

---

# DATALESS NEURAL NETWORKS FOR RESOURCE-CONSTRAINED PROJECT SCHEDULING

---

A PREPRINT

**Marc Bara Iniesta**

marc.bara.iniesta@gmail.com

ORCID: 0009-0005-1480-5760

## ABSTRACT

Dataless neural networks represent a paradigm shift in applying neural architectures to combinatorial optimization problems, eliminating the need for training datasets by encoding problem instances directly into network parameters. Despite the pioneering work of Alkhouri et al. (2022) demonstrating the viability of dataless approaches for the Maximum Independent Set problem, our comprehensive literature review reveals that no published work has extended these methods to the Resource-Constrained Project Scheduling Problem (RCPSP). This paper addresses this gap by presenting the first dataless neural network approach for RCPSP, providing a complete mathematical framework that transforms discrete scheduling constraints into differentiable objectives suitable for gradient-based optimization. Our approach leverages smooth relaxations and automatic differentiation to unlock GPU parallelization for project scheduling, traditionally a domain of sequential algorithms. We detail the mathematical formulation for both precedence and renewable resource constraints, including a memory-efficient dense time-grid representation. Implementation and comprehensive experiments on PSPLIB benchmark instances (J30, J60, and J120) are currently underway, with empirical results to be reported in an updated version of this paper.

**Keywords** Dataless Neural Networks · Resource-Constrained Project Scheduling · RCPSP · Combinatorial Optimization · GPU Acceleration · PSPLIB

## 1 Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) stands as one of the most fundamental and widely studied problems in operations research and project management. Given its NP-hard nature and practical importance across industries, RCPSP has attracted diverse solution approaches ranging from exact methods to metaheuristics. Recently, neural network-based methods have emerged as a promising direction, yet all existing approaches share a critical limitation: they require extensive training data.

A revolutionary alternative emerged with the work of Alkhouri et al. [1], who introduced dataless neural networks for combinatorial optimization. Their approach eliminates the need for training datasets entirely, instead encoding the problem instance directly into network parameters and solving it through gradient-based optimization. This paradigm shift addresses fundamental limitations of learning-based methods, particularly their poor generalization to problem instances with different structural characteristics than their training data.

Despite this promising precedent and the extensive research on RCPSP, our comprehensive literature review reveals a striking gap: no published work has applied dataless neural networks to the Resource-Constrained Project Scheduling Problem. This absence is particularly notable given that RCPSP possesses structural properties that seem well-suited to the dataless approach, including well-defined constraints and established continuous relaxations.

This paper makes two primary contributions. First, we present a systematic survey examining why this gap exists and analyzing the current state of neural network approaches to RCPSP, all of which rely on training data. Second, and more importantly, we propose the first dataless neural network approach for RCPSP, providing a complete mathematical framework that shows how project scheduling can benefit from gradient-based optimization and GPU acceleration.

Our approach transforms the discrete scheduling problem into a continuous optimization problem through carefully designed smooth relaxations, enabling the use of automatic differentiation and parallel computation.

We detail how both precedence constraints and renewable resource constraints can be encoded in differentiable form, with a memory-efficient formulation that scales to large instances while maintaining minimal GPU memory footprint. Standard benchmark instances from PSPLIB, particularly the J30, J60, and J120 sets, provide the testing ground for our approach, with implementation currently in progress and empirical validation to follow in an updated version of this paper.

## 2 Background

### 2.1 The Resource-Constrained Project Scheduling Problem

The RCPSP can be formally defined as follows: given a project consisting of  $n$  activities, where each activity  $i$  has a duration  $d_i$  and requires  $r_{ik}$  units of renewable resource type  $k$  during its execution, the objective is to determine start times  $S_i$  for all activities such that the project makespan is minimized while respecting precedence constraints and resource availabilities.

