Graph Neural Networks for Inland Waterway Ship Scheduling

Master Thesis

Nahom Tsehaie

Delft University of Technology

TU Delft

Graph Neural Networks for Inland Waterway Ship Scheduling

By

Nahom Tsehaie

Master Thesis

in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology

to be defended publicly on Monday, December 18, 2023, at 01:30 PM

Student number:	4357728	
MSc track:	Transport Engineering and logistics	
Report number:	2023.MME.8885	
Supervisor:	ir. MSc. Peter Wenzel	
Thesis committee:	dr. Frederik Schulte,	Tu Delft committee Chair, Faculty of 3mE
	ir. MSc Peter Wenzel,	Tu Delft committee member, Faculty of 3mE
	ir. MSc Xinyu Tang,	Tu Delft committee member, Faculty of 3mE
Date:	December 2023	

TU Delft

Preface

As I draw the curtains on my time as a student in Mechanical Engineering at the Delft University of Technology, a whirlwind of memories, challenges, and growth comes to mind. The skills and knowledge I acquired during this period were instrumental in navigating the multifaceted challenges of this thesis.

However, every journey is seldom travelled alone, no matter how personal. I owe a debt of gratitude to several individuals who have been instrumental in this journey.
First of all, I would like to express my profound appreciation to my supervisor, Peter Wenzel. Your mentorship has been invaluable. From our biweekly brainstorming sessions to discussions on research outcomes and future directions, your insights have constantly illuminated my path. I am particularly grateful for the pragmatic advice on approaching the thesis and the invaluable tips for penning this report.
Frederik Schulte, a heartfelt thank you for presenting me with this intriguing and demanding thesis project. Your faith in my abilities and the opportunity you provided have been instrumental in my academic growth.
I also wish to extend my heartfelt thanks to Raka Jovanovic. Your feedback, suggestions, and encouragement have enriched my research and broadened my perspective. Collaborating with you has been both a pleasure and a privilege.
I am also deeply indebted to my fellow students and friends at the university. Their assistance has been crucial, be it in resolving minor hiccups or providing feedback. Their timely assistance, persistent questions, and invaluable feedback have played a critical role in refining this work.

However, this journey would not have been possible without the indomitable spirit of one person - my mother. She ventured as a young refugee from Eritrea, faced innumerable challenges, and single-handedly nurtured three children in the Netherlands, emphasising the power and importance of education. Her sacrifices, resilience, and unwavering belief in the power of education have been the bedrock upon which my siblings and I have built our lives. I dedicate this work to my mother, who is not just a hero in our eyes but an inspiration to many. You have taught us that with determination and hard work, no dream is too big. You are the true hero of this story!

In sum, this thesis is not merely a reflection of my academic endeavours but is intertwined with the contributions and faith of many. As I close this chapter, I carry forward not just knowledge but memories, lessons, and gratitude.

Nahom Tsehaie
Delft, December 2023

Summary

Inland waterway shipping represents a vital transportation medium, characterised by its unpredictability and varying nature. This research's principal objective is to address these inconsistencies by constructing a robust scheduling model tailored to waterway systems' specific needs and challenges. The model is enhanced with predictive analytics and optimisation methods to ensure efficient and reliable operations. Within this framework, the research delves deep into four distinct optimisation problems: the Travelling Salesman Problem (TSP), the Job Shop Scheduling Problem (JSSP), the Resource-Constrained Project Scheduling Problem (RCPSP), and the Waterway Ship Scheduled Problem (WSSP). The underlying theme connecting these problems is the application of Graph Neural Networks (GNN) as a tool to model these complex systems.

The Travelling Salesman Problem (TSP) is a cornerstone in combinatorial optimisation. Specifically, for this research, TSP serves as a means of understanding the challenges and intricacies of inland waterway networks. Employing the GNN model, the research provides insights into potential solutions for TSP within this context. When comparing various TSPLIB instances, the GNN model showcases its efficiency and potential for further refinement, especially in real-world routing and logistics.

Shifting the focus towards production planning, the Job Shop Scheduling Problem (JSSP) emerges as a pivotal problem. It aims to optimise the order and timing of tasks for various ships, ensuring minimal usage of time and resources. By implementing the GNN architecture, the research offers a fresh perspective on JSSP. When applied to real-world scenarios, it is evident that the model can predict optimal scheduling sequences, matching the actual time frames and resource allocation required, thereby promising significant advancements in maritime trade efficiency.

Diving deeper into operations research, the Resource-Constrained Project Scheduling Problem (RCPSP) surfaces as a challenge that focuses on optimising project schedules, considering resource constraints and task precedents. The research introduces a forward-thinking approach to address this problem, especially concerning cargo operations within port networks. The research promises heightened efficiency, reliability, and adaptability in Inland Waterway Transport (IWT) scheduling practices by integrating renewable resources, managing precedence relationships, and applying predictive analytics.

Lastly, the Waterway Ship Scheduling Problem (WSSP) centres on efficiently managing ship movements within a defined time frame, optimising the sequence and timing of vessels to minimise delays and maximise the utilisation of waterway infrastructure and resources. Building upon the foundational work of previous research, this problem was translated and redefined in the context of the Resource-Constrained Project Scheduled Problem. Using this foundation, two distinct RCPSP problems were formulated to reflect real-world scenarios, particularly emphasising the port of Duisburg. Drawing upon the results, it is clear that the GNN model demonstrates high efficiency and accuracy in addressing the WSSP. While traditional tools like OR TOOLS provided optimal results, the GNN model closely mirrored these benchmarks, solidifying its position as a formidable solution for complex scheduling issues, especially given its rapid computation times.

In conclusion, this research presents a cohesive understanding of various optimisation problems within the realm of inland waterway shipping, all while harnessing the power of GNNs. Through systematic exploration and application, the research underscores the potential of GNNs to revolutionise how we approach and solve these challenges, promising a future of enhanced efficiency and reliability in waterway shipping operations.

