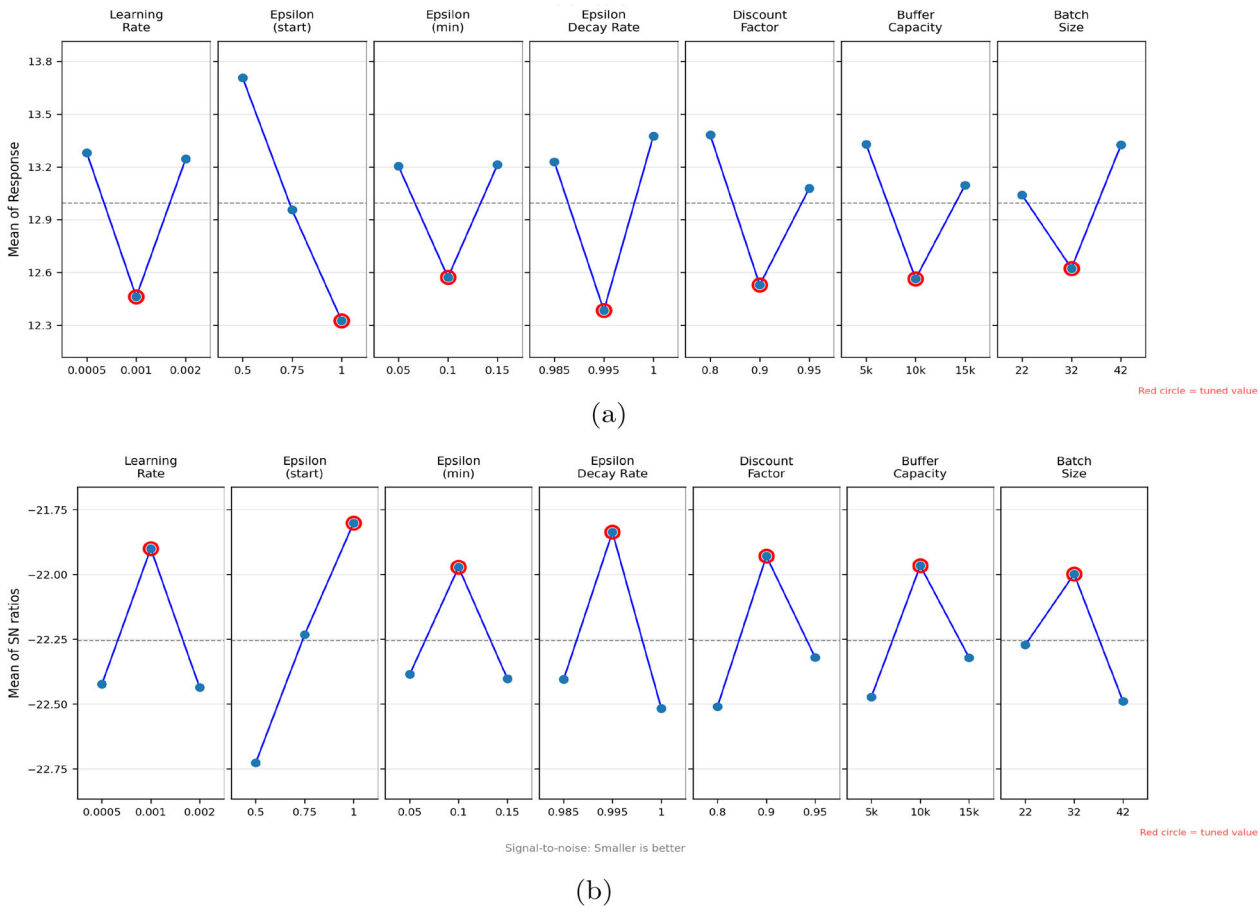

Figure 2. Taguchi main effects plots for the seven NN_MOVNS hyperparameters. (a) Main effects plot for means and (b) Main effects plot for SN ratios (smaller is better).

6.2. Comparison with the exact solver

The CP model was solved using IBM CP Optimiser via the bidirectional ε-constraint method. A fixed time budget was assigned to each ε-constraint subproblem depending on the benchmark family: 300 s for JACKSON and MITCHELL, 600 s for BUXEY 1800 s for GUNTHER, and 3600 s for KILBRID. For the JACKSON and MITCHELL families, all subproblems were solved to optimality, so the CP front constitutes the exact Pareto front for those instances. For the remaining families, optimality was not always certified within the time limit. For all instances beyond the Small and KILBRID groups (T26–T55), CP failed to produce any feasible solution within the imposed time limits, confirming the scalability limitations of exact methods and motivating the metaheuristic approach.

For NN_MOVNS, the running time was determined adaptively according to Equation (26), yielding a size-scaled time budget with a lower bound of 120 s and an upper bound of 1200 s. These computational budgets are not identical but were selected according to the intended behaviour of each method: CP was given a family-dependent limit for each ε-subproblem, whereas NN_MOVNS was given an instance-dependent global limit scaled by the number of tasks and workstations. CP produced feasible solutions for 25 of the 55 test instances: all 20 Small instances (T1–T20) and the 5 KILBRID instances (T21–T25) within the Medium group. To analyse the impact of problem size, the CP comparison is partitioned into a Small group (JACKSON, MITCHELL, BUXEY, GUNTHER; 20 instances) and a Medium, KILBRID group (5 instances). Table 5 presents the pooled-front indicator values for CP and NN_MOVNS on the 20 Small instances, while Table 6 reports the corresponding results for the five KILBRID instances.

For the Small instances, CP performs better on HV_{rel} with 14 wins against 6, and on IGD^{+} with 12 wins against 8. The mean HV_{rel} favours CP (0.960 versus 0.925). However, NN_MOVNS consistently generates larger pooled Pareto sets (mean $|\mathcal{P}| = 8.9$ versus 6.2), providing decision-makers with richer trade-off information. Notably, on the GUNTHER family (T16–T20), where optimality is not always certified within the CP time limit, NN_MOVNS achieves superior HV_{rel} in

Table 5. Pooled-front comparison between CP and NN_MOVNS on Small instances (T1–T20).

		HV_{rel}		IGD^+		$\lvert\mathcal{P}\rvert$	
Inst.	Family	CP	NN_MOVNS	CP	NN_MOVNS	CP	NN_MOVNS
T1	JACKSON	0.976	0.987	0.889	0.444	5	7
T2	JACKSON	0.978	0.957	0.900	2.073	6	8
T3	JACKSON	0.986	0.990	0.556	1.000	5	7
T4	JACKSON	0.966	0.988	1.300	1.100	6	8
T5	JACKSON	1.000	0.908	0.000	1.800	5	3
T6	MITCHELL	0.978	0.969	1.200	1.410	5	11
T7	MITCHELL	1.000	0.820	0.000	3.977	7	7
T8	MITCHELL	0.984	0.854	0.889	2.667	6	7
T9	MITCHELL	0.991	0.836	0.778	3.556	6	6
T10	MITCHELL	0.998	0.922	0.333	2.373	6	9
T11	BUXEY	0.995	0.950	0.875	3.236	4	8
T12	BUXEY	0.982	0.937	2.545	4.818	5	9
T13	BUXEY	0.970	0.833	3.111	5.205	9	10
T14	BUXEY	0.942	0.918	3.809	2.860	8	13
T15	BUXEY	0.964	0.895	6.340	2.374	6	17
T16	GUNTHER	0.981	0.948	1.970	4.410	9	9
T17	GUNTHER	0.848	0.978	6.172	2.931	7	10
T18	GUNTHER	0.970	0.892	3.867	5.478	6	10
T19	GUNTHER	0.865	0.992	21.404	1.021	5	11
T20	GUNTHER	0.834	0.933	17.604	4.808	6	9
Mean		0.960	0.925	3.677	2.887	6.2	8.9
Wins		14	6	12	8	2	17

Table 6. Pooled-front comparison between CP and NN_MOVNS on KILBRID instances (T21–T25).

		HV_{rel}		IGD^+		$\lvert\mathcal{P}\rvert$	
Inst.	Family	CP	NN_MOVNS	CP	NN_MOVNS	CP	NN_MOVNS
T21	KILBRID	0.999	0.874	0.200	6.468	9	10
T22	KILBRID	0.984	0.993	2.875	2.129	6	17
T23	KILBRID	0.919	0.985	7.492	2.300	5	9
T24	KILBRID	0.805	0.834	7.436	6.368	6	11
T25	KILBRID	0.824	0.956	18.991	1.373	4	15
Mean		0.906	0.928	7.399	3.728	6.0	12.4
Wins		1	4	1	4	0	5

three of five instances (T17, T19, T20) and substantially lower IGD^+ values, demonstrating that the metaheuristic becomes competitive as instances approach the tractability boundary of the exact method.

In contrast, the KILBRID instances (Table 6) show a clear advantage for NN_MOVNS. It outperforms CP in 4 of 5 instances on both HV_{rel} and IGD^+, with a higher mean HV_{rel} (0.928 versus 0.906) and a lower mean IGD^+ (3.728 versus 7.399). Only T21 favours CP, where its HV_{rel} of 0.999 and IGD^+ of 0.200 indicate a near-optimal front. The pooled NN_MOVNS Pareto sets are also larger (mean $|\mathcal{P}| = 12.4$ versus 6.0), reflecting its ability to discover more diverse trade-off solutions. These results confirm that, as instance size increases and the exact method approaches its tractability limits even with extended time budgets (3600 s per ε-subproblem), NN_MOVNS delivers superior approximation quality. Table 7 summarises the grouped comparison across both groups.

6.3. Extended comparison across all metaheuristics

This section extends the computational comparison by evaluating the proposed NN_MOVNS against three competing metaheuristics: MOVNS, NSGA-II, and MOPSO. All four use the common experimental setup described in Section 5.1.1, and the pooled-front strategy is applied consistently across all algorithms when computing the performance indicators.

6.3.1. Performance by problem scale

Table 8 shows the mean indicator values obtained from the pooled Pareto fronts for each problem scale. CP is reported only for the Small instances, where all 20 instances were solved. For the Medium group, CP solved only the five KILBRID instances and is therefore excluded from the group-level summary.

The results show that the relative performance of the algorithms changes with problem scale. On Small instances, CP obtains the highest mean HV_{rel} (0.960), as expected for these tractable instances. Among the metaheuristics, NN_MOVNS performs best with mean HV_{rel} of 0.925 and the lowest mean IGD^+ of 2.887 across all methods including CP, suggesting that the pooled metaheuristic front discovers solutions the ε-constraint grid misses on the harder BUXEY and GUNTHER families. The advantage of NN_MOVNS becomes more pronounced on Medium instances, where its mean IGD^+ is 3.560, compared with 10.780 for MOVNS, 67.980 for MOPSO, and 121.300 for NSGA-II. On Large instances, the performance gap widens further: NN_MOVNS achieves a mean IGD^+ of 2.640, compared with 22.790 for MOVNS, an 88.4% reduction. NSGA-II and MOPSO deteriorate to mean IGD^+ values of 526.100 and 301.500, respectively.

6.3.2. Overall pairwise comparison

Table 9 presents the pairwise comparison between NN_MOVNS and each competing metaheuristic across all 55 instances using the pooled Pareto fronts. For each indicator and algorithm pair, an instance is counted as a 'win' for the algorithm that obtains the strictly better value. 'Ties' correspond to instances in which neither algorithm obtains a strictly better value under the adopted pairwise comparison criterion.

The results show that NN_MOVNS consistently outperforms the competing metaheuristics in the pairwise comparisons. Against MOVNS, NN_MOVNS obtains better HV_{rel} values in 66.0% of the non-tied comparisons and better IGD^+ values in 72.0% of the cases. It also achieves a coverage advantage in 76.0% of the instances. Since MOVNS uses the same VNS framework but does not include the learning component, these results indicate that the DQN-guided operator selection improves

Table 7. Grouped pooled comparison between CP and NN_MOVNS.