The problem is subject to two fundamental types of constraints. Precedence constraints ensure that an activity cannot start until all its predecessors have completed, formally expressed as  $S_j \geq S_i + d_i$  for all  $(i, j)$  in the precedence relation. Resource constraints ensure that at any time point, the total resource consumption does not exceed availability:  $\sum_{i \in A(t)} r_{ik} \leq R_k$  for all time points  $t$  and resource types  $k$ , where  $A(t)$  represents the set of activities in progress at time  $t$  and  $R_k$  is the availability of resource type  $k$ .

The PSPLIB (Project Scheduling Problem Library) has become the standard benchmark repository for RCPSP research. The j30, j60, and j120 instance sets contain projects with 30, 60, and 120 activities respectively, providing a range of problem sizes with varying resource constraints and network complexities. These instances have enabled systematic comparison of solution methods over decades of research.

### 2.2 Dataless Neural Networks: A Paradigm Shift

Traditional neural network approaches to combinatorial optimization follow a learning paradigm: they train on numerous problem instances to learn patterns that generalize to new instances. Dataless neural networks fundamentally reject this approach. Instead of learning from data, they encode a specific problem instance directly into the network architecture or parameters, then use gradient-based optimization to find solutions.

The key insight is transforming the discrete combinatorial problem into a continuous optimization problem that can be solved via gradient descent. For a given problem instance, the network parameters themselves become the decision variables, and the loss function encodes both the objective and constraints. This approach offers several advantages: it eliminates the need for training data, avoids generalization issues, and can leverage modern automatic differentiation frameworks for efficient optimization.

Alkhouri et al. [1] demonstrated this concept on the Maximum Independent Set problem, showing that carefully designed continuous relaxations and loss functions enable gradient descent to find high-quality solutions to NP-hard problems without any training phase.

To illustrate this paradigm more concretely, consider how dataless neural networks encode problem structure: the network connectivity is directly derived from the problem instance (e.g., a graph’s topology), with fixed weights and bias terms explicitly set based on properties such as adjacency relations. The only trainable parameters are those that ultimately encode the solution. For instance, in the Maximum Independent Set problem, the network is fed a constant input—typically an all-ones vector—and employs fixed weights encoding the graph structure together with trainable parameters constrained to lie within  $[0, 1]$ . By minimizing a carefully designed loss function via backpropagation, local minima correspond to valid solutions once the learned parameters are appropriately thresholded.

## 3 Literature Review: The State of Neural Networks for RCPSP

### 3.1 Current Data-Driven Approaches

Our comprehensive review of neural network applications to RCPSP reveals a consistent pattern: all existing methods fundamentally rely on training data. These approaches can be broadly categorized into three paradigms, each requiring substantial datasets for effective operation.

First, supervised learning methods train neural networks to predict good scheduling decisions based on problem features. These typically use multilayer feed-forward neural networks (MLFNNs) or convolutional neural networks (CNNs) trained on thousands of solved instances from benchmark sets like PSPLIB [2]. The networks learn to map from problem characteristics (activity durations, resource requirements, network structure) to priority rules or direct schedule constructions.

Second, reinforcement learning approaches frame RCPSP as a sequential decision-making problem, where agents learn policies for selecting which activity to schedule next. These methods require extensive training through interaction with scheduling environments, often millions of episodes, to develop effective policies.

Third, hybrid approaches combine neural networks with traditional optimization techniques, using networks to guide search heuristics or predict promising solution regions. Even these hybrid methods require training phases to learn effective guidance strategies.

Research by Özkan and Gülçiçek [3] exemplifies the current paradigm. They design artificial neural networks that learn from datasets constructed using various scheduling heuristics, essentially teaching the network to mimic and potentially improve upon existing algorithms. While showing promise within their training distribution, these methods face the fundamental limitation that plagues all data-driven approaches: performance degradation on instances that differ structurally from their training data.

Recent advances have explored deep learning architectures for RCPSP. Zhang et al. [4] employ graph neural networks with reinforcement learning, training models on PSPLIB instances that generalize across problem scales. Similarly, a recent IEEE study [5] demonstrates that GNN-based approaches trained on small instances can effectively scale to larger problems while outperforming traditional priority rules.