Acronyms

AIS Automatic Identification System

ANN Artificial Neural Network

API Application Programming Interface

BGNN Bidirectional Graph Neural Network

BN Batch Normalization

BP Back Propagation

CP-sat CP-sat model

COP Combinatorial Optimization Problem

CNN Convolutional Neural Network

CO Combinatorial Optimization

COP Combinatorial Optimization Problem

CP-sat Constraint Programming satisfiability problem

CVRP Capacitated Vehicle Routing Problem

DL Deep Learning

DRL Deep Reinforcement Learning

DP Dynamic Programming

GA Genetic Algorithm

GIN Graph Isomorphism Network

GNN Graph Neural Network

GLS Guided Local Search

IWT Inland Waterway Transport

ILP Integer Linear Programming

JSSP Job Shop Scheduling Problem

LSTM Long Short-Term Memory

MILP Mixed-Integer Linear Programming

MLP Multi-Layer Perceptron

MDVRP Multi-Depot Vehicle Routing Problem

MSE Mean Square Error

NN Neural Network

NP-hard Non-deterministic Polynomial-time hard

PDR Priority Dispatching Rule

PSPLIB Library related to RCPSP

RCPSP Resource-Constrained Project Scheduling Problem

TSPLIB Traveling Salesman Problem Library

TSP Traveling Salesman Problem

VTS Vessel Traffic Service

VRPs Vehicle Routing Problems

WSSP Waterway Ship Scheduling Problem

List of Figures

List of Tables

Contents

1 Introduction

1.1 Waterway ship scheduling

Organising and coordinating the movement of vessels and their cargo on inland waterways is vital to waterway ship scheduling. This is a crucial element of the inland transportation system, a primary means of transporting large cargo. Inland waterway transport underpins the economic growth of regions near rivers and the optimal design of industrial facilities. The Rhine Basin, for instance, houses around 15% of the total populace of the 27 European Union nations and contributes about 18.7% of the EU-27's total GDP (T. Notteboom et al., 2020). Over the years, this mode of transport has gained traction and has become a robust alternative to road and rail systems. In China, for instance, the proportion of cargo transported via inland waterways surged from 10% in 1990 to 30% in 2018 (Tan et al., 2018). Across the EU-27, there is a network of around 30,000 km of rivers and canals facilitating the transportation of goods such as petroleum, construction materials, food products, containers, and metals, with the Netherlands relying on them for almost half of its total cargo movement (Hofbauer & Putz, 2020).

Ensuring timely and trustworthy waterway ship scheduling can significantly drive down transport expenses and improve the efficiency of the entire supply chain. However, factors such as unpredictable weather patterns, changes in water level, and sparse shipping data add layers of complexity. Researchers have been working with cutting-edge technology, such as predictive analytics and optimisation techniques, to enhance the dependability and efficiency of scheduling (Fan et al., 2020). With the growing number of ports globally, there is an increase in the variety and volume of ships utilising inland waterways. This surge increases traffic complexity, escalates navigation hazards, and enlarges accident risks (Sirimanne et al., 2019). Therefore, developing and refining tools to analyse the complexities of traffic flow becomes crucial to enhancing navigational safety.

The rise in maritime activities requires port operators to minimise ship waiting times in their facilities, not only to retain or expand their market share but also to limit emissions (Verstichel, De Causmaecker, et al., 2014b). A staggering 93.6% of schedule delays can be traced back to the port entry and terminal functions (T. E. Notteboom, 2006). This aspect becomes even more pronounced in ports like Shanghai, where numerous vessels traverse the Yangtze Delta channels daily. To avoid congestion or bottlenecks, maximising waterway efficiency becomes pivotal (T. E. Notteboom, 2006). The significance of utilising inland waterways for transportation at container hubs is highlighted by their increasing compatibility with other transport systems within multimodal freight networks, especially when considering the challenges posed by global economic interconnectivity (Sirimanne et al., 2019).

Delving into maritime traffic characteristics is foundational for improving navigational safety (Thieme et al., 2018). Over the past two decades, efforts have concentrated on decoding and categorising traffic flow dynamics. Traffic models can be broadly bucketed into macroscopic and microscopic types, emphasising the varied motion behaviours across transport systems (Wen et al., 2015).

1.2 Research Gap

Inland waterway transportation is marked by significant unpredictability due to factors such as fluctuating water levels and changing weather conditions (Jonkeren et al., 2014). These variations complicate route planning and optimisation, leading to potential delays and inefficiencies that can burden shipping entities (Jonkeren et al., 2014). Additionally, the scarcity of reliable information on cargo movement and volume hinders precise demand prediction and resource distribution. This can cause resource mismanagement, operational challenges, and financial implications (Jonkeren et al., 2014).

Precisely, inland water transportation must navigate challenges distinct from other transport methods, such as manoeuvring through confined channels and locks, affecting vessel size and thus cargo capacity (Defryn et al., 2021). With these complexities, there is an increasing demand for novel strategies to enhance the efficacy and trustworthiness of this shipping mode (Jonkeren et al., 2007). Our research seeks to formulate a scheduling model that recognises specific features of the waterways and employs predictive and optimisation tools for enhanced performance.

Effective management of inland waterways demands sophisticated models capable of anticipating shipping trends, considering unpredictable waterway conditions and external influences like traffic and lock timings Nelson et al., 2015; Nur et al., 2020. Data-based solutions can significantly improve the consistency of waterway operations Nur et al., 2020. Real-time data on water conditions and traffic can inform dynamic route and timetable adjustments, minimising disruptions and costs. Moreover, adopting emerging technologies like machine learning and automation can refine the scheduling process, minimising human errors and considering multiple factors like ship dimensions and traffic dynamics (Gutierrez-Franco et al., 2021).

In essence, enhancing inland waterway operations requires a combination of advanced forecasting tools, data-centric strategies, and innovative technology, all working in tandem to improve the dependability and productivity of the sector (Giusti et al., 2019). This study seeks to tackle these issues by crafting a scheduling framework tailored to the distinct characteristics of inland waterways, integrating predictive analytics and optimisation methods to enhance efficiency and dependability.

1.3 Contribution

This thesis introduces an approach using graph neural networks (GNNs) to enhance prediction and reliability in inland waterway shipping scheduling problems. Graph neural networks have emerged as a powerful tool for modelling complex networks such as transportation systems. Their ability to capture and represent non-linear relationships within various parts of a system makes them well-suited to modelling the multifaceted dynamics of inland waterway transport mechanisms. Drawing inspiration from Joshi et al., 2019, we initially adapted GNN for the Travelling Salesman Problem (TSP) to align with the specificities of the inland waterway context. Inspired by Teichteil-Königsbuch et al., 2023, we subsequently employed GNN in an innovative manner to both the Job Shop Scheduling Problem (JSSP) and the Resource-Constrained Project Scheduling Problem (RCPSP), showcasing its potential to outperform traditional methods. In an innovative approach, we merged insights from the Waterway Ship Scheduling Problem (WSSP) as described in Verstichel, De Causmaecker, et al., 2014b with the principles of RCPSP from Hill et al., 2019, and tested this approach through a practical case study at the Port of Duisburg. The primary question in this thesis is: *How can Graph Neural Networks be effectively applied to address the challenges of prediction and optimisation in inland waterway shipping scheduling problems in comparison to traditional scheduling and prediction methodologies?* To address this research question, we investigate several other subquestions. These sub-questions are:

- How can GNN be tailored and adapted for the Travelling Salesman Problem in the context of inland waterway shipping, and what benefits does this adaptation offer compared to other TSP solvers?
- How can we develop a compact representation of the Job Shop Scheduling Problem using Graph Neural Networks, determine which GNN architecture is most effective for representing JSSP's graph and accurately capturing its disjunctive graph structure, and assess the GNN application in inland waterway context?

- How can we implement Graph Neural Networks for the Resource-Constrained Project Scheduling Problem, compare the performance of GNN solutions to traditional methods, and iteratively adjust the model to accommodate specific constraints relevant to inland waterway shipping?
- What are the challenges and potential benefits of translating the Waterway Ship Scheduling Problem intricacies into the RCPSP formulation by converting key WSSP variables into the RCPSP context, and how does the proposed RCPSP-based approach perform in a real-world setting?

To the best of our knowledge, using GNN in a supervised setting to solve the RCPSP problem in an inland waterway scheduling context is a novel approach. Therefore, the contribution of this thesis could be significant.