Group	Metric	NN_MOVNS wins	CP wins	NN_MOVNS mean	CP mean	Direction
Small (T1–T20)	HV_{rel}	6	14	0.925	0.960	Higher better
Small (T1–T20)	IGD^+	8	12	2.887	3.677	Lower better
Small (T1–T20)	$\lvert\mathcal{P}\rvert$	17	2	8.9	6.2	Higher better
KILBRID (T21–T25)	HV_{rel}	4	1	0.928	0.906	Higher better
KILBRID (T21–T25)	IGD^+	4	1	3.728	7.399	Lower better
KILBRID (T21–T25)	$\lvert\mathcal{P}\rvert$	5	0	12.4	6.0	Higher better

Table 8. Mean indicator values by problem scale on pooled Pareto fronts.

Scale	Indicator	NN_MOVNS	MOVNS	NSGA-II	MOPSO	CP
Small$T1–T20)	HV_{rel}	0.925	0.907	0.603	0.682	0.960
	IGD^+	2.887	3.877	63.920	34.180	3.677
	$\lvert\mathcal{P}\rvert$	8.9	8.2	4.3	4.1	6.2
Medium$T21–T40)	HV_{rel}	0.949	0.904	0.421	0.626	–[†]
	IGD^+	3.560	10.780	121.300	67.980	–
	$\lvert\mathcal{P}\rvert$	10.4	8.6	5.4	6.5	–
Large$T41–T55)	HV_{rel}	0.969	0.942	0.262	0.499	–
	IGD^+	2.640	22.790	526.100	301.500	–
	$\lvert\mathcal{P}\rvert$	6.5	6.1	5.0	3.1	–

[†]CP solves KILBRID (T21–T25) only; see Table 6.

Table 9. Pairwise comparison of NN_MOVNS against all metaheuristics on pooled Pareto fronts across all 55 instances.

Comparison	Indicator	NN_MOVNS wins	Competitor wins	Ties	NN_MOVNS (%)
vs. MOVNS	HV_{rel}	33	17	5	66.0
	IGD^+	36	14	5	72.0
	Coverage advantage	38	12	5	76.0
vs. NSGA-II	HV_{rel}	47	3	5	94.0
	IGD^+	48	2	5	96.0
	Coverage advantage	47	3	5	94.0
vs. MOPSO	HV_{rel}	46	4	5	92.0
	IGD^+	46	4	5	92.0
	Coverage advantage	45	3	7	93.8

the effectiveness of the search over the static operator selection strategy.

The advantage of NN_MOVNS is more pronounced when compared with the two population-based algorithms. Against NSGA-II, NN_MOVNS wins 94.0% of the HV_{rel} comparisons and 96.0% of the IGD^+ comparisons. Although NSGA-II can generate diverse populations, its crossover and mutation operators often produce infeasible offspring in the highly constrained BSMTL setting. These offspring require repair operations, which consume part of the computational budget and do not always lead to improved solutions. In contrast, NN_MOVNS relies on specialised neighbourhood operators, including N_1–N_8 and SO_1–SO_2, which are designed to perform targeted task-level and block-level modifications while preserving feasibility more effectively.

A similar pattern is observed against MOPSO. NN_MOVNS wins 92.0% of the comparisons for both HV_{rel} and IGD^+, and achieves a coverage advantage in 93.8% of the non-tied cases. MOPSO is less effective in this discrete and highly constrained search space because its velocity-based search mechanism must be adapted through discrete task relocation toward personal and global best positions. This limits its ability to perform precise, constraint-respecting moves. Overall, the pairwise results confirm that the block-based VNS structure is more suitable for the BSMTL than the population-based search mechanisms, and that the learning-guided operator selection further strengthens this advantage.

Table 10. Mean pairwise coverage $C(A, B)$ among the four metaheuristics across all 55 instances.

$C(A \rightarrow B)$	NN_MOVNS	MOVNS	NSGA-II	MOPSO
NN_MOVNS	–	0.54	0.87	0.73
MOVNS	0.27	–	0.84	0.62
NSGA-II	0.04	0.06	–	0.41
MOPSO	0.07	0.10	0.39	–

6.3.3. Coverage analysis

Table 10 presents the mean pairwise coverage values among the four metaheuristics. The coverage value $C(A, B)$ measures the fraction of solutions generated by algorithm B that are dominated by at least one solution generated by algorithm A. Therefore, higher values indicate that the row algorithm provides stronger solutions than the column algorithm.

The results show a clear advantage for the VNS-based methods, especially NN_MOVNS. On average, NN_MOVNS dominates 54% of the solutions produced by MOVNS, whereas MOVNS dominates only 27% of the solutions produced by NN_MOVNS. This difference is important because both algorithms use the same neighbourhood structure. The main difference is the DQN-guided operator selection in NN_MOVNS, which helps the search move toward better trade-off regions.

The advantage of NN_MOVNS is even stronger when compared with the population-based algorithms. NN_MOVNS dominates 87% of the solutions generated by NSGA-II and 73% of those generated by MOPSO. In contrast, NSGA-II and MOPSO dominate only a small fraction of the NN_MOVNS solutions, with coverage values of 0.04 and 0.07, respectively. This indicates that the solutions produced by NN_MOVNS are usually closer to the preferred Pareto front than those produced by the population-based methods.

Table 11. Wilcoxon signed-rank test results with Holm–Bonferroni correction.

Comparison	Indicator	p-value	Cliff's δ	N	Significant?
MOVNS vs. NN_MOVNS	HV_{rel}	0.012	−0.273	55	Yes
	IGD^+	<0.001	+0.517	55	Yes
MOPSO vs. NN_MOVNS	HV_{rel}	<0.001	−0.990	55	Yes
	IGD^+	<0.001	+0.968	55	Yes
MOPSO vs. MOVNS	HV_{rel}	<0.001	−0.967	55	Yes
	IGD^+	<0.001	+0.800	55	Yes
NSGA-II vs. NN_MOVNS	HV_{rel}	<0.001	−0.987	55	Yes
	IGD^+	<0.001	+0.910	55	Yes
NSGA-II vs. MOVNS	HV_{rel}	<0.001	−0.970	55	Yes
	IGD^+	<0.001	+0.797	55	Yes
NSGA-II vs. MOPSO	HV_{rel}	0.575	+0.017	55	No
	IGD^+	0.575	−0.025	55	No

Effect-size thresholds for $|\delta|$: < 0.147 negligible, < 0.33 small, < 0.474 medium, ≥ 0.474 large.

MOVNS also performs well against NSGA-II and MOPSO, with coverage values of 0.84 and 0.62, respectively. This further confirms that the VNS framework is well suited to the constrained, block-based structure of the BSMTL. Finally, the coverage values between NSGA-II and MOPSO are relatively similar, with NSGA-II dominating 41% of MOPSO solutions and MOPSO dominating 39% of NSGA-II solutions. This suggests that the two population-based algorithms perform at a broadly comparable level on this problem.

6.4. Statistical significance and overall ranking

Wilcoxon signed-rank tests (Osama et al. 2026) are conducted at the 5% significance level with Holm–Bonferroni correction across all 55 instances. Cliff's δ is considered as a non-parametric effect-size measure to quantify the magnitude and direction of the observed differences. The effect size is interpreted as negligible for $|\delta| < 0.147$, small for $|\delta| < 0.33$, medium for $|\delta| < 0.474$, and large for $|\delta| \geq 0.474$. The tests are based on the pooled-front indicator values across all instances. Table 11 shows the main pairwise comparisons.

NN_MOVNS significantly outperforms MOVNS on both indicators. For IGD^+, the improvement is particularly strong, with $p < 0.001$ and a large effect size of $|\delta| = 0.517$. This indicates that the DQN-guided operator selection substantially improves the ability of the search to approach the reference front. The improvement in HV_{rel} is also statistically significant, with $p = 0.012$, although the corresponding effect size is small ($|\delta| = 0.273$). Thus, NN_MOVNS provides a statistically supported improvement over the MOVNS baseline.

Both VNS-based algorithms also significantly outperform the two population-based algorithms. NN_MOVNS and MOVNS are superior to both NSGA-II and MOPSO for HV_{rel} and IGD^+, with $p < 0.001$ in all corresponding comparisons. The associated effect sizes are large in most cases, reaching $|\delta| = 0.990$ for MOPSO versus NN_MOVNS on HV_{rel} and $|\delta| = 0.970$ for NSGA-II versus MOVNS on HV_{rel}. These results confirm that the VNS framework, even without the DQN enhancement, is much better suited to the constrained, block-based structure of the BSMTL than the population-based search strategies. In contrast, NSGA-II and MOPSO show no statistically significant difference from each other on either indicator, with $p = 0.575$ and negligible effect sizes. Overall, the statistical tests support the ranking NN_MOVNS first, MOVNS second, and NSGA-II and MOPSO at a comparable lower level.

Table 12. Overall algorithm ranking across all 55 instances.

Algorithm	Mean HV_{rel}	Mean IGD^+	Mean $\lvert\mathcal{P}\rvert$	Win %	Rank
NN_MOVNS	0.946	3.020	8.6	56.4	1
MOVNS	0.916	11.760	7.5	30.9	2
MOPSO	0.612	119.000	4.7	1.8	3
NSGA-II	0.443	206.100	4.8	1.8	4

CP is excluded because it is tractable on only 25 of 55 instances.

Table 12 confirms this ranking from the aggregate performance indicators, establishing NN_MOVNS $\succ$ MOVNS $\succ$ {MOPSO $\approx$ NSGA-II}. Two effects drive it: the VNS framework already outperforms both population-based methods even without learning, and the DQN-guided operator selection further improves this baseline, most clearly on IGD^+.

6.4.1. Pareto front visualisation

To complement the numerical comparison, Figure 3 shows the pooled Pareto fronts for four representative instances spanning the three problem scales: T4 (JACKSON, small, 11 tasks), T22 (KILBRID, medium, 45 tasks), T34 (TONGE70, medium, 70 tasks), and T55 (BARTHOLD, large, 148 tasks). Filled markers denote the non-dominated solutions in each algorithm's pooled front, while open markers denote dominated solutions obtained in individual runs. Across all four instances, NN_MOVNS produces the best converged and most evenly distributed front. On the small instance T4 it lies on or very near the CP reference frontier, and as problem size grows it dominates the competitors by progressively wider margins, maintaining consistently low PP values across a broad CT range. MOVNS is consistently the second best method, covering a similar region but with a less dense front and several solutions dominated by NN_MOVNS. MOPSO returns only sparse clusters that miss the best trade-off regions, and NSGA-II produces few feasible solutions and remains far from the efficient frontier. The widening gap with instance size is fully consistent with the quantitative results and confirms the benefit of combining the VNS framework with DQN-guided operator selection.