Priority rule selection through neural networks has emerged as another direction. Golab [2] develops neural networks that learn to select appropriate priority rules based on project state parameters, avoiding the need to generate multiple solution populations.

Hybrid approaches combining neural networks with other metaheuristics have shown particular promise. Agarwal et al. [6] demonstrate that combining genetic algorithms with neural network local search outperforms either method independently. For construction-specific applications, researchers have combined neural networks with metaheuristics to address multi-mode RCPSP variants [7]. Additionally, Jędrzejowicz and Ratajczak-Ropel [8] use reinforcement learning to coordinate multiple optimization agents solving RCPSP instances.

Despite these advances, all approaches fundamentally rely on training data—either from solved instances, simulation trajectories, or evolutionary populations—highlighting the absence of truly dataless methods.

The closest related work comes from Liu et al. [9], who developed a differentiable scheduling framework that operates without training data. However, their work targets scheduling in chip design and high-performance computing, focusing on latency and dependency constraints rather than the renewable resource constraints central to project management. The mathematical structures and constraint types differ substantially, preventing direct application to RCPSP.

## 4 Analysis: Why the Gap Exists

### 4.1 Technical Challenges

Several technical factors may explain why dataless neural networks have not yet been applied to RCPSP. The problem’s constraint structure presents unique challenges for continuous relaxation. Unlike the Maximum Independent Set problem, which involves only pairwise constraints between vertices, RCPSP requires modeling temporal dependencies and cumulative resource consumption over time.

Creating differentiable formulations of precedence constraints is conceptually straightforward through soft ordering penalties. However, resource constraints pose a more complex challenge. They involve checking that cumulative resource usage remains within bounds across all time points, requiring either discretization of time or sophisticated continuous-time formulations. The interaction between precedence and resource constraints further complicates the design of effective loss functions.

Additionally, the typical RCPSP objective of minimizing makespan requires identifying the maximum completion time across all activities, which can create challenging optimization landscapes for gradient-based methods. Designing smooth approximations that maintain good gradient flow while accurately representing the objective remains an open challenge.

## 4.2 Research Community Factors

Beyond technical challenges, the structure of research communities may contribute to this gap. The project scheduling community has deep expertise in problem-specific algorithms, constraint programming, and operations research techniques. Meanwhile, the neural combinatorial optimization community has focused primarily on problems with simpler constraint structures or those arising in machine learning contexts.

The interdisciplinary expertise required to successfully apply dataless neural networks to RCPSP—combining understanding of project scheduling semantics, continuous optimization, and neural architecture design—may explain why this intersection remains unexplored.

## 5 A Dataless Neural Network Approach for RCPSP

Building on the successful application of dataless neural networks to the Maximum Independent Set problem by Alkhouri et al. [1], we propose a novel approach for solving the Resource-Constrained Project Scheduling Problem without requiring any training data. Our method encodes RCPSP instances directly into neural network parameters and uses gradient-based optimization to find high-quality schedules.

### 5.1 Solution Representation

Our approach represents activity start times as trainable parameters in a neural network, specifically using a softplus activation to ensure non-negativity:

$$s_i = \text{softplus}(\theta_i) - \min_j (\text{softplus}(\theta_j)) \quad (1)$$

where  $\theta_i$  are the raw trainable parameters and  $s_i$  are the resulting start times anchored at zero. This continuous parameterization enables gradient-based optimization while maintaining feasible time values.

This architecture represents a fundamental departure from traditional neural networks. In our dataless approach, the network contains exactly  $n$  trainable parameters  $\{\theta_1, \theta_2, \dots, \theta_n\}$ , one for each activity in the scheduling problem. There are no hidden layers, no weight matrices, and no feature extraction—just a direct parameterization where each  $\theta_i$  corresponds to activity  $i$ .