1.4 Approach

We propose an approach to address the challenges of prediction and reliability in inland waterway shipping utilising graph neural networks for various scheduling problems. We believe that GNN has shown great potential in modelling complex networks such as transportation systems and could be applied to the challenges faced by inland waterway shipping. Our first step is to explore GNN for TSP and adapt them to the inland waterway shipping context. This would provide us with a foundation for further exploration of the application of GNN to inland waterway shipping.

Next, in the realm of optimising the scheduling process, GNN offers a compelling avenue to tackle the intricate Job Shop Scheduling Problem. JSSP involves planning a series of jobs through various machines to minimise the completion time while respecting machine constraints. By harnessing the power of GNNs, we leverage their ability to capture complex interdependencies within scheduling tasks. This allows GNNs to learn and reason about the intricate relationships that govern scheduling decisions. Applying GNNs to JSSP introduces a novel approach that transcends traditional methods, potentially revolutionising the optimisation of inland waterway ship scheduling.

Next, we plan to use GNN for the Resource-Constrained Project Scheduling Problem. RCPSP is a problem in which the goal is to determine the optimal sequence of tasks, subject to precedence constraints and resource limitations, to minimise the project's completion time. This would involve creating a GNN model for Resource-Constrained Project Scheduling and introducing constraints specific to inland waterway shipping iteratively. When comparing GNN solutions with traditional solutions, we can determine the potential of using GNN for more complex scheduling problems in inland waterway shipping.

To solve the Waterway Ship Scheduling Problem, we explore a novel approach by drawing inspiration from the well-established field of the Resource-Constrained Project Scheduling Problem. Building on the foundational work of Verstichel, De Causmaecker, et al., 2014a, we translate the complexity of the WSSP into the RCPSP formulation. This involves converting various key variables from the WSSP domain into the RCPSP context. These variables include critical aspects such as scheduling time windows, ship measurements, and constraints related to the waterway capacity. By bridging the gap between these two distinct problem domains, we aim to leverage the insights and techniques developed in the RCPSP literature to tackle the unique challenges posed by the WSSP, ultimately seeking more effective solutions for efficient waterway scheduling and shipping operations. Furthermore, we conducted a case study on the port of Duisburg. This real-world investigation allowed us to apply and validate our proposed RCPSP-based approach in a practical setting, where the complexities of waterway scheduling and shipping operations come to life. By delving into the specifics of the port of Duisburg, we were able to assess the effectiveness and feasibility of our model.

In general, we believe that using GNN for scheduling and prediction could lead to more accurate and reliable solutions for inland waterway shipping operations. By leveraging the capabilities of GNN and adapting them to the specific context of inland waterway shipping, we can overcome the challenges of making accurate predictions and ensuring reliability in the operations of inland waterway shipping.

1.5 Outline

This thesis explores four optimisation problems: the Travelling Salesman Problem, the Job Shop Scheduling Problem, the Resource-Constrained Project Scheduling Problem, and the Waterway Ship Scheduling Problem. Section 2 focuses on TSP and begins with an introduction to the problem, followed by a formulation of the problem. A general literature review on TSP and a specific review on TSP with Graph Neural Networks are provided. Additionally, this section discusses TSP in the context of waterways. The data used, the methodology, and the results are then presented. Section 3 addresses JSSP with a similar structure. Section 4 follows a similar structure but focuses on RCPSP, including an introduction, problem formulation, literature review on RCPSP and RCPSP with GNN, data used, methodology, and results. Lastly, section 5 addresses the Waterway Ship Scheduling Problem with an introduction, problem formulation, literature review on the topic, translation of the WSSP to single mode RCPSP ship scheduling problem with GNN, data used, and results. The thesis concludes in Section 7 with a summary of the findings, implications, limitations, and suggestions for future research.

2 Travelling Salesman Problem

2.1 Introduction

The Travelling Salesman Problem (TSP) stands as a time-honoured combinatorial optimisation challenge that has consistently garnered the attention of mathematicians, IT experts, and scholars across various fields (Cook, 2012). Tracing its roots back to the nineteenth century, the TSP seeks the most concise route a salesperson should adopt to traverse several cities once and then return to the origin. Although its initial premise appears straightforward, the TSP's intrinsic complexities, combined with its broad spectrum of practical applications render it a formidable topic of exploration. It holds the distinction of being among the most dissected combinatorial optimisation issues (Cook, 2012). Numerous efforts have been directed towards uncovering effective solutions and heuristic strategies. A comprehensive overview of this problem can be found Dantzig et al., 1954, with the mathematical layout detailed in subsequent sections.

Applications The practicality of the Travelling Salesman Problem is evident in numerous sectors, highlighting its significance and applicability in research. It is instrumental in fields such as Supply Chain Management, Circuit Board Creation, data traffic management, and planning touristic ventures (Kumar, 2012). The TSP is pivotal in logistics for refining delivery pathways and pinpointing the optimal strategy to cover multiple destinations while reducing distance, duration, or expenditure. Within the realm of electronics, the TSP facilitates devising effective circuits for borehole creation in circuit boards, minimising equipment motion and maintaining operational efficacy. In the sphere of digital networks, the TSP optimises data packet routing, ensuring the shortest pathways between network vertices for streamlined data exchange. Furthermore, it helps to generate travel agendas for vacationers, proposing the most efficient itineraries to cover several points of interest within a predetermined time (Kumar, 2012).

The long-standing fascination of researchers with the Travelling Salesman problem is attributed to its mesmerising characteristics (Larranaga et al., 1999). In contrast to its simple formulation, identifying a solution remains challenging since it lacks a polynomial-time resolution algorithm. The challenge has extensive consequences in diverse routing and planning paradigms. The pre-existing insights regarding the TSP have prompted its recognition as a standard evaluation problem, serving as a preliminary step for developing novel combinatorial optimisation techniques.

Complexity Utilising a brute force tactic to assess every possible tour for a circuit encompassing n cities yields $(n-1)!$ outcomes with time complexity of $O(n!)$. However, taking advantage of the dynamic programming approach, a complexity emerges from a circuit of n cities, splintered into $n-2$ subsets, each spanning $n-1$ cities, all while excluding the n^{th} city. Hence, a maximum of $O(n2^n)$ analogous challenges can be linearly resolved, resulting in a time complexity of $O(n^2 2^n)$. The spatial and time complexities of the TSP can be typified as exponential (Davendra & Bialic-Davendra, 2020).

Algorithms and optimisation techniques The TSP is also often used as a benchmark for new heuristics and exact solution methods (Smith, 1996). Over the years, numerous algorithms and optimisation techniques have been developed to tackle the Travelling Salesman problem. These approaches range from exact methods, such as branch and bound, to approximation algorithms, such as nearest neighbour and genetic algorithms. Furthermore, researchers have explored heuristics, metaheuristics, and mathe-

matical programming techniques to find efficient solutions for different instances of the problem (Smith, 1996).

The TSP frequently benchmarks novel heuristics and exact solution methodologies (Smith, 1996). Over the decades, many algorithms and optimisation methods targeting the Travelling Salesman Problem have emerged. These range from exact methods like a branch and bound to approximation solutions such as nearest neighbour and evolutionary algorithms. Moreover, research has been done towards heuristics, metaheuristic procedures, and mathematical programmatic techniques to find efficient solutions for varying problem instances (Smith, 1996).