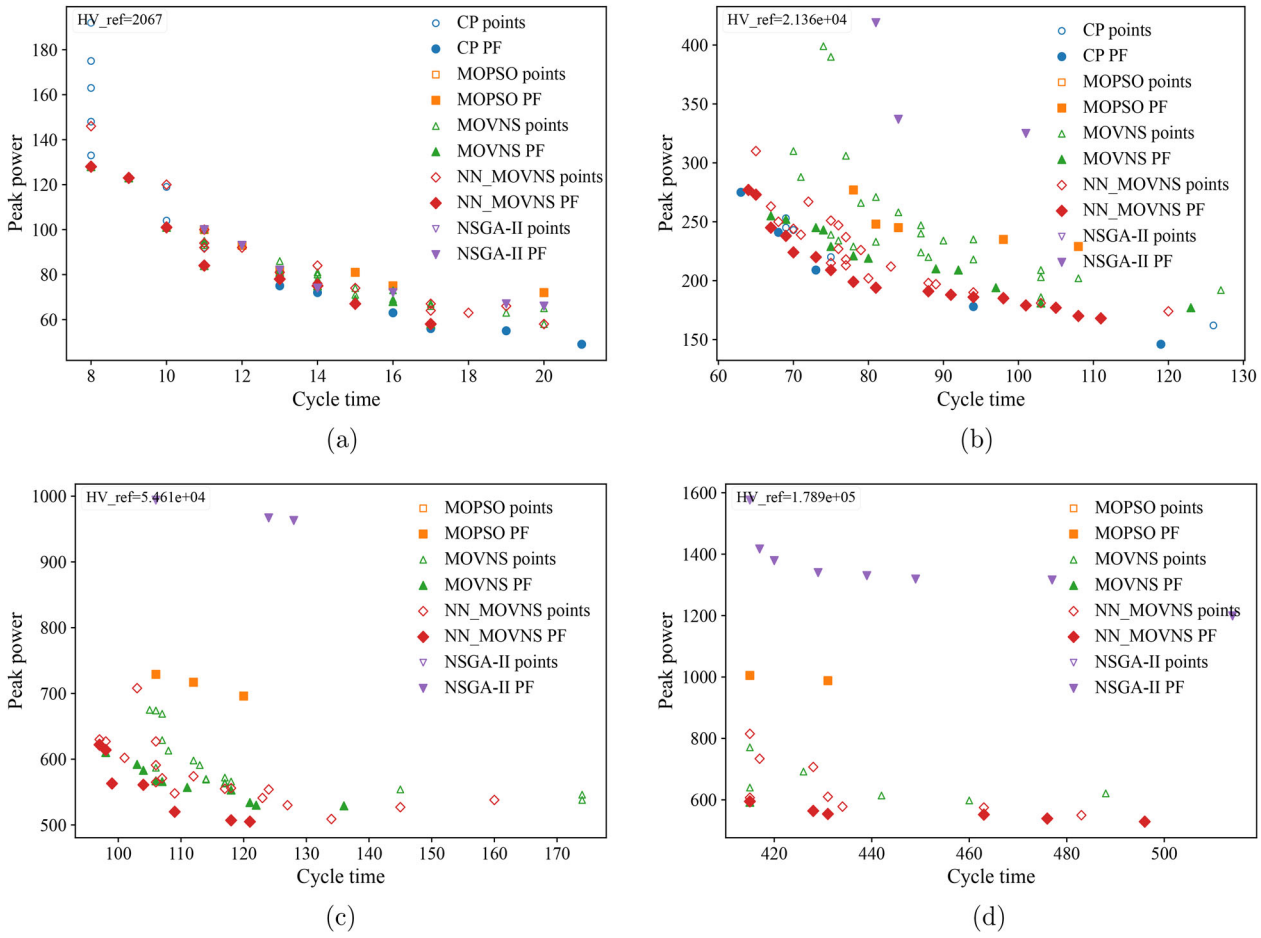

Figure 3. Representative pooled Pareto fronts comparing CP, NN_MOVNS, MOVNS, NSGA-II, and MOPSO across different instance sizes. (a) T4 – JACKSON (small, 11 tasks). (b) T22 – KILBRID (medium, 45 tasks). (c) T34 – TONGE70 (medium, 70 tasks) and (d) T55 – BARTHOLD (large, 148 tasks).

6.5. Neural network component analysis

Figure 4 presents the early (0–25%) and late (75–100%) phase neighbourhood selection frequencies for the NN_MOVNS and MOVNS algorithms across three representative instances: T12, T39, and T55. The comparative analysis reveals distinct behavioural patterns between the two algorithms.

In the case of NN_MOVNS, the results demonstrate an adaptive shift in neighbourhood utilisation over the search process. Certain neighbourhoods (e.g. N2, N5, or N8 depending on the instance) become more prominent in the late phase, indicating that the neural network guidance encourages adaptive intensification and diversified search trajectories as the search progresses. Conversely, MOVNS exhibits either a more uniform or instance-specific pattern of neighbourhood selection. In some cases, the late-phase distribution reinforces early-phase tendencies, while in others, the algorithm shifts its focus dramatically toward a dominant neighbourhood (e.g. N7 in T55).

Overall, these observations from Figure 4 suggest that the neural guidance embedded in NN_MOVNS enables a more dynamic adjustment of search behaviour, potentially contributing to a better exploration–exploitation balance across complex problem instances.

7. Industrial-scale test instance and managerial insights

7.1. Industrial-scale test instance

This section presents a discussion of the computational performance of the developed NN_MOVNS algorithm in the context of a large-scale industrial case. The case is based on the operations of an automotive manufacturing system producing engine cylinder heads on a transfer line. The case is characterised by the following key parameters: number of tasks ($NT = 148$), number of workstations ($NM = 10$), number of blocks ($Nb = 9$), number of tasks per block ($Mtb = 7$), precedence relationships ($pre_{ij} = 175$), inclusion constraints ($inc_{ij} =$

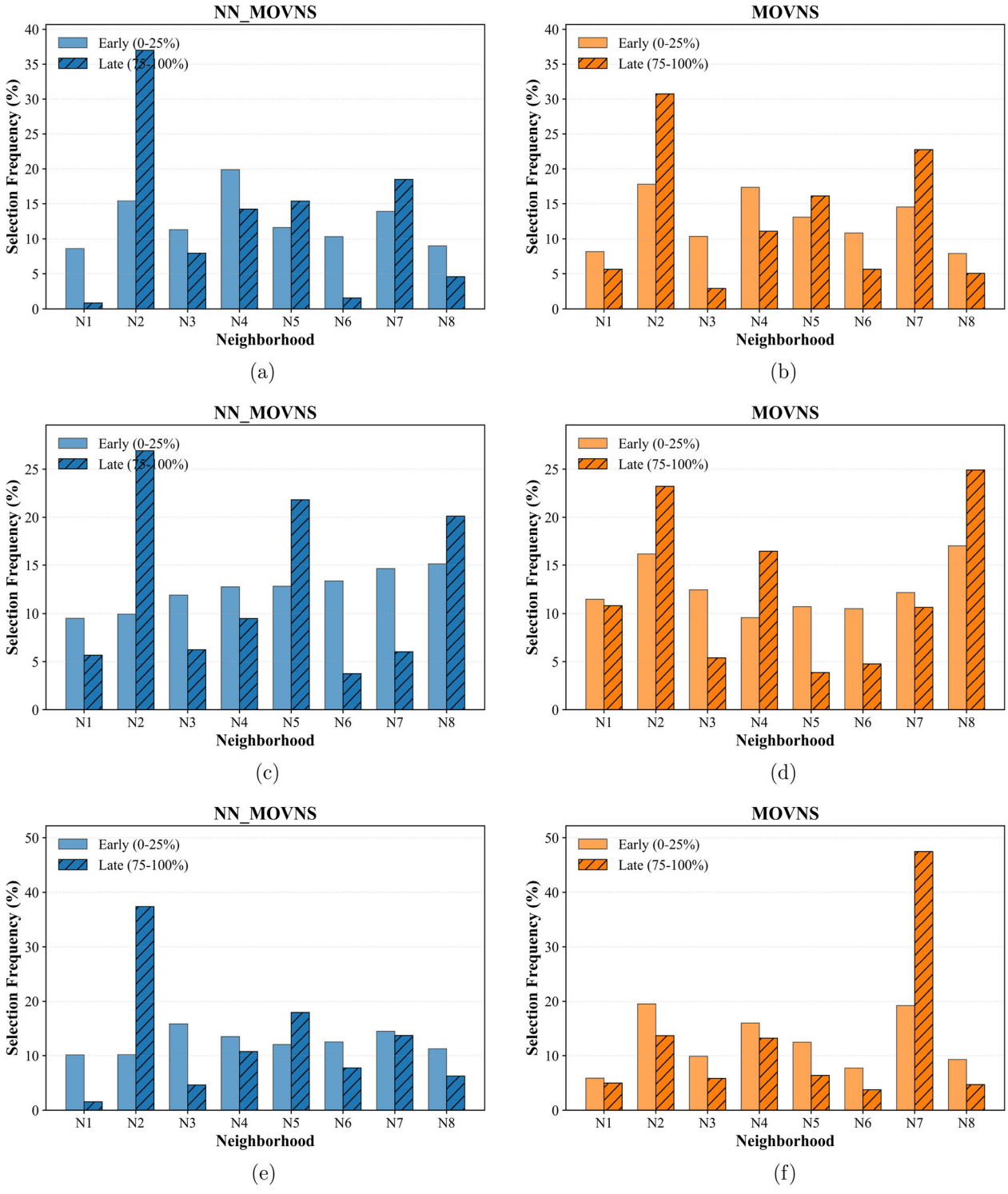

Figure 4. Early vs late phase neighbourhood selection frequencies for NN_MOVNS and MOVNS algorithms on the T12_BUXEY, T39_WEE-MAG, and T55_BARTHOLD instances. (a) NN_MOVNS: T12_BUXEY. (b) MOVNS: T12_BUXEY. (c) NN_MOVNS: T39_WEE-MAG. (d) MOVNS: T39_WEE-MAG. (e) NN_MOVNS: T55_BARTHOLD and (f) MOVNS: T55_BARTHOLD.

23), exclusion constraints ($exc_{ij} = 23$), block exclusion constraints ($exc_b_{ij} = 17$), and technological relationships ($tec_{ik} = 9$).

The comparison between the initial solution and the knee-point solution obtained through NN_MOVNS, as illustrated in Figure 5, demonstrates a substantial improvement in both CT and PP.

The initial configuration exhibits a CT of 485 s, PP of 1469 kW, and a PAPR of 2.35, reflecting highly uneven workload distribution and strong temporal

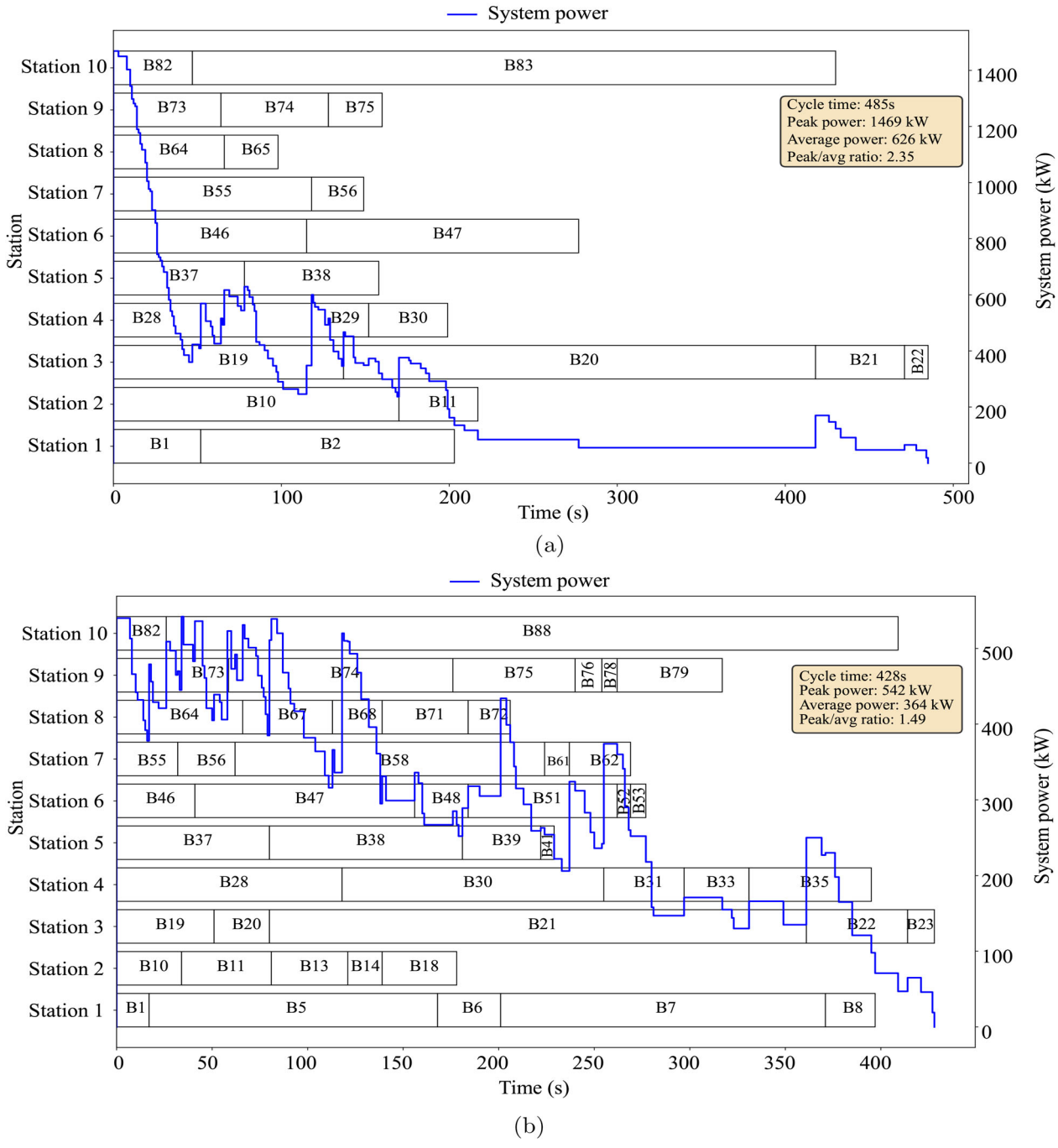

Figure 5. Comparison of (a) the initial solution and (b) the knee-point solution.

concentration of energy-intensive tasks. In contrast, the knee-point solution achieves a CT of 428 s (an 11.8% reduction), PP of 542 kW (a 63.1% reduction), and a PAPR of 1.49, indicating a significantly smoother and more balanced power profile. These improvements mainly result from the refined task-to-block and block-to-workstation assignments generated by NN_MOVNS. By increasing the number of blocks and redistributing them across stations, the algorithm enhances scheduling granularity and inter-task flexibility. Large, power-intensive blocks in the initial configuration are decomposed and distributed across multiple stations, reducing concurrent PP points. Consequently, stations that previously handled few large blocks now operate with several smaller ones, achieving smoother workload distribution and reduced idle times. The detailed assignments for the initial and knee-point configurations are available at https://figshare.com/s/baac254dff6c29fdb5e0.