The softplus transformation serves purely to maintain feasibility (non-negative times) and differentiability, not to learn features. When we optimize these parameters through gradient descent, we are directly optimizing the schedule itself: each gradient step adjusts the raw parameters  $\theta$ , which immediately translates to adjusted start times  $s$  through the softplus mapping. The network architecture is entirely determined by the problem structure, providing a differentiable parameterization of the solution space rather than a learned input-output mapping.

### 5.2 Objective Function

The primary objective is to minimize the project makespan (total duration). We employ a smooth maximum approximation using log-sum-exp:

$$\text{makespan} = \frac{1}{\beta} \log \sum_{i=1}^n \exp(\beta \cdot (s_i + d_i)) \quad (2)$$

where  $\beta$  is a sharpness parameter that increases during optimization to better approximate the true maximum.

### 5.3 Precedence Constraints

Precedence relationships require that each activity completes before its successors can begin. We encode violations as:

$$v_{ij} = \max(0, s_i + d_i - s_j) \quad \forall (i, j) \in E \quad (3)$$

where violations are measured in time units. A positive  $v_{ij}$  indicates that activity  $i$  finishes  $v_{ij}$  time units after activity  $j$  starts, violating the precedence constraint.

To maintain gradient flow even for satisfied constraints, we employ a smooth penalty function:

$$\text{penalty}(v) = \begin{cases} \frac{1}{2}v^2 & \text{if } v < 1 \\ v - \frac{1}{2} & \text{otherwise} \end{cases} \quad (4)$$

This quadratic-linear hybrid ensures active gradients near constraint boundaries while preventing gradient explosion for large violations. The quadratic region ( $v < 1$ ) provides increasingly strong gradients as violations approach zero, encouraging precise constraint satisfaction. The linear region prevents the gradient magnitude from growing unboundedly for large violations, maintaining numerical stability.

The total precedence penalty is computed as the mean of individual violation penalties:

$$\mathcal{L}_{\text{precedence}} = \frac{1}{|E|} \sum_{(i,j) \in E} \text{penalty}(v_{ij}) \quad (5)$$

where  $|E|$  is the number of precedence constraints and the penalty function is applied to each violation  $v_{ij}$  individually.

## 5.4 Resource Constraints

Renewable resource constraints require that the total resource usage never exceeds capacity at any time point:  $\sum_{i \in A(t)} r_{ik} \leq R_k$  for all time points  $t$  and resource types  $k$ , where  $A(t)$  is the set of activities in progress at time  $t$ ,  $r_{ik}$  is the amount of resource  $k$  required by activity  $i$ , and  $R_k$  is the capacity of resource  $k$ . To encode these constraints differentiably, we employ a dense time-grid formulation.

### 5.4.1 Dense Time-Grid Formulation

For a project with planning horizon  $T$  (an upper bound on project completion time), we create a time grid  $\{0, 1, \dots, T-1\}$  and compute which activities are running at each time point:

$$\rho_{it} = \begin{cases} 1 & \text{if } s_i \leq t < s_i + d_i \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

To maintain differentiability, we implement this using tensor operations:

$$\rho_{it} = \sigma(t \geq s_i) \cdot \sigma(t < s_i + d_i) \quad (7)$$

where  $\sigma$  represents a differentiable approximation of the step function.

The total resource usage at time  $t$  for resource  $k$  is then:

$$u_{tk} = \sum_{i=1}^n \rho_{it} \cdot r_{ik} \quad (8)$$

In matrix form, let  $\mathbf{P}$  be the  $n \times T$  running-activity matrix with entries  $\rho_{it}$ , and  $\mathbf{R}$  be the  $n \times K$  resource requirement matrix with entries  $r_{ik}$ . The complete resource usage matrix  $\mathbf{U} = \mathbf{P}^T \mathbf{R}$  provides all resource usage values  $u_{tk}$  in a single operation, enabling efficient computation within automatic differentiation frameworks.