Variations Throughout its evolution, the TSP has diversified into a range of variations, each encompassing its distinct challenges (Davendra & Bialic-Davendra, 2020). The first and most common variation is the Symmetric TSP (STSP), where the distance between cities i and j is reciprocated when moving from city j to city i. In contrast, the Asymmetric TSP (ATSP) presents a scenario where the distances between city pairs are not necessarily symmetric. Another intriguing form is the Hamiltonian Cycle Problem (HCP), which examines whether a given graph G, whether directed or undirected, contains a route that touches each vertex V only once. The Sequential Ordering Problem (SOP) sets out with a predefined set of n cities, aiming to unearth the most direct Hamiltonian path from the first to the last city, all while respecting certain precedence constraints. Diving into logistics, the Capacitated Vehicle Routing Problem (CVRP) is centred around efficiently serving the needs of $n-1$ nodes, utilising a depot, and ensuring that vehicles of uniform capacity are optimally routed. Another notable version is the TSP within Neighbourhoods (TSPN), where the goal is to plot the shortest route encompassing a collection of L areas in the plane, labelled as neighbourhoods.

It is crucial to understand that the TSP has several forms. While the categories mentioned above are the primary classifications, other types also exist. This thesis will focus exclusively on the Symmetric Hamiltonian TSP.

2.2 TSP Problem formulation

The Travelling Salesman Problem is mathematically represented as detailed in Papadimitriou and Steiglitz, 1998. The problem involves a complete, undirected graph, denoted as $G = (V, E)$. Here, V signifies the set of cities, and E indicates the edges connecting these cities. Each edge $(i, j) \in E$ possesses a defined non-negative distance or cost, denoted as $c(i, j)$. This cost signifies the travel distance between the cities i and j. The aggregated cost for a specific tour is the sum of its individual edge costs:

$$\sum (c_{i_1 i_2} + c_{i_2 i_3} + \cdots + c_{i_{n-1} i_n} + c_{i_n i_1}), \tag{1}$$

where $i_1, i_2, \ldots, i_n$ is a permutation of $\{1, \ldots, n\}$. The TSP is to find a tour of minimum total cost.

The primary objective of the TSP is to discover a Hamiltonian cycle—a closed route visiting each city only once, ending where it began, with minimised cost. To depict the TSP precisely, a decision variable is used:

$$x(i, j) = \begin{cases} 1, & \text{if the edge } (i, j) \text{ is included in the tour,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The integer linear programming (ILP) formulation for the TSP includes the following:

$$\text{minimise:} \quad \sum_{(i,j) \in E} c(i, j) x(i, j) \tag{3}$$

$$\text{Subject to: } \sum_{(j,i)\in E} x(j,i) = 1 \qquad \text{for all } i \in V \tag{4}$$

$$\sum_{(i,j)\in E} x(i,j) = 1 \qquad \text{for all } j \in V \tag{5}$$

$$\sum_{(i,j)\in \delta(S)} x(i,j) \leq |S| - 1 \qquad \text{for all subsets } S \subseteq V,\ |S| > 1 \tag{6}$$

$$x(i,j) \in \{0,1\} \qquad \text{for all } (i,j) \in E \tag{7}$$

The constraints mentioned ensure that every city is visited and departed only once. Additionally, a subtour removal constraint is included to avoid suboptimal smaller tours and to ensure a valid Hamiltonian cycle.

Solving the TSP means determining the correct values for decision variables $x(i,j)$ that adhere to the ILP structure and minimise the primary objective function. Throughout the years, various optimisation techniques, heuristics, and approximation methods have been introduced to tackle the TSP's complexities. The overarching aim remains to find efficient solutions within a realistic computational timeframe, adaptable to the problem's diverse scopes and characteristics.

2.3 Literature

2.3.1 TSP Problems in general

Literature regarding TSP can be divided into several components. One primary segment emphasises exact and heuristic algorithms, striving for efficient solutions with commendable quality. A large data repository on the TSP problem, called *TSP Library*, conducted by Reinelt, 1991, serves as a beneficial resource for research and benchmarking. TSP literature reveals that initial solutions leaned heavily on linear programming and deterministic strategies (Davendra & Bialic-Davendra, 2020). Recent methods include the 2-Opt Algorithm (Hougardy et al., 2020), the Branch and Cut Algorithm (Yuan et al., 2020), varied algorithms both approximate and exact (S. Wang et al., 2020), and Branch and Bound (Salman et al., 2020). However, the challenging nature of the TSP has seen a subtle reduction in these mathematical models.

The second aspect of the literature refers to evolutionary methodologies (Davendra & Bialic-Davendra, 2020). Over the past few decades, with the advancement of computational power, a new branch of algorithms known as metaheuristics, based on evolutionary dynamics, has gained prominence in solving combinatorial optimisation problems. Inspired by naturally occurring phenomena, these meta-heuristics treat each problem as a black box and aim to find feasible, high-quality solutions within acceptable space and time constraints (Davendra & Bialic-Davendra, 2020). Numerous evolutionary algorithms have been developed and applied to the TSP problem. The important work on the Ant Colony optimisation algorithm by Dorigo and Gambardella, 1997 paved the way for further research in this area. Notably, the pioneering Ant Colony optimisation technique of Dorigo and Gambardella, 1997 initiated substantial subsequent investigations. Contemporary explorations in this realm incorporate Artificial Bee Colony (Pandiri & Singh, 2019), Differential Evolution (Ali et al., 2020), Genetic Algorithm (Dong & Cai, 2019), and other innovative approaches, among others.

The third component of the literature emphasises the role of high-performance computing, particularly its relevance to the TSP challenge (Davendra & Bialic-Davendra, 2020). With the widespread availability of parallel computing, especially with the rise of multi-core and GPU-based techniques, many algorithms

have been parallelised for enhanced performance. Some of the latest approaches in this segment include the Multi-Core method, OpenMP, and CUDA.

The ongoing research in the field of TSP demonstrates a shift from linear programming and deterministic methods towards metaheuristics based on evolutionary dynamics. These algorithms offer promising approaches to find optimal or near-optimal solutions to the TSP, making significant contributions to the field (Davendra & Bialic-Davendra, 2020). Looking ahead, even as combinatorial and scheduling challenges increase, TSP remains a focal research point. The consensus among researchers is that leveraging advanced heuristics on powerful parallel-computing machines may pave the way for addressing expansive TSP instances.