7.2. Peak shaving and energy redistribution across problem scales

Peak shaving is critical in industrial facilities, where electricity costs are heavily influenced by demand charges based on maximum power demand. Unlike residential users, where fixed capacity charges dominate, industrial users face contractual limits and subscription agreements that make PP reduction directly tied to cost efficiency and operational compliance. In this context, NN_MOVNS reduces PP primarily resulting in energy redistribution. Energy redistribution refers to reshaping the power–time profile such that the same total electrical energy is consumed, but high instantaneous power demands are reduced by spreading energy usage more evenly over the operating period. Task execution is rescheduled to avoid simultaneous high-power events, resulting in lower peaks while preserving the integral (total energy) of the power profile.

To quantify the peak reduction achieved by NN_MOVNS, the PSI is defined as the percentage reduction in maximum PP:

$$\mathrm{PSI} = \frac{(PP^{\max}_{\mathrm{initial}}) - (PP^{\max}_{\mathrm{NN_MOVNS}})}{(PP^{\max}_{\mathrm{initial}})} \times 100$$

A higher PSI indicates a greater reduction in PP.

To directly measure the degree of load reshaping attributable to energy redistribution, the discrete ERI is defined as:

$$\mathrm{ERI} = \frac{\sum \max(0, P_{\mathrm{initial}}[k] - P_{\mathrm{NN_MOVNS}}[k])\,\Delta t}{\sum P_{\mathrm{initial}}[k]\,\Delta t}$$

Here, the numerator represents the energy removed from time instants where the optimised profile is lower than the initial profile (i.e. energy shaved from high-demand periods), while the denominator is the total electrical energy over the cycle. Therefore, $\mathrm{ERI} \in [0, 1]$, where $\mathrm{ERI} = 0$ indicates no redistribution (the profiles coincide in the sense of no shaved energy), and larger ERI values indicate a stronger redistribution effect, meaning more energy has been shifted away from high-power instants and spread across the operating horizon.

Figure 6 presents peak shaving results for six representative instances spanning small, medium, and large problem scales. Table 13 reports the corresponding PSI and ERI values, complementing the visual trends observed in the figure. PP reductions range from 33.33% to 73.17%, with problem size influencing the extent of improvement. Small-scale cases (T11, T18) achieve notable reductions of 46.03% and 53.57%, respectively, despite fewer tasks available for rescheduling. Even in these compact systems, NN_MOVNS eliminates sharp peaks in the initial schedules, confirming that meaningful peak shaving can be obtained even under limited scheduling flexibility. Medium-scale cases (T30, T38) show moderate improvements of 40.68% and 33.33%, respectively. Their initial power curves contain multiple local peaks that are smoothed in the optimised profiles, although the achievable improvements are less pronounced than in larger systems due to fewer degrees of freedom for redistribution. Large-scale cases (T45, T54) achieve the strongest results, with PP reductions of 70.27% and 73.17%, respectively.

Consistent with these results, the proposed method yields notable ERI values across all test problems, with particularly high percentages in T45 and T54, reaching 35.28% and 31.53%. These results indicate that a substantial portion of the high power demand in large machining transfer lines is temporally redistributed over the operating cycle, leading to significantly smoother power profiles without altering the total energy consumption per cycle. However, these improvements can slightly increase CT (by an average of 5.7%, with a maximum of 12.7% in T30), the reductions in PP and the associated redistribution of energy outweigh the relatively small CT increases.

Figure 7 illustrates the system-level impact of the proposed peak shaving strategy on an industrial manufacturing site connected to the regional power grid through a high-voltage (HV)/medium-voltage (MV) transformer. In the initial operating scenario (orange power profile), the machining transfer line feeder imposes a peak demand of approximately 2050 kW, significantly contributing to the overall site peak at both MV and downstream low-voltage (LV) levels. After applying the proposed CT–based peak shaving approach (blue profile), the maximum power demand is reduced to approximately 550 kW, corresponding to a released capacity of nearly 1500 kW. This released electrical capacity can be safely utilised by other MV or LV loads within the manufacturing site, such as auxiliary production lines, utility systems, or additional process equipment, without violating grid constraints or transformer limits. Importantly, this reduction is achieved through temporal redistribution of power demand rather than energy curtailment, preserving the total energy required per production cycle while substantially alleviating peak loading at the feeder and grid interface levels.

7.3. PAPR improvements

Figure 8 demonstrates the consistent effectiveness of the NN_MOVNS approach in lowering PAPR, thereby reducing power demand variability across the test instances. The orange line, which shows initial schedules, indicates significant differences in baseline scheduling quality, with PAPR values ranging from 1.29 to 2.55. This

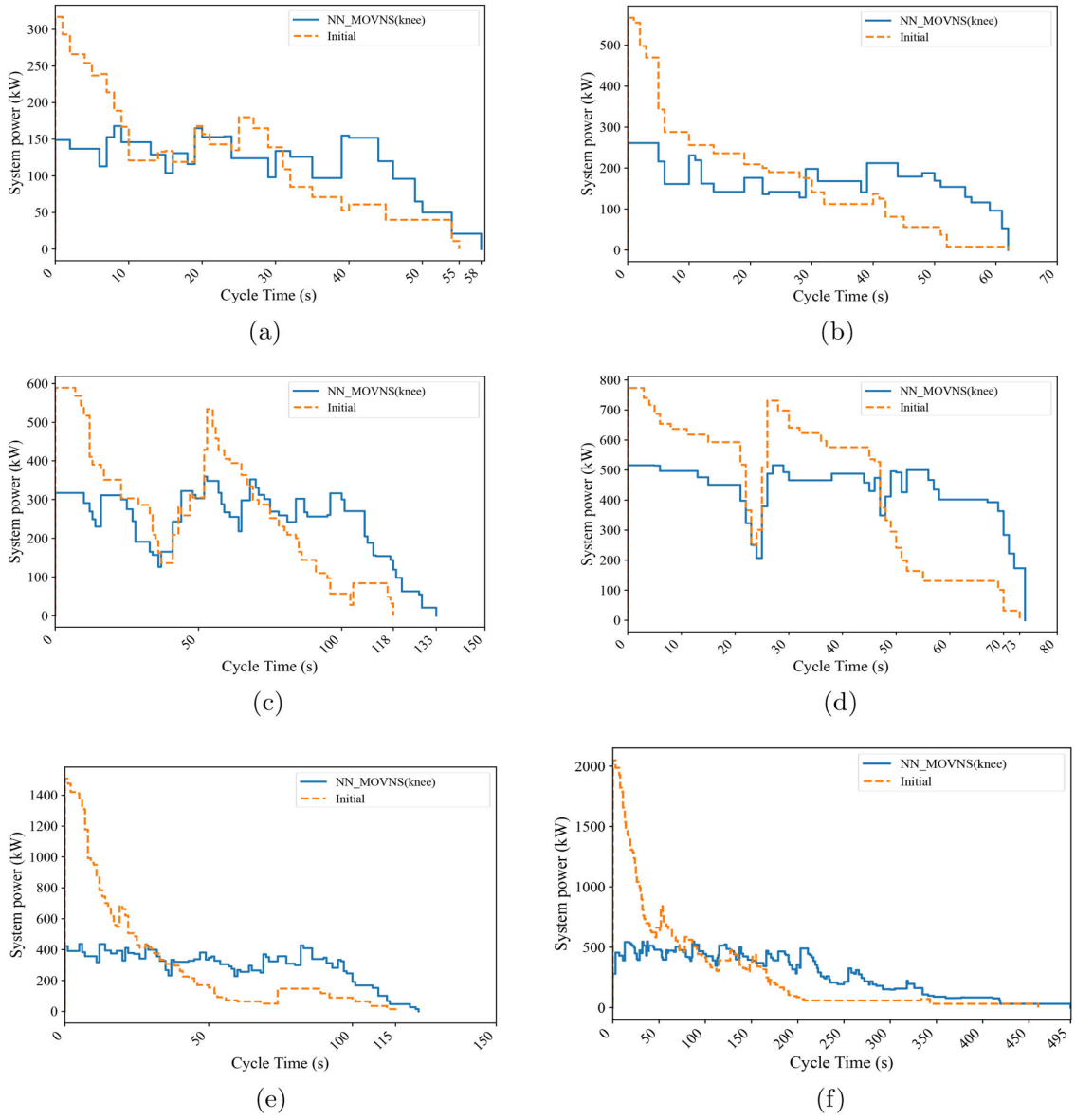

Figure 6. Power profile comparisons for six representative test instances showing initial schedules (orange dashed) versus NN_MOVNS optimised solutions (blue solid). (a) T11 (Small). (b) T18 (Small). (c) T30 (Medium). (d) T38 (Medium). (e) T45 (Large) and (f) T54 (Large).

Table 13. PSI and ERI values for representative test problems.

Test problems		$PP^{max}_{initial}$ (kW)	$PP^{max}_{NN_MOVNS}$ (kW)	PP reduction (kW)	PSI (%)	ERI (%)
Small	T11	315	170	145	46.03	15.32
	T18	560	260	300	53.57	27.05
Medium	T30	590	350	240	40.68	25.28
	T38	780	520	260	33.33	22.19
Large	T45	1480	440	1040	70.27	35.28
	T54	2050	550	1500	73.17	31.53

variety points to notable variability in the power profiles. In contrast, the blue line, representing results at knee points, shows stable improvement with most ratios falling between 1.15 and 1.85. This pattern suggests the algorithm reliably finds and uses available flexibility in each test instance, regardless of starting conditions.

Significantly, the instances with the highest initial PAPR (greater than 2.0) achieve the largest reductions after optimisation, while cases starting closer to 1.5 still see meaningful improvements. This supports the observation that the potential for optimisation is related to how variable the initial power profile is. As such, managers can expect greater performance gains in systems with more uneven starting profiles, and initial PAPR measurements can help estimate the expected improvements for different operational scenarios.