### 5.4.2 Normalized Resource Penalties

To handle resources with different scales uniformly, we normalize violations by capacity. Let  $o_{tk}$  denote the normalized resource overshoot for resource  $k$  at time  $t$ :

$$o_{tk} = \max\left(0, \frac{u_{tk} - C_k}{C_k}\right) \quad (9)$$

where  $C_k$  is the capacity of resource  $k$ . This normalization ensures that a 10% overuse is penalized equally regardless of whether it's 11/10 workers or 110/100 machines. The total resource penalty becomes:

$$\mathcal{L}_{\text{resource}} = \frac{1}{T \cdot K} \sum_{t=0}^{T-1} \sum_{k=1}^K o_{tk}^2 \quad (10)$$

Using the mean rather than sum prevents numerical overflow and makes the penalty magnitude independent of project duration.

### 5.4.3 Integrated Objective Function

The complete loss function combines makespan, precedence penalties, and resource penalties:

$$\mathcal{L} = \text{makespan} + \lambda_p \cdot \mathcal{L}_{\text{precedence}} + \lambda_r \cdot \mathcal{L}_{\text{resource}} \quad (11)$$

where  $\lambda_p$  and  $\lambda_r$  are penalty coefficients.

#### 5.4.4 Adaptive Penalty Management

When penalty coefficients are fixed, the optimization may converge slowly on some instances or oscillate without finding feasible solutions on others. To address this, an adaptive mechanism can adjust  $\lambda_p$  and  $\lambda_r$  during optimization based on constraint satisfaction progress.

The mechanism monitors violation history over a lookback window and applies three types of adjustments:

- **Escalation:** When constraint violations remain large or stable, the corresponding penalty coefficient increases to apply stronger pressure toward feasibility.
- **Reduction:** When violations become small and stable, the penalty coefficient can be reduced to allow the optimizer to focus on improving the objective.
- **Recovery:** If violations begin trending upward after initial improvement, aggressive penalty increases help restore progress toward feasibility.

This adaptive approach helps maintain consistent convergence behavior across problem instances with varying characteristics, without requiring manual tuning of penalty parameters for each instance.

#### 5.4.5 Memory Efficiency

Despite the dense representation, memory requirements remain minimal:

Instance	Activities $\times$ Horizon	Memory (forward)	Memory (w/ gradients)
J30	$32 \times 158$	26 KB	52 KB
J60	$60 \times 250$	60 KB	120 KB
J120	$120 \times 400$	192 KB	384 KB

These requirements are negligible for individual projects. The dense matrix  $\mathbf{P}$  of size  $n \times T$  requires approximately  $8nT$  bytes with gradients (assuming float32). This linear scaling enables handling large-scale applications:

- Enterprise project portfolio:  $10,000 \text{ activities} \times 2,000 \text{ days} \approx 160 \text{ MB}$
- Construction mega-program:  $50,000 \text{ activities} \times 5,000 \text{ days} \approx 2 \text{ GB}$
- National infrastructure planning:  $200,000 \text{ activities} \times 10,000 \text{ days} \approx 16 \text{ GB}$

Modern GPUs with 24-80 GB of memory can thus handle even national-scale project portfolios, while CPU systems with hundreds of gigabytes of RAM could manage virtually unlimited project complexity. The memory efficiency ensures that storage will not be the limiting factor for large-scale applications.

#### 5.4.6 Addressing Optimization Challenges

The discrete time grid introduces potential challenges for gradient-based optimization, particularly the emergence of flat regions in the loss landscape where multiple solutions yield identical objective values. When activities can shift by full time units without affecting resource usage patterns, traditional gradient descent may struggle to find improving directions.

We address these challenges through several complementary strategies. Small random perturbations to activity start times can help break symmetries and escape plateau regions during optimization. The continuous softplus activation function maintains smooth gradients even near constraint boundaries, preventing the optimization from getting trapped in non-differentiable regions. Additionally, normalizing resource violations by their respective capacities ensures that no single constraint type dominates the optimization dynamics, maintaining balanced progress across all problem requirements.