2.3.2 TSP Problems solved with GNN

This section delves into a comprehensive review of several important papers that have explored the intriguing intersection of TSP and GNNs.Prates et al., 2018 champions GNNs for the decision variant of TSP. Their model adeptly extrapolates to larger problem scales, boasting estimations of optimal route costs with a deviation of less than 2% from actual values. On the other hand, Joshi et al., 2019 promotes a unique GNN strategy for TSP on 2D Euclidean graphs. Their deep Graph Convolutional Networks outshine existing autoregressive deep learning solutions in aspects such as quality, speed, and efficiency for specific graph sizes. Diving into practical applications, Dobrilovic, 2022 formulates a TSP-based path planning model for UAVs in urban scenarios. By fusing the Genetic Algorithm (GA) and Dijkstra's algorithm, the study achieves an optimised UAV deployment that effectively follows city roads while maximising site coverage. A few works present hybrid strategies. Hudson et al., 2022 combines GNNs with Guided Local Search (GLS) to address the TSP, emphasising predictions on the regret associated with each edge. Results underscore this method's dominance over preceding learning-based approaches, highlighting a reduced optimality gap. Similarly, Dai et al., 2018 integrates reinforcement learning with graph embedding, offering a heuristic learning framework for multiple graph optimisation challenges, including TSP. Further enriching the GNN-TSP realm, Hu et al., 2021 introduces a bidirectional GNN (BGNN) for TSP on generic symmetric graphs. Using imitation learning, this network sequentially selects cities to visit, with its unique bidirectional layer enhancing its interpretative capacity. The model exhibits superior solution quality by converging BGNN with informed search strategies. Lastly, Shi and Zhang, 2022 provides an overview of neural network-driven solutions for TSP, encompassing Hopfield neural networks, GNNs, and reinforcement learning-backed neural networks. This research trajectory affirms the revolutionary role of neural networks in refining TSP approximations within combinatorial optimisation. In the context of inland waterways, our primary emphasis is on the supervised setting and 2D Euclidean graphs. This focus is crucial as inland waterway systems are inherently based on 2D geographical coordinates for effective optimisation and management. To continue our efforts, we intend to build upon the work of Joshi et al., 2019. Their research, which also harnesses coordinates in a 2D supervised framework, offers a foundational base and aligns closely with our objectives, making it an ideal starting point for our exploration.

Table 1: Literature overview of GNN used for TSP in general

Article	Model
Prates et al., 2018	Graph Neural Networks solving the decision variant of TSP
Joshi et al., 2019	Deep Graph Convolutional Networks solving TSP on 2D graphs
Dobrilovic, 2022	Solving TSP with Genetic Algorithm + Dijkstra algorithm
Hudson et al., 2022	Hybrid approach: GNN + Guided Local Search (GLS)
Dai et al., 2018	Combination of RL and graph embedding to automate learning heuristic algorithms
Hu et al., 2021	Bidirectional Graph Neural Network (BGNN) for solving TSP on arbitrary symmetric graphs.
Shi and Zhang, 2022	Hopfield neural networks, GNN and RL for TSP

2.3.3 TSP problems in Inland Waterway Transport context

This section delves into a comprehensive review of several important papers that have explored the intriguing intersection of TSP in the waterway context.

Miranda et al., 2018 broaches a bi-objective TSP variant tailored for isolated regions where a lone barge reaches rural islands. The study uses a mixed integer programming model by balancing sea and terrestrial transport costs and factoring in variable freight at each dock. This is illuminated through instances from Chile, demonstrating its relevance for goods distribution in remote areas.

Shifting to maritime logistics, Malaguti et al., 2018 introduces a TSP generalisation integrating pickup and delivery components. The ports, symbolised as nodes, come with distinct draught limits. These constraints interplay with specific customer demands, which a single ship navigates. The proposition is a meticulous integer linear programming model enhanced by heuristic techniques and an exact branch-and-cut algorithm, the efficacy of which is empirically proven.

Delving into the practical nuances of maritime transportation, Arnesen et al., 2017 details an in-port ship routing issue prevalent in chemical shipping. This challenge is formulated as the TSPPD-TWDL, which amalgamates aspects of pickups, deliveries, time constraints, and draught limits. By leveraging forward dynamic programming, the method proves its mettle through real-world scenarios.

In another innovative approach, Bauk, 2004 remodels the Hopfield-Tank neural network methodology for TSP, mainly tailoring it for delineating optimal ship routes. Grounded in a mathematical structure, the study accentuates its capability through a case from the Adriatic Sea.

Further nuances in maritime TSP are delved into Ghasemi-Sardabrud et al., 2019, spotlighting a variant encumbered with pickup, delivery, time constraints, and draught limits. Through a comparative analysis between integer linear programming and forward dynamic programming, the research highlights the intricacies of the challenge and avenues for refinement.

Fagerholt and Christiansen, 2000 elaborates on a unique TSP iteration entwined with bulk ship scheduling, encompassing allocation, time mandates, and sequence constraints. Cast as a graph-centric shortest path dilemma, the study harnesses forward dynamic programming, punctuated by tests, revealing its real-world relevance and signposting further research directions.

Lastly, Mahmoodjanloo et al., 2021 focuses on multi-ship routing within ports. The crux is to orchestrate optimal paths for multiple vessels, each navigating unique terminal draught restrictions. This two-phase strategy, merging dynamic programming and branch-and-bound principles, demonstrates its

prowess through a practical example and sheds light on computational complexities and potential enhancements.

Table 2 provides an overview of the literature concerning TSP in the waterway context. Notably, the prevailing methodologies in this literature have predominantly revolved around Mixed-Integer Linear Programming or Dynamic Programming techniques. However, employing Graph Neural Networks for TSP within the inland waterway context represents, to the best of our knowledge, a novel and essential approach that has not been extensively explored. This perspective could offer unique solutions and insights, differentiating it from traditional methodologies.

Table 2: Literature overview of TSP used in waterways

Article	Model
Miranda et al., 2018	MIP model to solve a bi-objective variant of the TSP where rural islands need to be serviced
Malaguti et al., 2018	LP formulation and heuristic procedures for TSP with pickup and delivery
Arnesen et al., 2017	Dynamic programming for TSP with Pickups and Deliveries, Time Windows, and Draft Limits (TSPPD-TWDL)
Bauk, 2004	Hopfield-Tank neural network algorithm for TSP, specifically applied to determining the optimal linear ship route
Ghasemi-Sardabrud et al., 2019	ILP for determining the optimal routes and schedules for a pickup fleet of ships.
Fagerholt and Christiansen, 2000	Shortest path problem on a graph and a forward dynamic programming algorithm to solve TSP optimal routing and scheduling.
Mahmoodjanloo et al., 2021	A two-stage solution method based on DP and branch-and-bound algorithms for multi-ship routing and scheduling problem

2.4 Data used

In our research of the TSP, we primarily rely on two data sets: benchmark and waterway systems. The benchmark dataset delivers outcomes from heuristic approaches and the Concorde solver, showcasing time frames and disparities in optimal solutions. Conversely, the waterway dataset chronicles ship movements among ports, considering each port as a TSP "city". The distances among these "cities" are derived from the waterway infrastructure, factoring in geographical points, straight-line calculations, or other criteria. It is worth highlighting that we use the API to collate coordinates and crucial data. Our model is refined and analysed on the benchmark dataset, with a comprehensive evaluation using the API merged with port positions.

The study primarily examines the 2D Euclidean TSP, where a graph has n cities confined to a two-dimensional unit square $S = \{x_i\}_{i=1}^{n}$. x represents the coordinates of cities in a two-dimensional unit square S. The objective is to determine a tour permutation $\hat{\pi}$ that visits each city once, minimising the overall distance, as represented by the equation, where the notation $|\cdot|_2$ signifies the ℓ_2 norm:

$$L(\hat{\pi}|s) = \|\mathbf{x}_{\hat{\pi}(n)} - \mathbf{x}_{\hat{\pi}(1)}\|_2 + \sum_{i=1}^{n-1} \|\mathbf{x}_{\hat{\pi}(i)} - \mathbf{x}_{\hat{\pi}(i+1)}\|_2 \tag{8}$$

Inspired by Vinyals et al., 2015, deep learning-based TSP solutions focus on fixed-size instances. The training set has 100,000 pairs, and both validation and test sets have 10,000 pairs. The node positions are

randomised in the unit square, and the Concorde solver (Applegate et al., 2006) determines the optimal tours within a 10-minute limit.