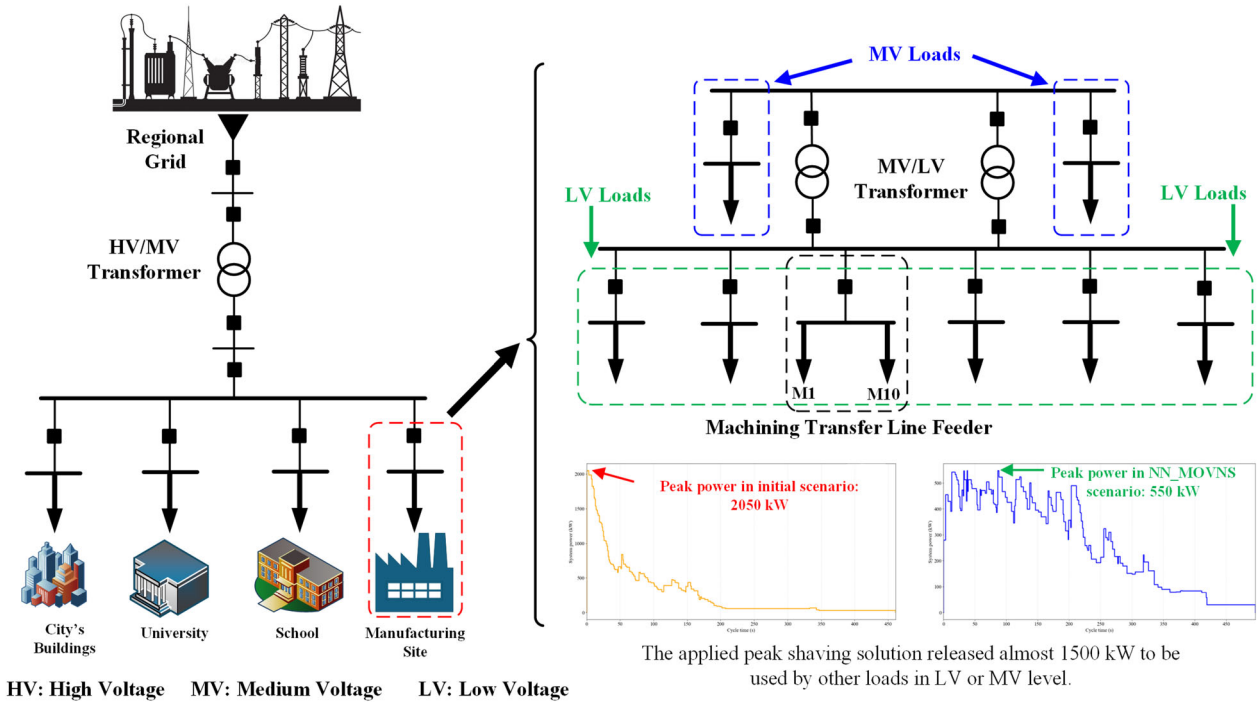

Figure 7. System-level illustration of the proposed peak shaving approach applied to a manufacturing site.

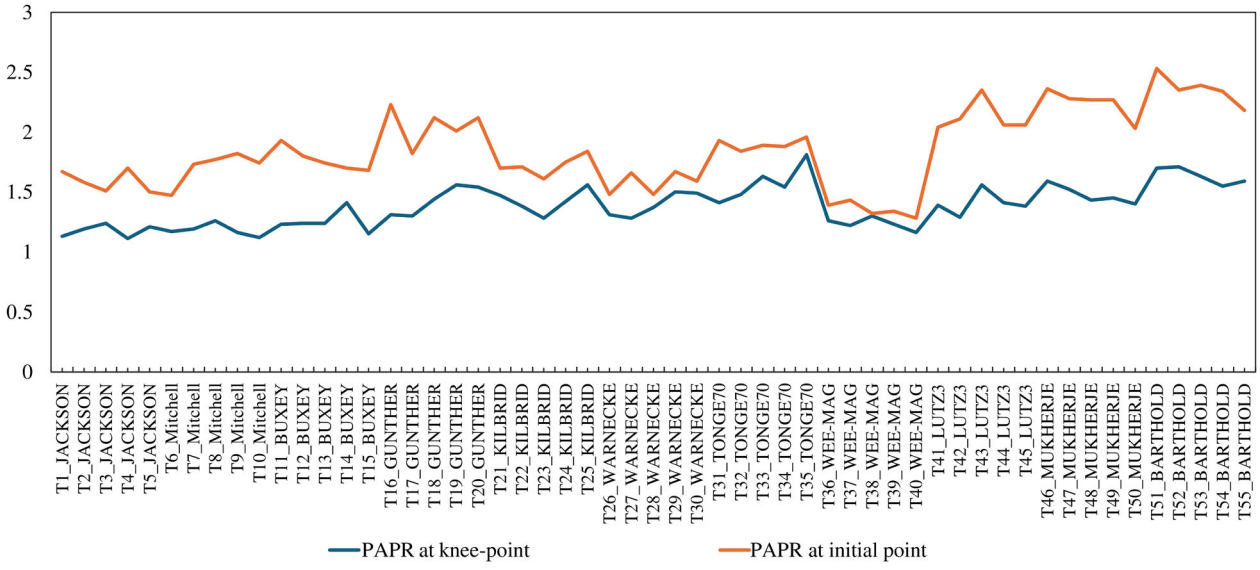

Figure 8. PAPR values across all test instances before and after optimisation.

7.4. Findings and managerial insights

This subsection synthesises the key findings of the study by explicitly addressing the three research questions posed in the Introduction, and by drawing out their practical implications for production and energy management.

The first research question (RQ1) asked how the BSMTL can be jointly formulated as a bi-objective optimisation problem that simultaneously minimises CT and PP under the full set of precedence, inclusion/exclusion, and technological compatibility constraints. The proposed CP model answers this by integrating task-to-block assignment, block formation, and block execution scheduling within a unified framework that captures the temporal interaction between task assignments and system-level power demand (Section 3.1). The formulation confirms that CT and PP cannot be optimised in isolation, since the assignment of tasks to blocks determines which power demands accumulate simultaneously, while the block execution sequence determines when those

peaks occur. Optimising either dimension independently therefore leaves the other uncontrolled, as illustrated by the contrasting solutions in Figure 1. For production managers in automotive and aerospace manufacturing, this implies that energy and scheduling decisions must be treated jointly rather than sequentially, because facilities that balance and schedule as separate steps may be systematically generating avoidable peak demand and inflated electricity charges.

The second research question (RQ2) examined whether a MOVNS enhanced with deep reinforcement learning can outperform both exact methods and standard metaheuristics in terms of Pareto front quality across small, medium, and large-scale instances. The computational experiments confirm that NN_MOVNS consistently outperforms the baseline MOVNS across all three performance indicators and all problem scales, with the advantage becoming increasingly pronounced as instance size grows and reaching an 88.4% reduction in mean IGD^+ on large instances, a difference confirmed as statistically significant by the Wilcoxon signed-rank test (Tables 8, 9, and 11). While CP produces exact or near-exact Pareto sets on small instances, it becomes intractable at medium and large scales, and both VNS-based methods significantly outperform NSGA-II and MOPSO, confirming that the block-based neighbourhood structure is better suited to the BSMTL than population-based search. For production planners and operations engineers, these results indicate that learning-enhanced metaheuristics offer a robust and scalable alternative to exact solvers, capable of delivering high-quality CT and PP trade-off solutions within realistic computation times even for industrial-scale problems.

The third research question (RQ3) focussed on the practical implications of the approach in terms of PP reduction, energy redistribution, and PAPR improvement. As reported in Sections 7 and the peak-shaving analysis above, NN_MOVNS delivers consistent PP reductions across all problem scales, with the accompanying ERI values confirming that these reductions are achieved through temporal redistribution of energy demand rather than curtailment, so that the total energy consumed per production cycle is preserved. Large-scale instances benefit most, owing to their greater scheduling flexibility, and the industrial-scale test instance further demonstrates the same behaviour at realistic problem size. For manufacturers facing electricity demand charges, transformer capacity limits, or site-level grid constraints, these findings confirm that intelligent rescheduling via NN_MOVNS is a practically viable peak shaving tool. It lowers energy costs and improves operational compliance without sacrificing production continuity, while the released electrical capacity can be safely reallocated to auxiliary production lines or utility systems within the same manufacturing site.

8. Conclusions

This study addressed the balancing and scheduling of machining transfer lines (BSMTL), in which cycle time (CT) and peak power (PP) are simultaneously minimised under precedence, inclusion/exclusion, and technological compatibility constraints. Three main contributions were made. First, a multi-objective CP model was developed and solved via a bidirectional ε-constraint method, generating high-quality Pareto sets for small-scale instances and establishing a rigorous mathematical foundation for the joint CT–PP optimisation in machining transfer lines. Second, a MOVNS algorithm with a two-phase constructive heuristic was proposed to address the scalability limitations of the CP model for medium- and large-scale problems. Third, the MOVNS framework was augmented with a DQN-based operator selection mechanism (NN_MOVNS) that dynamically adapts search behaviour during optimisation, improving both convergence and Pareto front diversity over the static baseline.

Computational experiments across 55 benchmark instances confirm that NN_MOVNS consistently outperforms the baseline MOVNS and both population-based methods across all performance indicators and problem scales, with the advantage widening on larger instances and confirmed as statistically significant by Wilcoxon signed-rank tests. On the industrial-scale test instance of a 148-task automotive cylinder head production line, it reduced peak power by 63.1% through temporal energy redistribution rather than curtailment, releasing substantial grid capacity for reallocation to other site loads while keeping the total energy per cycle unchanged.

Beyond the BSMTL problem, the proposed NN_MOVNS framework can generalise to other production line types and combinatorial optimisation settings, provided that its problem-specific components are appropriately redefined. The archive-guided Pareto search structure and the DQN-based adaptive operator selection mechanism are line-type independent and require no structural modification. For assembly lines, the block-based solution representation would be replaced by a station-task assignment encoding, and the neighbourhood operators would be redesigned to act on task reassignment and within-station sequencing rather than block formation. For reconfigurable manufacturing systems, an additional configuration-selection layer would need to be incorporated into the solution encoding and feasibility checks to account for varying machine configurations across production runs. Regarding objective

extensions, incorporating a third objective such as total energy consumption, workload imbalance, or line cost would require extending the $r_{\text{objective}}$ component of the DQN reward function to cover the additional dimension, adjusting the Pareto dominance check accordingly, and updating the DQN state vector to encode the new objective value. The overall learning architecture would otherwise remain unchanged. More broadly, applying the DQN-guided operator selection to other combinatorial problems such as flexible job shop scheduling, vehicle routing, or supply chain network design requires defining a meaningful state representation, a set of complementary neighbourhood operators, and a reward signal reflecting solution quality improvement.

Several limitations of the present study should be acknowledged. First, task processing times and power demand values are assumed to be deterministic and constant; real-world variability due to tool wear, material inconsistencies, or machine conditions is not captured. Second, power demand values for the benchmark instances were estimated from manufacturers' catalogs rather than measured directly on physical systems, which may not fully reflect operational variability. Third, the CP model becomes intractable for medium- and large-scale instances, limiting its use to small-scale validation. Fourth, the study focuses on a single-product, single-cycle setting; multi-product transfer lines with mixed precedence structures are outside the current scope. Fifth, the DQN reward weights ($w_1 = 0.8$, $w_2 = 0.9$, $w_3 = 0.3$) were determined through preliminary experimentation and may require recalibration for significantly different problem structures.

Future research could extend this work in several directions. Incorporating stochastic factors such as machine breakdowns, processing time variability, and real-time energy pricing would better reflect industrial uncertainties. Additional sustainability objectives, including total energy consumption, carbon emissions, and renewable energy utilisation, could also be considered. Extending the framework to multi-product settings and hybrid exact–metaheuristic schemes may further broaden its applicability. Furthermore, combining the complementary strengths of the CP and MILP formulations through decomposition schemes such as Benders decomposition or Dantzig–Wolfe reformulation could yield more scalable exact methods for larger instances. Moreover, generative AI techniques, such as diffusion models or AI-generated content frameworks (Leng et al. 2025), could be integrated into the initialisation or neighbourhood design phases of NN_MOVNS to improve convergence speed and Pareto front diversity in dynamic production environments. Future extensions could also explore federated learning frameworks in which the DQN operator selection policy is trained collaboratively across multiple production sites without sharing sensitive operational data, enabling privacy-preserving peak shaving and energy redistribution in distributed, multi-plant manufacturing networks.