These design choices work synergistically to maintain reliable convergence properties. The same hyperparameter configuration that proves effective for precedence-only instances is expected to generalize to resource-constrained problems with minimal adjustment, though empirical validation across the full range of problem instances remains ongoing.

### 5.5 Implementation and Preliminary Results

We have implemented the precedence-constrained version of our dataless scheduler in PyTorch, leveraging automatic differentiation for efficient gradient computation. The implementation includes:

- Vectorized constraint evaluation for computational efficiency
- Adam optimizer with learning rate reheating to escape local minima
- Feasible candidate tracking to ensure solution quality
- Early stopping based on constraint satisfaction thresholds

Preliminary experiments on precedence-only project networks demonstrate that our approach successfully finds optimal makespans. Figure 1 shows convergence behavior on instance j301\_1 (precedence constraints only), where the solver achieves constraint satisfaction within 1500 epochs. Across different instances, convergence typically occurs within 400-3000 epochs, depending on problem complexity.

Detailed performance metrics across all benchmark instances, including the full RCPSP with resource constraints, will be included in an updated version of this paper once our experiments are complete.

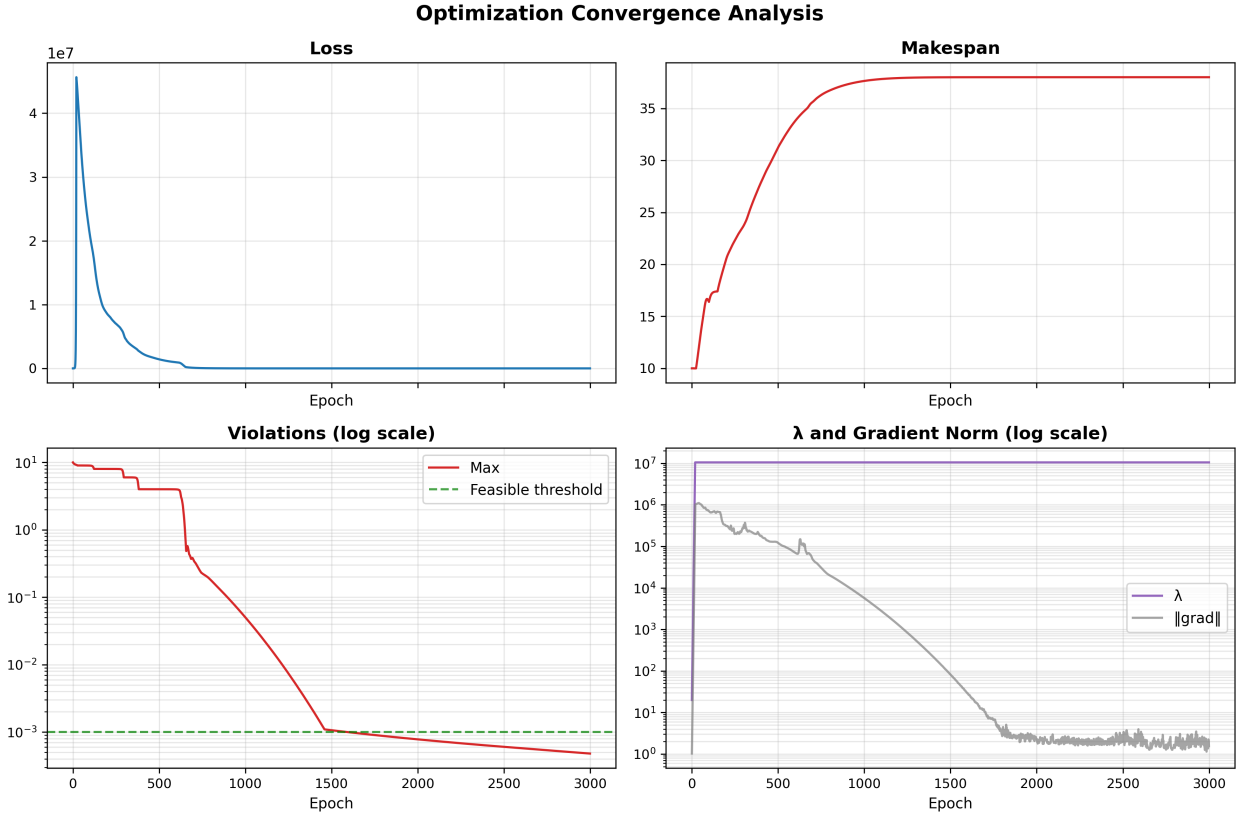


Figure 1: Convergence behavior on PSPLIB instance j301\_1 with precedence constraints only. The solver finds a feasible solution (violations  $< 10^{-3}$ ) around epoch 1500, with the makespan stabilizing at exactly 38.0 time units. Note that a fixed penalty coefficient suffices for precedence-only problems— $\lambda$  remains at  $10^7$  after initial escalation.

## 5.6 Ongoing Experimental Validation

We are currently conducting comprehensive experiments on the PSPLIB benchmark instances:

- **J30 dataset:** 480 instances with 30 activities and 4 renewable resources
- **J60 dataset:** 480 instances with 60 activities and 4 renewable resources
- **J120 dataset:** 600 instances with 120 activities and 4 renewable resources

Our experimental protocol includes:

1. Validation of the precedence-only solver against known optimal makespans

2. Integration of resource constraint handling using the dense time-grid approach
3. Hyperparameter robustness testing across the entire benchmark suite
4. Comparison with state-of-the-art RCPSP solvers

Experiments are currently in progress. Complete experimental results, including detailed performance metrics on all PSPLIB datasets, statistical analyses, and comprehensive comparisons with existing methods, will be reported in an updated version of this paper.