TSPLIB data set Also, using the TSPLIB data set (Reinelt, 1991), this thesis explores the Symmetric Travelling Salesman Problem. The TSPLIB is a well-established benchmark library in combinatorial optimisation, containing a diverse collection of TSP instances that have been extensively studied and used for benchmarking various optimisation algorithms.

We have selected nine representative instances from the TSPLIB: *berlin52, bays29, gr96, gr120, lin105, rd100, gr666, ulysses16*, and *eil101*. These instances have been chosen to demonstrate and compare the performance of different solution approaches, allowing rigorous evaluation and benchmarking of algorithms in the context of real-world TSP instances with varying characteristics and complexities. Within the TSPLIB dataset, *berlin52, bays29, and gr120* emerge as noteworthy instances in the TSP. These instances encapsulate various challenges: *berlin52* represents a classic TSP with 52 cities, bays29 offers insights into the intricacies of navigating 29 cities in the San Francisco Bay area, and *gr120* presents a formidable challenge with 120 cities, often serving as a testbed for assessing algorithm scalability. Each of these instances presents unique characteristics, including the city distribution and geometric structure, making them crucial for benchmarking and refining TSP-solving algorithms. Researchers and practitioners frequently turn to these instances to assess the efficiency and adaptability of optimisation techniques, ensuring their applicability to various real-world routing and logistics scenarios. By exploring *berlin52, bays29, and gr120*, we deepen our understanding of combinatorial optimisation's practical implications and capabilities.

Instances *gr96* and *gr120* exhibit intricate geometric structures and larger scales, which pose considerable computational challenges. On the other hand, instance *lin105* represents instances with more linear and clustered city layouts. While smaller in scale, the instance *bays29* introduces irregularities commonly encountered in real-world TSP scenarios.

By grounding our analysis in these carefully chosen TSPLIB instances, this research aims to provide a comprehensive benchmark for evaluating and comparing various TSP solution approaches and algorithms. Our investigation seeks to shed light on the adaptability and robustness of optimisation techniques in addressing real-world TSP complexities efficiently, thus advancing the understanding of practical problem-solving within the realm of combinatorial optimisation.

Data used in Inland Waterway Transport context To diversify the application of the Travelling Salesman Problem, we sought to adapt it to the unique challenges presented by the inland waterway networks, specifically focusing on the Rotterdam region. The importance of such waterways, particularly in this region, cannot be underestimated (Waterstaat, 2021). Not only do they serve as crucial nodes in the logistical networks of the country, but they also have significant economic value, facilitating trade and movement of goods (Waterstaat, 2021). The port of Duisburg, Europe's top inland port, faces challenges similar to Rotterdam's port because of its crucial role in logistics. Both navigate complex inland waterways, highlighting the need for better vessel coordination. Adapting the TSP for Rotterdam offers insights for ports like Duisburg, potentially refining routes, reducing downtime, and improving efficiency. Essentially, our Rotterdam study could guide operations in key inland ports such as Duisburg.

To obtain accurate and relevant data for our research, we leveraged the Teqplay API, a well-regarded source for port data. Through this API, we were able to extract comprehensive information regarding five key ports: "Terneuzen", "Vlissingen", "Rotterdam", "Hansweert", and "Zeeland overig". Rather than using point coordinates, the extracted data represented each port as a polygon, a shape that considers

the vastness and irregularities of the real-world port boundaries. This can be seen in Figures 1 and 2.

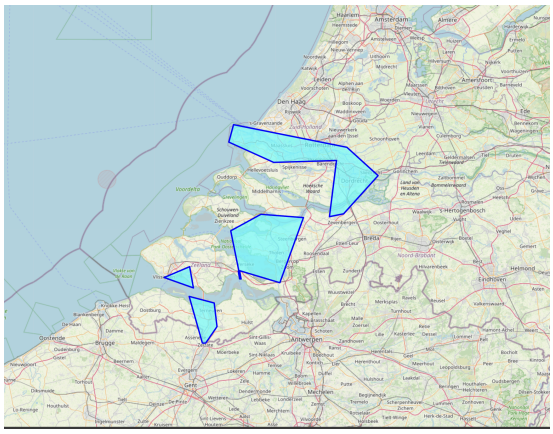

Figure 1: Utilising Teqplay API for in-depth port data study: Mapping the polygonal boundaries of ports on actual map.

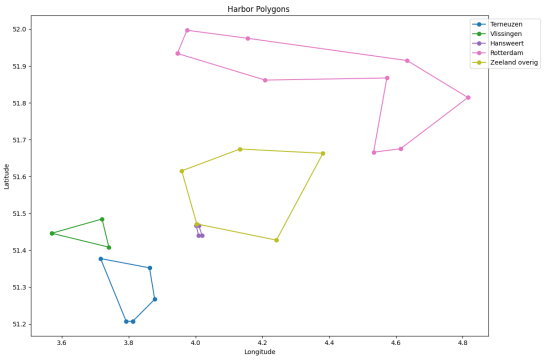

Figure 2: Utilising Teqplay API for in-depth port data study: portraying the polygonal borders of ports on a schematic diagram.

To effectively integrate the polygonal data into our TSP model, we employed a methodological step: calculating the centroids of each polygon. The centroid of a polygon, essentially its geometric centre, served as a representative point or node for the port in the TSP. This conversion from polygons to nodes is a critical step, ensuring that our TSP model remains tractable while retaining a semblance of realism.

However, it is important to highlight that this dataset might seem rudimentary with only five nodes compared to more complex TSP scenarios. As such, we opted to utilise this dataset as a supplementary application to showcase the model's potential in this context rather than as primary test data. Although limited in scale, this demonstration provides a glimpse into the potential for applying optimisation techniques to real-world logistical challenges in the inland waterway sector.

2.5 Methodology

Before delving into the specific GNN model and beam search decoding, let's first provide an overview of neural networks.

Neural Networks are inspired by human brain neuron structures (Krose, 1996). Early models of artificial neurons produced binary outputs based on input thresholds (McCulloch & Pitts, 1943). This was expanded by Rosenblatt, 1957, leading to the creation of the perceptron, which processes weighted inputs, sums them, adds a bias, and then applies an activation function for non-linearity.

The Multilayer Perceptron (MLP) is an advanced neural network with three main layers: input, hidden, and output. It operates in two phases: forward propagation, where inputs are processed to produce an output, and backpropagation, which adjusts weights based on the difference between expected and actual outputs using methods like Gradient Descent.

On the other hand, the Convolutional Neural Network (CNN) is a neural network variant designed for tasks like image classification. CNNs use convolutional layers to reduce parameters and employ filters to extract key features from inputs, such as images. They can produce various outputs like edge detection or blurring and excel in tasks like identifying handwritten digits in datasets like MNIST.