Author contributions

CRediT: **Pedram Beldar:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing; **Ehsan Mahmoodi:** Conceptualization, Formal analysis, Methodology, Software, Validation, Visualization, Writing – review & editing; **Mostafa Kermani:** Conceptualization, Methodology, Supervision, Validation, Writing – review & editing; **Gary Linnéusson:** Investigation, Supervision, Validation, Writing – review & editing; **Olga Battaia:** Conceptualization, Methodology, Supervision, Writing – review & editing

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was partly funded by VINNOVA, Sweden's Innovation Agency, through the REFUSE 2.0 project (Grant No. 2025-04156).

Notes on contributors

Pedram Beldar is a PhD student in Informatics at the School of Engineering Science, University of Skövde, Sweden. His research is centered on operations research and optimization in production systems, with particular interest in batch processing machines and machining transfer lines. His work involves the development of mathematical programming, heuristic, metaheuristic, and learning-based approaches for complex decision-making problems in manufacturing. His broader research interests include production scheduling, reconfigurable manufacturing, sustainable production, and energy-efficient manufacturing systems.

Ehsan Mahmoodi is a researcher at the University of Auckland, New Zealand. He holds a PhD in Information Technology from the University of Skövde, Sweden. His research brings together operations research, simulation-based optimization, artificial intelligence, and data-driven decision support to address complex engineering and socio-technical problems. A central focus of his work is the design of manufacturing systems: he develops knowledge-driven methods that use simulation, digital twins, and multi-objective optimization to improve decisions about resources, production flow, and system performance under uncertainty. His broader research spans sustainable operations,

logistics, and human-centered production, often in collaboration with industry. His work has appeared in Computers & Operations Research, Applied Energy, Computers & Industrial Engineering, Journal of Manufacturing Systems, and Applied Ergonomics.

Mostafa Kermani (Member, IEEE) received the Ph.D. degree in electrical engineering from the Sapienza University of Rome, Rome, Italy, in 2019 where he was a Postdoctoral Researcher from 2019 to 2020 and as an RTDA from 2023 to 2024. He was a Senior Researcher from 2020 to 2022 with the Electrical Engineering Department, Chalmers University of Technology, Gothenburg, Sweden. In 2022, he was a Visiting Researcher with EPFL, Switzerland. He was also a Senior Project Engineer from 2022 to 2023 in the Network Control Department, Hitachi Energy, Västerås, Sweden. He is currently an Assistant Professor with the School of Engineering Science, University of Skövde, Skövde, Sweden. He is author of more than 60 papers in peer-reviewed international journals and conferences. He is an Associate Editor for IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS SOCIETY and a Member of the scientific committee at IEEE EEEIC conference. His research interests include modern power systems, microgrids, energy storage systems, energy systems modeling and optimization, electric vehicles and intelligent energy management systems.

Gary Linnéusson is an Associate Professor at the Department of Human, Technology and Management at the School of Engineering Science, University of Skövde, Sweden. He holds a background in mechanical engineering (2003), a Ph.D. in Informatics (2018), and experience in both industrial and academic settings, including production engineering, strategic maintenance, integrated product and production development, healthcare management, and electrification challenges in sub-regional contexts. He has applied simulation tools within system dynamics and discrete event simulation. His primary research interests focus on transition research, especially how systems thinking can help structure complex and wicked problems, using simulation and simulation-based optimization to support management areas within sustainable production, regional development, and the strategic development of the healthcare system in West Sweden.

Prof. Olga Battaïa is a Full Professor at the Department of Operations Management and Information Systems at KEDGE Business School. She currently serves as Associate Editor for Journal of Manufacturing Systems, IISE Transactions, and Omega – The International Journal of Management Science. She is also a member of the Editorial Board of the International Journal of Production Economics and a member of the International Advisory Board for Nexus 2050. She also acts as DEI Chair of the IFAC Technical Committee on Management and Control in Manufacturing and Logistics. Prof. Battaïa's research lies at the intersection of supply chain management, sustainable manufacturing, business analytics, and equity-aware decision support. She has co-authored more than 190 peer-reviewed publications, coordinated and contributed to numerous competitive research projects funded by the French National Research Agency, the German Research Foundation (DFG), and the European Union, addressing complex challenges such as decision-making under uncertainty, sustainable and reconfigurable production systems, and human-centric operations management. An internationally recognized speaker, she has presented her work across Europe, Africa, the Americas, Asia, and Australia.

Data availability statement

Data will be made available on request.

References

Bänsch, K., J. Busse, F. Meisel, J. Rieck, S. Scholz, T. Volling, and M. G. Wichmann. 2021. "Energy-Aware Decision Support Models in Production Environments: A Systematic Literature Review." *Computers & Industrial Engineering* 159:107456. https://doi.org/10.1016/j.cie.2021.107456.

Battaïa, O., and A. Dolgui. 2022. "Hybridizations in Line Balancing Problems: A Comprehensive Review on New Trends and Formulations." *International Journal of Production Economics* 250:108673. https://doi.org/10.1016/j.ijpe.2022.108673.

Battaïa, O., A. Dolgui, and N. Guschinsky. 2023. "MIP-based Heuristics for Combinatorial Design of Reconfigurable Rotary Transfer Machines for Production of Multiple Parts." *International Journal of Production Economics* 262:108904. https://doi.org/10.1016/j.ijpe.2023.108904.

Battaïa, O., A. Dolgui, and N. Guschinsky. 2024. "An Exact Method for Machining Lines Design with Equipment Selection and Line Balancing." *International Journal of Production Research* 62 (1–2): 71–91. https://doi.org/10.1080/00207543.2023.2289182.

Beldar, P., M. Fathi, A. Nourmohammadi, X. Delorme, O. Battaïa, and A. Dolgui. 2025. "Transfer Line Balancing Problem: A Comprehensive Review, Classification, and Research Avenues." *Computers & Industrial Engineering* 201:110913. https://doi.org/10.1016/j.cie.2025.110913.

Beldar, P., E. Mahmoodi, M. Kermani, E. Mardaneh, and A. H. Ansaripoor. 2026. "Energy-Aware Batch Scheduling with Flexible Maintenance under Spot Electricity Pricing and Onsite Renewable Generation." *Applied Energy* 422:128283. https://doi.org/10.1016/j.apenergy.2026.128283.

Beldar, P., A. Nourmohammadi, M. Fathi, and A. H. C. Ng. 2024. "A Heuristic Approach for Flexible Transfer Line Balancing Problem." *Procedia CIRP* 130:1144–1149. https://doi.org/10.1016/j.procir.2024.10.219.

Bérubé, J.-F., M. Gendreau, and J.-Y. Potvin. 2009. "An Exact ϵ-constraint Method for Bi-objective Combinatorial Optimization Problems: Application to the Traveling Salesman Problem with Profits." *European Journal of Operational Research* 194 (1): 39–50. https://doi.org/10.1016/j.ejor.2007.12.014.

Chen, Y., X. Huang, W. Yi, Z. Pei, C. Wang, and Z. Ji. 2025. "Bi-level Collaborative Day-Ahead Optimization Scheduling of Manufacturing-Energy System Targeting Energy Cost Reduction." *Energy* 338:138672. https://doi.org/10.1016/j.energy.2025.138672.

Delorme, X., A. Cerqueus, P. Gianessi, and D. Lamy. 2023. "RMS Balancing and Planning under Uncertain Demand and Energy Cost Considerations." *International Journal of Production Economics* 261:108873. https://doi.org/10.1016/j.ijpe.2023.108873.

Delorme, X., and P. Gianessi. 2022. "Designing Reconfigurable Manufacturing Systems to Minimize Power Peak." *IFAC-PapersOnLine* 55 (10): 1296–1301. https://doi.org/10.1016/j.ifacol.2022.09.569.

Delorme, X., and P. Gianessi. 2024. "Line Balancing and Task Scheduling to Minimise Power Peak of Reconfigurable Manufacturing Systems." *International Journal of Production Research* 62 (14): 5061–5086. https://doi.org/10.1080/00207543.2023.2283568.

Destouet, C., H. Tlahig, B. Bettayeb, and B. Mazari. 2024. "Multi-objective Sustainable Flexible Job Shop Scheduling Problem: Balancing Economic, Ecological, and Social Criteria." *Computers & Industrial Engineering* 195:110419. https://doi.org/10.1016/j.cie.2024.110419.

Ersin Telemeci, Y., and M. Azizoğlu. 2024. "Type-II Transfer Line Balancing Problem – A Branch and Bound Approach." *Computers & Industrial Engineering* 198:110689. https://doi.org/10.1016/j.cie.2024.110689.

Fang, Y., Q. Liu, M. Li, Y. Laili, and D. T. Pham. 2019. "Evolutionary Many-objective Optimization for Mixed-Model Disassembly Line Balancing with Multi-robotic Workstations." *European Journal of Operational Research* 276 (1): 160–174. https://doi.org/10.1016/j.ejor.2018.12.035.

Fang, Y., H. Ming, M. Li, Q. Liu, and D. T. Pham. 2020. "Multi-objective Evolutionary Simulated Annealing Optimisation for Mixed-Model Multi-robotic Disassembly Line Balancing with Interval Processing Time." *International Journal of Production Research* 58 (3): 846–862. https://doi.org/10.1080/00207543.2019.1602290.

Gianessi, P., X. Delorme, and O. Masmoudi. 2019. "Simple Assembly Line Balancing Problem with Power Peak Minimization." In *Advances in Production Management Systems. Production Management for the Factory of the Future*, 239–247. Cham: Springer International Publishing.

Guo, L., Z. Zhang, Y. Zhang, Y. Li, and H. Song. 2025. "Exploring Engineering Applications of Two-Sided Partial Destructive Disassembly Line Balancing Problem under Electrical Limiting and Time-of-use Pricing." *Expert Systems with Applications* 262:125573. https://doi.org/10.1016/j.eswa.2024.125573.

Guo, T., Y. Mei, W. Du, Y. Lv, Y. Li, and T. Song. 2025. "Emergency Scheduling of Aerial Vehicles via Graph Neural Neighborhood Search." *IEEE Transactions on Artificial Intelligence* 6 (7): 1808–1822. https://doi.org/10.1109/TAI.2025.3528381.

Guschinskaya, O., and A. Dolgui. 2009. "Comparison of Exact and Heuristic Methods for a Transfer Line Balancing Problem." *International Journal of Production Economics* 120 (2): 276–286. https://doi.org/10.1016/j.ijpe.2008.11.018.

Helgeson, W. B., and D. P. Birnie. 1961. "Assembly Line Balancing Using the Ranked Positional Weight Technique." *Journal of Industrial Engineering* 12 (6): 394–398.

Kalatzantonakis, P., A. Sifaleras, and N. Samaras. 2025. "An Offline Data-Driven Process for Learning Operator Selection from Metaheuristic Search Traces." *Swarm and Evolutionary Computation* 98:102058. https://doi.org/10.1016/j.swevo.2025.102058.

Kheirabadi, M., S. Keivanpour, J.-M. Frayret, and Y. Chinniah. 2025. "Safety-Driven Optimisation of Human–robot Collaborative Assembly Line Balancing." *International Journal of Production Research* 63 (17): 6203–6228. https://doi.org/10.1080/00207543.2025.2469288.

Lahrichi, Y., L. Deroussi, N. Grangeon, and S. Norre. 2021. "A Balance-First Sequence-Last Algorithm to Design RMS: A Matheuristic with Performance Guaranty to Balance Reconfigurable Manufacturing Systems." *Journal of Heuristics* 27 (1-2): 107–132. https://doi.org/10.1007/s10732-021-09473-1.

Lahrichi, Y., N. Grangeon, L. Deroussi, and S. Norre. 2021. "A New Split-Based Hybrid Metaheuristic for the Reconfigurable Transfer Line Balancing Problem." *International Journal of Production Research* 59 (4): 1127–1144. https://doi.org/10.1080/00207543.2020.1720929.