### 5.7 Technical Advantages

Our dataless approach offers several advantages over traditional learning-based methods:

- **No training data required:** Each instance is solved independently without relying on historical solutions
- **No generalization issues:** The method optimizes directly on the given instance rather than learning patterns
- **Interpretable optimization:** The gradient dynamics provide insights into constraint interactions
- **Minimal memory footprint:** Even for J120 instances, memory usage remains below 1MB
- **Native GPU optimization:** While some metaheuristics can be adapted for parallel execution, our approach is inherently based on matrix operations and automatic differentiation, making it naturally suited for GPU acceleration without requiring special parallel implementations

This work represents the first application of dataless neural networks to the Resource-Constrained Project Scheduling Problem, opening new avenues for neural combinatorial optimization in project management contexts.

## 6 Future Directions

Our proposed dataless neural network approach for RCPSP opens several avenues for future research. The most immediate extensions include applying the framework to RCPSP variants such as multi-mode scheduling (MRCPSP) and problems with generalized precedence relations.

The continuous relaxation strategy could also be adapted to related scheduling domains including job shop scheduling and vehicle routing with time windows. More broadly, the core methodology of encoding resource constraints into neural network parameters extends beyond traditional scheduling. The same mathematical framework could address cloud computing resource allocation (where activities represent compute jobs and resources represent GPUs/CPUs), healthcare operations scheduling (surgical procedures competing for operating rooms and staff), and logistics optimization (deliveries constrained by vehicle capacity). Each domain presents unique constraint patterns that could benefit from the dataless approach’s ability to optimize directly on problem instances without training data.

From a theoretical perspective, establishing convergence guarantees and approximation bounds for the continuous relaxation remains an open challenge. Understanding when and why the gradient-based optimization finds high-quality solutions would provide valuable insights for improving the method.

On the practical side, developing efficient rescheduling capabilities for dynamic project environments and integrating the approach with existing project management software would facilitate real-world adoption. By reformulating scheduling as matrix operations, our dataless approach unlocks the true potential of GPU parallelism, making it possible to optimize mega-scale instances—such as national infrastructure programs or global supply chain coordination—that remain intractable for traditional methods.