Our research employs a non-autoregressive deep learning method to approximate the TSP using the Graph Convolutional Network, as highlighted in Bresson and Laurent, 2017. Our approach, visualised in Figure 3, is inspired by the methodology in Joshi et al., 2019. Upon receiving a graph input, the

graph GNN model is trained to produce an adjacency matrix indicating the TSP tour likelihoods. The resultant edge projections, forming a *heatmap*, are then transformed into a valid tour using beam search, proposed by (Medress et al., 1977). Model training occurs in a supervised setting, leveraging optimal solutions from the Concorde TSP solver (Applegate et al., 2006), and the model parameters are optimised end-to-end by minimising cross-entropy loss using gradient descent.

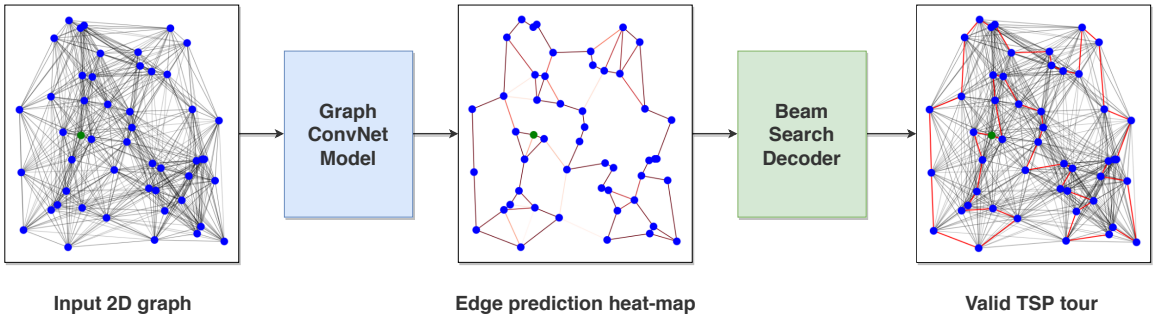

Figure 3: Visualising of our method: we process a 2D graph through a graph GNN model, yielding an edge probability matrix for potential TSP tour paths. Beam search refines this to a valid tour (Joshi et al., 2019)

2.5.1 Graph Convolutional Network Model

For the initial node feature, two-dimensional coordinates $x_i \in [0, 1]$ are presented, which are then transformed to a feature set of dimension h:

$$\alpha_i = A_1 x_i + b_1 \tag{9}$$

with $A_1 \in \mathbb{R}^{h \times 2}$. The Euclidean distance between edges, d_{ij}, is represented by a vector of dimension $\frac{h}{2}$.

With node and edge features at layer ℓ being x_i^ℓ and e_{ij}^ℓ respectively, the subsequent layer's features are defined as:

$$x_i^{\ell+1} = x_i^\ell + \mathrm{ReLU}\Big(\mathrm{BN}\Big(W_1^\ell x_i^\ell + \sum_{j\sim i} \eta_{ij}^\ell \odot W_2^\ell x_j^\ell\Big)\Big) \text{ with } \eta_{ij}^\ell = \frac{\sigma(e_{ij}^\ell)}{\sum_{j'\sim i}\sigma(e_{ij'}^\ell) + \varepsilon}, \tag{10}$$

$$e_{ij}^{\ell+1} = e_{ij}^\ell + \mathrm{ReLU}\Big(\mathrm{BN}\Big(W_3^\ell e_{ij}^\ell + W_4^\ell x_i^\ell + W_5^\ell x_j^\ell\Big)\Big), \tag{11}$$

The given equations describe an iteration step of a GNN with both node and edge updates. The term x_i^ℓ represents the feature vector of node i at layer ℓ. This vector captures the information or attributes of the node i at a specific GNN layer. Meanwhile, e_{ij}^ℓ denotes the feature vector of the edge connecting nodes i and j at layer ℓ, encapsulating the relationship attributes between the two nodes. Weight matrices associated with the ℓ^{th} layer of the GNN are denoted as $W_1^\ell, W_2^\ell, W_3^\ell, W_4^\ell$, and W_5^ℓ. These matrices are the learned parameters that the GNN utilises to update the features of nodes and edges. The notation $\sum_{j\sim i}$ is a summation over all nodes j that are directly connected or neighbours to node i. Moreover, η_{ij}^ℓ is a critical component that signifies a normalised attention weight or coefficient between node i and node j at layer ℓ. This normalisation is executed over all the neighbours j' of the node i, ensuring the sum of attention weights for node i converges to approximately one. The symbol $\odot$ signifies element-wise multiplication.

The activation function ReLU, mathematically expressed as $f(x) = \max(0, x)$, introduces non-linearities, while BN denotes Batch Normalization, a technique optimising deep neural network training by normalising layer inputs. The function σ denotes the sigmoid activation, and ε is a small constant added to denominators for stability against division by zero.

In essence, Equation 10 elucidates the update mechanism for node features. The process integrates the node's existing features with the aggregated features from its neighbouring nodes, adjusted by the weights or attention coefficients of the edges. Concurrently, Equation 11 delineates the updating process for edge features, considering the attributes of the connecting nodes and the edge's prior features. Collectively, these equations enable information propagation across the graph, facilitating the GNN's learning of intricate patterns embedded within the graph data. At the input layer, the model initializes $x_i^{\ell=0} = \alpha_i$ and $e_{ij}^{\ell=0} = \beta_{ij}$. Figure 4 illustrates the model.

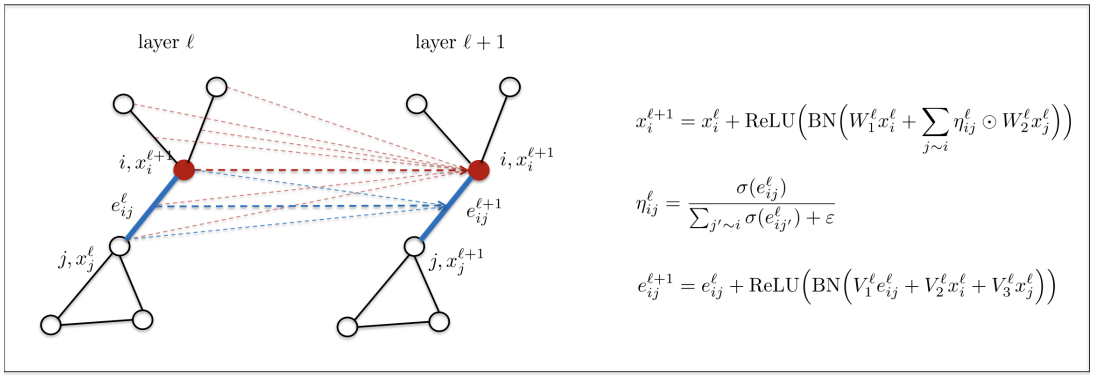

Figure 4: The illustrated graph convolutional layer determines the h-dimensional features, x_i, for each node i and e_{ij} for edges linking nodes i and j. Red arrows depict the flow of node information, while blue arrows showcase the edge information influencing the subsequent layer's representations. Through successive graph convolution layers, the system incrementally derives intricate features from the input graph (Joshi et al., 2019).