Lamy, D., X. Delorme, and P. Gianessi. 2020. "Line Balancing and Sequencing for Peak Power Minimization." *IFAC-PapersOnLine* 53 (2): 10411–10416. https://doi.org/10.1016/j.ifacol.2020.12.2781.

Leng, J., G. Ruan, C. Xu, X. Song, Q. Liu, Y. Fu, and J. Wan. 2024. "Deep Reinforcement Learning of Graph Convolutional Neural Network for Resilient Production Control of Mass Individualized Prototyping toward Industry 5.0." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54 (12): 7152–7165.

Leng, J., X. Su, Z. Liu, L. Zhou, C. Chen, X. Guo, Y. Wang, et al. 2025. "Diffusion Model-Driven Smart Design and Manufacturing: Prospects and Challenges." *Journal of Manufacturing Systems* 82:561–577. https://doi.org/10.1016/j.jmsy.2025.07.011.

Li, J.-K., X.-X. Geng, G.-J. Hu, and Z.-S. Du. 2026. "Q-learning-Driven Growing Neural Gas Network for Job Shop with Machine Breakdown and Setup." *Engineering Applications of Artificial Intelligence* 175:114618. https://doi.org/10.1016/j.engappai.2026.114618.

Li, P., Q. Xue, Z. Zhang, J. Chen, and D. Zhou. 2023. "Multi-objective Energy-Efficient Hybrid Flow Shop Scheduling Using Q-learning and GVNS Driven NSGA-II." *Computers and Operations Research* 159:106360. https://doi.org/10.1016/j.cor.2023.106360.

Liu, C., K. Chen, H. Wang, B. Yang, and J. Leng. 2025. "Job Shop Scheduling by Deep Dual-Q Network with Prioritized Experience Replay for Resilient Production Control in Flexible Manufacturing System." *Computers and Operations Research* 183:107190. https://doi.org/10.1016/j.cor.2025.107190.

Mahmoodi, E., M. Fathi, M. Tavana, M. Ghobakhloo, and A. H. C. Ng. 2024. "Data-Driven Simulation-Based Decision Support System for Resource Allocation in Industry 4.0 and Smart Manufacturing." *Journal of Manufacturing Systems* 72:287–307. https://doi.org/10.1016/j.jmsy.2023.11.019.

Meng, K., Q. Tang, and Z. Zhang. 2022. "Robust Assembly Line Balancing Problem considering Preventive Maintenance Scenarios with Interval Processing Time." *Engineering Applications of Artificial Intelligence* 116:105417. https://doi.org/10.1016/j.engappai.2022.105417.

Ming, F., W. Gong, L. Wang, and Y. Jin. 2024. "Constrained Multi-objective Optimization with Deep Reinforcement Learning Assisted Operator Selection." *IEEE/CAA Journal of Automatica Sinica* 11 (4): 919–931. https://doi.org/10.1109/JAS.2023.123687.

Ming, H., Q. Liu, and D. T. Pham. 2019. "Multi-robotic Disassembly Line Balancing with Uncertain Processing Time." *Procedia CIRP* 83:71–76. https://doi.org/10.1016/j.procir.2019.02.140.

Nourmohammadi, A., T. Arbaoui, M. Fathi, and A. Dolgui. 2025. "Balancing Human–robot Collaborative Assembly Lines: A Constraint Programming Approach." *Computers & Industrial Engineering* 205:111154. https://doi.org/10.1016/j.cie.2025.111154.

Nourmohammadi, A., P. Beldar, M. Fathi, and E. Mahmoodi. 2024. "Balancing and Scheduling of Sustainable Flexible Transfer Lines." *IFAC-PapersOnLine* 58 (19): 664–669. https://doi.org/10.1016/j.ifacol.2024.09.224.

Osama, M., S. S. S. M. Qadri, M. B. Riaz, and M. F. Shahid. 2026. "AGENT: An Elitism-Guided Evolutionary Framework for Enhanced Task Allocation Performance in Heterogeneous Cloud Systems." *Results in Engineering* 29:109742. https://doi.org/10.1016/j.rineng.2026.109742.

Petersen, J., A. Nourmohammadi, M. Fathi, and C. Burmeister. 2024. "Multi-objective Optimization of Transfer Line Balancing Problem considering Cycle Time and Energy Expenditure." *Procedia CIRP* 130:1378–1383. https://doi.org/10.1016/j.procir.2024.10.255.

Rakovitis, N., D. Li, N. Zhang, J. Li, L. Zhang, and X. Xiao. 2022. "Novel Approach to Energy-Efficient Flexible Job-Shop Scheduling Problems." *Energy* 238:121773. https://doi.org/10.1016/j.energy.2021.121773.

Sandalow, D., M. Meidan, P. Andrews-Speed, A. Hove, S. Y. Qiu, and E. Downie. 2022. "Manufacturing Sector." Chapter 11. In *Guide to Chinese Climate Policy*. Oxford: Oxford Institute of Energy Studies. https://chineseclimatepolicy.oxfordenergy.org/

Scholl, A., N. Boysen, M. Fliedner, and R. Klein. 1995. "Homepage for Assembly Line Optimization Research." Online Database.

U.S. Energy Information Administration. 2021. *International Energy Outlook 2021*. https://www.eia.gov/outlooks/ieo/consumption/sub-topic-03.php.

Zhang, L., Q. Tang, Z. Wu, and F. Wang. 2017. "Mathematical Modeling and Evolutionary Generation of Rule Sets for Energy-Efficient Flexible Job Shops." *Energy* 138:210–227. https://doi.org/10.1016/j.energy.2017.07.005.

Zhang, Y., X. Rao, C. Liu, X. Zhang, and Y. Zhou. 2023. "A Cooperative EV Charging Scheduling Strategy Based on Double Deep Q-network and Prioritized Experience Replay." *Engineering Applications of Artificial Intelligence* 118:105642. https://doi.org/10.1016/j.engappai.2022.105642.

Zhang, Z., L. Guo, Y. Zhang, Y. Zeng, H. Song, Y. Li, H. Chen, and M. W. Aslam. 2025. "Industrial Application of Multi-robot Destructive Disassembly Line Balancing under Component Non-disassemblability for Multi-product Scenarios." *International Journal of Production Research* 63 (18): 6801–6826. https://doi.org/10.1080/00207543.2025.2487923.

Zhao, L., Q. Tang, and Z. Zhang. 2022. "An Improved Preference-Based Variable Neighborhood Search Algorithm with Ar-Dominance for Assembly Line Balancing considering Preventive Maintenance Scenarios." *Engineering Applications of Artificial Intelligence* 109:104593. https://doi.org/10.1016/j.engappai.2021.104593.

Zhao, L., Q. Tang, Z. Zhang, and Y. Zhu. 2024. "A Knowledge-Assisted Variable Neighborhood Search for Two-Sided Assembly Line Balancing considering Preventive Maintenance Scenarios." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54 (11): 6859–6872. https://doi.org/10.1109/TSMC.2024.3407724.

Zheng, C., Z. Li, M. Janardhanan, Z. Zhang, and L. Zhang. 2025. "Balancing and Scheduling Human-Robot Collaborative Assembly Lines with Heterogeneous Robots and Limited Resources: Constraint Programming Approach and Fruit Fly Optimization Algorithm." *Computers & Industrial Engineering* 203:111046. https://doi.org/10.1016/j.cie.2025.111046.

Zheng, C., Z. Li, and S. G. Ponnambalam. 2025. "Balancing U-shaped Assembly Lines with Collaborative Robots: Constraint Programming Approaches and Benders' Decomposition Algorithms." *International Journal of Production Research* 63 (24): 10468–10489. https://doi.org/10.1080/00207543.2025.2551245.

Zheng, Z., S. Cherif, and R. S. Shibasaki. 2024. "Optimizing Power Peaks in Simple Assembly Line Balancing through Maximum Satisfiability." In *2024 IEEE 36th International Conference on Tools with Artificial Intelligence* (ICTAI), 363–370. Herndon, VA. https://doi.org/10.1109/ICTAI62512.2024.00060.

Zhou, F.-S., R. Hu, W.-B. Zhang, B. Qian, Z.-Q. Zhang, and L. Wang. 2026. "Spatial Decomposition-Based Variable Neighborhood Search Algorithm for the Integrated Semiconductor Backend Production and Distribution Scheduling Problem." *Computers & Operations Research* 189:107388. https://doi.org/10.1016/j.cor.2026.107388.

Appendices

Appendix 1. Mixed-integer linear programming formulation

Parameters:

NT	Number of tasks
NM	Number of workstations
Nb	Number of blocks per workstation
Ta_i	Processing time of task i
Tp_i	Power demand of task i
$pre_{i,j}$	1 if task i must precede task j
$inc_{i,j}$	1 if tasks i and j must be on the same workstation
$exc_{i,j}$	1 if tasks i and j cannot be on the same workstation
$exc_b_{i,j}$	1 if tasks i and j cannot be in the same block
$tec_{i,k}$	1 if task i is technologically restricted to workstation k
Mtb	Maximum number of tasks per block
H	Safe upper bound on the time horizon, $H = \sum_{i \in O} Ta_i$
$\mathcal{M}^{\text{big}}$	Big-M constant

Sets:

O	Set of tasks indexed by i ($O = \{1, \ldots, NT\}$)
M	Set of workstations indexed by k ($M = \{1, \ldots, NM\}$)
B	Set of blocks indexed by q ($B = \{1, \ldots, Nb \cdot NM\}$)
$R(k)$	Set of blocks on workstation k, $R(k) = \{(k-1) \cdot Nb + 1, \ldots, k \cdot Nb\}$
$\mathcal{T}$	Discrete time horizon, $\mathcal{T} = \{0, 1, \ldots, H\}$

Decision Variables:

$z_{i,q}$	Binary variable: 1 if task i is assigned to block q, $\forall i \in O, \forall q \in B$
$s_{i,q}$	Start time of task i within block q (zero when $z_{i,q} = 0$), $\forall i \in O, \forall q \in B$
BS_q	Start time of block q, $\forall q \in B$
BE_q	End time of block q, $\forall q \in B$
a_q	Binary variable: 1 if block q is active, 0 otherwise, $\forall q \in B$
$u_{t,q,i}$	Binary variable: 1 if task i assigned to block q is being processed at time t, $\forall t \in \mathcal{T}, \forall q \in B, \forall i \in O$
$w^1_{t,q,i}$	Binary variable: 1 if task i in block q has not yet started at time t, $\forall t \in \mathcal{T}, \forall q \in B, \forall i \in O$
$w^2_{t,q,i}$	Binary variable: 1 if task i in block q has already finished by time t, $\forall t \in \mathcal{T}, \forall q \in B, \forall i \in O$
$\mathrm{act}_{t,q,i}$	Power drawn by task i in block q at time t, $\forall t \in \mathcal{T}$, $q \in B, i \in O$
$\mathrm{blk}_{q,t}$	Total power in block q at time t, $\forall q \in B, t \in \mathcal{T}$
$\mathrm{mach}_{k,t}$	Total power in workstation k at time t, $\forall k \in M, t \in \mathcal{T}$
sys_t	System-level power at time t, $\forall t \in \mathcal{T}$
PP	Maximum system peak power
CT	Cycle time