These directions suggest that dataless neural networks may provide a general framework for combinatorial scheduling problems beyond the specific case of RCPSP, warranting continued investigation of this paradigm.

## 7 Conclusion

This paper has identified and addressed a significant gap in the intersection of dataless neural networks and project scheduling. Our comprehensive survey revealed that despite the success of dataless approaches for combinatorial problems like Maximum Independent Set, no prior work has applied these methods to the Resource-Constrained Project Scheduling Problem—a fundamental challenge in operations research with widespread practical applications.

We have presented the first dataless neural network approach for RCPSP, providing a complete mathematical framework that encodes project scheduling instances directly into network parameters for gradient-based optimization. The method



eliminates training data requirements and their associated generalization issues while maintaining the computational advantages of continuous optimization. Through smooth relaxations of objectives and constraints, our approach enables efficient optimization via automatic differentiation.

Our mathematical framework encompasses both precedence and renewable resource constraints through a memory-efficient dense time-grid formulation. Crucially, by transforming the traditionally discrete RCPSP into matrix operations, we unlock GPU parallelization for project scheduling—enabling optimization of instances far beyond the reach of traditional sequential methods. Preliminary experiments on precedence-constrained instances demonstrate convergence to optimal solutions, with full RCPSP experiments on PSPLIB benchmarks currently underway.

This work establishes dataless neural networks as a viable paradigm for project scheduling optimization, opening new research directions at the intersection of neural computation and combinatorial optimization. By demonstrating that complex scheduling constraints can be effectively encoded in differentiable form, we hope to inspire further applications of the dataless paradigm throughout operations research.

## References

- [1] Ismail R. Alkhouri, George K. Atia, and Alvaro Velasquez. A differentiable approach to the maximum independent set problem using dataless neural networks. *Neural Networks*, 155:168–176, 2022.
- [2] Amir Golab. *Automatisation de la planification dynamique de projet à ressource limitée par l’intelligence artificielle*. PhD thesis, Université de Bretagne Occidentale, Brest, France, 2023. Thèse présentée et soutenue à Pole numérique, UBO, Brest, le 18/09/2023. Unité de recherche : ISEN Yncréa Ouest, Brest.
- [3] Ömer Özkan and Ümit Gülççek. A neural network for resource constrained project scheduling programming. *Journal of Civil Engineering and Management*, 2015.
- [4] H. Zhang et al. Deep reinforcement learning for solving resource constrained project scheduling problems with resource disruptions. *Computers & Industrial Engineering*, 2023.
- [5] X. Zhao et al. A deep reinforcement learning approach for resource-constrained project scheduling. In *Proceedings of the IEEE Conference Publication*. IEEE, 2022.
- [6] A. Agarwal, S. Colak, and S. Erenguc. A neurogenetic approach for the resource-constrained project scheduling problem. *Computers & Operations Research*, 38(1):44–50, 2011.
- [7] J. Białek et al. A hybrid approach for solving multi-mode resource-constrained project scheduling problem in construction. *Open Engineering*, 9(1):7–13, 2019.
- [8] P. Jędrzejowicz and E. Ratajczak-Ropel. Reinforcement learning strategies for A-Team solving the resource-constrained project scheduling problem. *Neurocomputing*, 146:301–307, 2014.
- [9] Mingju Liu, Yingjie Li, Jiaqi Yin, Zhiru Zhang, and Cunxi Yu. Differentiable combinatorial scheduling at scale. *ArXiv*, 2024. arXiv preprint.
- [10] Ismail Alkhouri, Cedric Le Denmat, Yingjie Li, Cunxi Yu, Jia Liu, Rongrong Wang, and Alvaro Velasquez. Dataless quadratic neural networks for the maximum independent set problem. *ArXiv*, 2024. arXiv preprint.