Model Formulation:

$$\text{Minimize} \quad \mathrm{CT} \tag{A1}$$

$$\text{Minimize} \quad \mathrm{PP} \tag{A2}$$

subject to:

$$\sum_{q \in B} z_{i,q} = 1, \quad \forall\, i \in O \tag{A3}$$

$$\sum_{q \in B} q \cdot z_{i,q} \le \sum_{q \in B} q \cdot z_{j,q}, \quad \forall\, i < j, \text{ if } pre_{i,j} = 1 \tag{A4}$$

$$z_{i,q} + z_{j,q} \le 1, \quad \forall\, i < j, \forall\, q \in B, \text{ if } exc_b_{i,j} = 1 \tag{A5}$$

$$\sum_{q \in R(k)} z_{i,q} = \sum_{q \in R(k)} z_{j,q}, \quad \forall\, i < j, \forall\, k \in M, \text{ if } inc_{i,j} = 1 \tag{A6}$$

$$\sum_{q \in R(k)} z_{i,q} + \sum_{q \in R(k)} z_{j,q} \le 1, \quad \forall\, i < j, \forall\, k \in M, \text{ if } exc_{i,j} = 1 \tag{A7}$$

$$\sum_{q \in R(k)} z_{i,q} = 1, \quad \forall\, i \in O, \forall\, k \in M, \text{ if } tec_{i,k} = 1 \tag{A8}$$

$$\sum_{i \in O} z_{i,q} \le Mtb, \quad \forall\, q \in B \tag{A9}$$

$$s_{i,q} \le \mathcal{M}^{\mathrm{big}} \cdot z_{i,q}, \quad \forall\, i \in O, \forall\, q \in B \tag{A10}$$

$$s_{i,(k-1)\cdot Nb+1} \le \mathcal{M}^{\mathrm{big}} \cdot \left(1 - z_{i,(k-1)\cdot Nb+1}\right), \quad \forall\, i \in O, \forall\, k \in M \tag{A11}$$

$$\mathrm{BS}_q \ge s_{i,q} - \mathcal{M}^{\mathrm{big}} \cdot (1 - z_{i,q}), \quad \forall\, i \in O, \forall\, q \in B \tag{A12}$$

$$\mathrm{BS}_q \le s_{i,q} + \mathcal{M}^{\mathrm{big}} \cdot (1 - z_{i,q}), \quad \forall\, i \in O, \forall\, q \in B \tag{A13}$$

$$\mathrm{BE}_q \ge s_{i,q} + Ta_i - \mathcal{M}^{\mathrm{big}} \cdot (1 - z_{i,q}), \quad \forall\, i \in O, \forall\, q \in B \tag{A14}$$

$$a_q \ge z_{i,q}, \quad \forall\, i \in O, \forall\, q \in B \tag{A15}$$

$$a_q \le \sum_{i \in O} z_{i,q}, \quad \forall\, q \in B \tag{A16}$$

$$\mathrm{BE}_{q_1} \ge \mathrm{BS}_{q_2} - \mathcal{M}^{\mathrm{big}} \cdot (2 - a_{q_1} - a_{q_2}), \quad \forall\, k \in M, \forall\, q_1, q_2 \in R(k),\ q_2 = q_1 + 1 \tag{A17}$$

$$\mathrm{BE}_{q_1} \le \mathrm{BS}_{q_2} + \mathcal{M}^{\mathrm{big}} \cdot (2 - a_{q_1} - a_{q_2}), \quad \forall\, k \in M, \forall\, q_1, q_2 \in R(k),\ q_2 = q_1 + 1 \tag{A18}$$

$$a_{q-1} \ge a_q, \quad \forall\, k \in M, \forall\, q \in R(k),\ q > (k-1) \cdot Nb + 1 \tag{A19}$$

$$\mathrm{BE}_q \ge \mathrm{BS}_q - \mathcal{M}^{\mathrm{big}} \cdot a_q, \quad \forall\, q \in B \tag{A20}$$

$$\mathrm{BE}_q \le \mathrm{BS}_q + \mathcal{M}^{\mathrm{big}} \cdot a_q, \quad \forall\, q \in B \tag{A21}$$

$$u_{t,q,i} \le z_{i,q}, \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{A22}$$

$$s_{i,q} \le t + \mathcal{M}^{\mathrm{big}} \cdot (1 - u_{t,q,i}), \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{A23}$$

$$s_{i,q} + Ta_i \ge (t+1) - \mathcal{M}^{\mathrm{big}} \cdot (1 - u_{t,q,i}), \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{A24}$$

$$s_{i,q} \ge (t+1) \cdot w^1_{t,q,i}, \quad \forall t \in \mathcal{T},\ q \in B,\ i \in O \tag{A25}$$

$$s_{i,q} - t \le \mathcal{M}^{\mathrm{big}} \cdot w^1_{t,q,i}, \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{A26}$$

$$s_{i,q} + Ta_i \le t + \mathcal{M}^{\mathrm{big}} \cdot (1 - w^2_{t,q,i}), \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{A27}$$

$$s_{i,q} + Ta_i \ge (t+1) - \mathcal{M}^{\mathrm{big}} \cdot w^2_{t,q,i}, \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{A28}$$

$$u_{t,q,i} \ge z_{i,q} - w^1_{t,q,i} - w^2_{t,q,i}, \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{A29}$$

$$\mathrm{act}_{t,q,i} = Tp_i \cdot u_{t,q,i}, \quad \forall\, t \in \mathcal{T},\ q \in B,\ i \in O \tag{A30}$$

$$\mathrm{blk}_{q,t} = \sum_{i \in O} \mathrm{act}_{t,q,i}, \quad \forall\, q \in B,\ t \in \mathcal{T} \tag{A31}$$

$$\mathrm{mach}_{k,t} = \sum_{q \in R(k)} \mathrm{blk}_{q,t}, \quad \forall\, k \in M,\ t \in \mathcal{T} \tag{A32}$$

$$\mathrm{sys}_t = \sum_{k \in M} \mathrm{mach}_{k,t}, \quad \forall\, t \in \mathcal{T} \tag{A33}$$

$$\mathrm{PP} \ge \mathrm{sys}_t, \quad \forall\, t \in \mathcal{T} \tag{A34}$$

$$\mathrm{CT} \ge s_{i,q} + Ta_i - \mathcal{M}^{\mathrm{big}} \cdot (1 - z_{i,q}), \quad \forall\, i \in O, \forall\, q \in B \tag{A35}$$

The objective functions (A1) and (A2) minimise CT and PP, respectively. Constraint (A3) guarantees each task is assigned to exactly one block. Constraint (A4) enforces precedence relationships between tasks. Constraint (A5) prevents incompatible tasks from being assigned to the same block. Constraints (A6) and (A7) enforce workstation inclusion and exclusion rules for task assignments, respectively. Constraint (A8) ensures tasks are assigned only to technologically compatible workstations. Constraint (A9) limits the number of tasks per block to *Mtb*.

Constraint (A10) suppresses phantom start times for unassigned task–block pairs. Constraint (A11) anchors the first block on each workstation to start at time zero. Constraints (A12) and (A13) define the block start time BS_q via big-M linearisation. Constraint (A14) defines the block end

time BE_q as the latest task completion within the block. Constraints (A15) and (A16) define the block activity indicator a_q, which equals 1 if and only if block q contains at least one task. Constraints (A17) and (A18) enforce contiguous sequential scheduling of active blocks on the same workstation, eliminating idle time between them. Constraint (A19) prevents gaps in block usage by requiring earlier blocks to be filled before later ones. Constraints (A20) and (A21) set the duration of any inactive block to zero by equating its start and end time.

Constraints (A22)–(A29) linearise the activity indicator $u_{t,q,i}$, which equals 1 if and only if task i is assigned to block q and is being processed at time t. This is achieved via two auxiliary binaries: $w^1_{t,q,i}$ (not-yet started) and $w^2_{t,q,i}$ (already finished), whose combined logic forces $u_{t,q,i} = 1$ exactly over the active window $s_{i,q} \leq t < s_{i,q} + Ta_i$. Constraint (A30) computes the power contribution of each active task as $Tp_i \cdot u_{t,q,i}$. Constraints (A31), (A32), and (A33) aggregate power bottom-up through the task, block, workstation, and system levels. Constraint (A34) determines the PP as the maximum system power across all time slots. Constraint (A35) computes the CT as the maximum completion time of any assigned task across all blocks.

Appendix 2. Sensitivity analysis of reward function weights

To assess the sensitivity of NN_MOVNS to the reward function weighting coefficients (w_1, w_2, w_3), five weight configurations were evaluated on five representative instances drawn from the main experimental testbed: T4 (JACKSON, 11 tasks), T14 (BUXEY, 29 tasks), T19 (GUNTHER, 35 tasks), T22 (KILBRID, 45 tasks), and T25 (KILBRID, 45 tasks). Each configuration was executed ten times under the same time limits and parameter settings used in the main experiments, with only the reward weights varied. The five configurations are: C1 (baseline, w_1=0.8, w_2=0.9, w_3=0.3), C2 (equal, 0.5, 0.5, 0.5), C3 (objective-heavy, 0.3, 0.3, 0.9), C4 (diversity-heavy, 0.5, 1.0, 0.1), and C5 (Pareto-heavy, 1.0, 0.5, 0.1). Table A1 reports the mean HV_{rel} and IGD^+ values across the ten independent runs.

The baseline configuration C1 achieves the best mean HV_{rel} and the lowest mean IGD^+ across all five tested instances. This consistently strong performance supports the baseline reward-weight configuration adopted in the main computational experiments.

Configurations that preserve relatively high weights on the Pareto expansion and diversity components exhibit only moderate performance reductions. Under C4 (diversity-heavy), HV_{rel} decreases by 0.8% on T4 (from 0.988 to 0.980), by 1.2% on T14 (from 0.918 to 0.907), by 0.9% on T19 (from 0.992 to 0.983), by 0.9% on T22 (from 0.993 to 0.984), and by 1.4% on T25 (from 0.956 to 0.943). Under C2 (equal weights), the reductions are slightly larger but remain below 3% across all instances. Under C5 (Pareto-heavy), the largest HV_{rel} reduction is 2.5% on T25 (from 0.956 to 0.932). These results indicate that the algorithm retains competitive performance under the moderate weight perturbations examined in this analysis. Among these configurations, C4 is consistently the second-best, which aligns with the multi-objective design rationale: prioritising solution spread along the Pareto front yields better approximation quality than emphasising archive cardinality or objective improvement alone.

In contrast, configuration C3 (objective-heavy, w_3=0.9) produces the largest performance degradation across all five instances. On T4, IGD^+ increases from 1.100 to 2.150 (+95.5%). On T14, IGD^+ rises from 2.860 to 5.480 (+91.6%). On T19, IGD^+ increases from 1.021 to 2.340 (+129.2%). On T22, IGD^+ rises from 2.129 to 4.970 (+133.4%). On T25, IGD^+ increases from 1.373 to 3.280 (+138.9%). The corresponding HV_{rel} reductions range from 3.8% on T19 (from 0.992 to 0.954) to 5.7% on T25 (from 0.956 to 0.901). This outcome is consistent with the multi-objective nature of the problem: when single-objective improvement is excessively prioritised, the DQN agent learns to favour operators that improve one objective at the expense of Pareto front coverage and diversity, resulting in narrower and less well-distributed approximation sets.

Overall, the analysis confirms that the selected weight configuration (0.8, 0.9, 0.3) provides an effective balance among the three reward components. The algorithm tolerates moderate deviations without substantial quality loss, but heavily skewing the reward toward direct objective improvement is detrimental, reinforcing the design rationale of prioritising archive expansion and solution spread in the multi-objective search context.

Table A1. Sensitivity of NN_MOVNS performance to reward weight configurations across five representative instances. Best values per instance are shown in bold.

		T4 – JACKSON		T14 – BUXEY		T19 – GUNTHER		T22 – KILBRID		T25 – KILBRID	
Config	(w_1, w_2, w_3)	HV_{rel}	IGD^+	HV_{rel}	IGD^+	HV_{rel}	IGD^+	HV_{rel}	IGD^+	HV_{rel}	IGD^+
C1 (baseline)	(0.8, 0.9, 0.3)	**0.988**	**1.100**	**0.918**	**2.860**	**0.992**	**1.021**	**0.993**	**2.129**	**0.956**	**1.373**
C2 (equal)	(0.5, 0.5, 0.5)	0.974	1.380	0.900	3.620	0.975	1.450	0.975	2.840	0.935	1.890
C3 (obj.-heavy)	(0.3, 0.3, 0.9)	0.941	2.150	0.866	5.480	0.954	2.340	0.928	4.970	0.901	3.280
C4 (div.-heavy)	(0.5, 1.0, 0.1)	0.980	1.240	0.907	3.240	0.983	1.180	0.984	2.450	0.943	1.620
C5 (Pareto-heavy)	(1.0, 0.5, 0.1)	0.968	1.470	0.894	3.810	0.971	1.520	0.966	3.080	0.932	2.010