The edge representation from the final layer, e_{ij}^{ℓ}, assists in determining the likelihood of a particular edge forming part of the TSP graph's tour. This likelihood essentially forms a probabilistic heatmap on the adjacency matrix, represented as *Multilayer Perceptron* (MLP):

$$p_{ij}^{\mathrm{TSP}} = \mathrm{MLP}(e_{ij}^{\ell}) \tag{12}$$

The ground truth TSP tour permutation π can be converted into an adjacency matrix, with each element $\hat{p}_{ij}^{\mathrm{TSP}}$ indicating the existence of a connecting edge. A weighted binary cross-entropy loss is minimised to refine the model.

2.5.2 Beam Search Decoding

Our model produces a probabilistic adjacency matrix representation that signifies tour linkages. The measure $p^{\mathrm{TSP}}ij \in [0,1]^2$ represents the strength of the prediction of a connection between the nodes i and j. The likelihood of a partial TSP route π' can be expressed as:

$$p(\pi') = \prod_{j'\sim i'\in\pi'} p_{i'j'}^{\mathrm{TSP}} \tag{13}$$

Here, every node j' succeeds node i' in the partial route π'. It is worth noting that transforming this probabilistic heatmap directly into an adjacency matrix representation of the predicted TSP tour might lead to incorrect routes either having surplus or inadequate edges in $\hat{\pi}$. Consequently, we adopt two distinct methodologies during the evaluation to transform the probabilistic edge representation into a valid node sequence $\hat{\pi}$.

Greedy search Typically, greedy algorithms target local optimal outcomes to quickly approximate the global optimal solution. Initiating with the primary node, the following node is greedily chosen from its adjacent nodes based on the maximum edge probability. This process concludes once every node has been covered. To ensure valid routes, nodes that have already been traversed are omitted.

Beam search Starting with the first node, we delve into the representation, unfolding the top b probable edge linkages among the neighbouring nodes. b is denoted as the beam width. We consistently unfold the highest probability b intermediary tours at every phase until every node in the network is covered. An identical exclusion principle used in greedy search is applied here to derive correct routes. The route with the most significant likelihood among the b finalised routes is chosen as the final prediction.

2.5.3 Hyper parameter Configurations

For all problem dimensions, we maintain consistent model hyperparameters. Our model incorporates $l_{conv} = 30$ graph convolutional layers and $l_{mlp} = 3$ MLP layers, each having a hidden size of $h = 300$. A set beam width of $b = 1,280$ is chosen, facilitating a direct comparison with the other results in Kool et al., 2019. We examine the nearest 20 neighbours for every node in the adjacency matrix W^{adj}.

2.6 Experiments

In our experimental approach, we consistently train the GNN model across various dimensions to generate an adjacency matrix corresponding to a TSP tour, leveraging gradient descent to minimise cross-entropy loss. Each training cycle involves processing a randomly chosen 10,000 sample subset from the 100,000 training instances, further subdivided into 500 mini-batches of 20 instances each, with the Adam optimiser set at a learning rate of 0.001. We assess our model's performance against a 10,000-sample validation set every fifth cycle. For evaluation, the model output is converted into a valid tour as detailed in Section 2.5.2. The metrics aligned with Joshi et al., 2019 include the average TSP tour length over the test samples and the optimality gap, representing the percentage variance between our predicted tour lengths and the optimal solutions. The unit used in this study for the 2D Euclidean TSP is the "unit square." The cities are limited to a unit square of two dimensions $S = x_i{}_{i=1}^{n}$:

Predicted Tour Length: The average TSP tour length over 10,000 test samples, represented as $\frac{1}{m}\sum_{i=1}^{m} l_m^{\text{TSP}}$.

Optimality Gap: This quantifies the mean percentage difference between our predicted tour length and the optimal solution, calculated as $\frac{1}{m}\sum_{i=1}^{m}\left(l_m^{\text{TSP}}/\hat{l}_m^{\text{TSP}} - 1\right)$.

2.7 Results

The code was implemented in Python within a Visual Studio Code environment. It incorporated several libraries, including Networkx, Pytorch, Scikit-learn and Tensorflow, utilising an 8-core Apple M1 CPU @3.20Ghz, 16GB RAM and an 8-core GPU. Table 3 presents our method's performance against both benchmarks with varied TSP sizes. In the table, methods are categorised into exact algorithms, heuristic strategies (G), and sampling/search techniques (S) and further sorted by training type: supervised learning (SL), reinforcement learning (RL), and traditional heuristics (H). All results, barring ours (highlighted in bold), originate from Kool et al., 2019, which also offers detailed analysis of the Concorde algorithm and standard benchmarks like Random Insertion and Nearest Neighbour.

Table 3: Results of our method's performance against non-learned baselines for different TSP sizes. In the table, methods are categorised into exact algorithms, heuristic strategies (G), and sampling/search techniques (S) and further sorted by training type: supervised learning (SL) and traditional heuristics (H).

Method	Type	TSP10			TSP20			TSP50		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
Concorde	Solver	2.87	0.00%	(20 sec)	3.84	0.00%	(1m)	5.70	0.00%	(2m)
Random Insertion	H, G				4.00	4.36%	(0s)	6.13	7.65%	(1s)
Nearest Neighbour	H, G				4.50	17.23%	(0s)	7.00	22.94%	(0s)
GCN	**SL, G**	**2.93**	2.09%	**(2s)**	**3.86**	**0.60**%	**(6s)**	**5.87**	**3.10**%	**(55s)**
OR Tools	H, S				3.85	0.37%		5.80	1.83%	
GCN	**SL, BS**	**2.90**	1.04%	**(10s)**	**3.84**	**0.10**%	**(20s)**	**5.71**	**0.26**%	**(2m)**

From Table 3, we can observe a comparative analysis of our method's (GCN) performance against various non-learned baselines across different TSP instance sizes: TSP10, TSP20, and TSP50. When evaluated using the Concorde solver, an optimal solution is achieved with a tour length of 2.87, 3.84, and 5.70 for TSP10, TSP20, and TSP50, respectively, with no optimality gap and computation times ranging from 20 seconds to 2 minutes.

Traditional heuristic methods, like Random Insertion, show an optimality gap of up to 7.65% on TSP50, and Nearest Neighbour exhibits a more pronounced gap of 22.94% for the same TSP size. Both methods compute almost instantaneously, reflecting the trade-off between computational time and solution accuracy.

In contrast, when trained via supervised learning (SL) and combined with greedy heuristics, our GCN method demonstrates promising results. The tour lengths for TSP10, TSP20, and TSP50 are 2.93, 3.86, and 5.87, respectively. The optimality gaps remain low, with 2.09% for TSP10 and as minimal as 3.10% for TSP50. This suggests that the instance's size may be too simplistic for a meaningful comparison between models. The computational times are also considerably efficient, ranging from 2 seconds to just under a minute.

When our GCN model employs a different strategy (SL, BS), it further enhances the performance, reducing the optimality gap to a mere 0.26% for TSP50 and outperforming OR tools in both solution accuracy and evaluation time. The computation times are slightly higher but remain efficient, especially considering enhanced accuracy. When evaluating time, comparing our non-autoregressive model to autoregressive ones may not be fair. Autoregressive models build the TSP tour step-by-step, while ours predicts connections individually, adding extra time due to a two-stage process.

In summary, our GCN method consistently outperforms traditional heuristics, achieving near-optimal solutions with efficient computation times across all TSP instance sizes.

Figure 5 offers visual representations of the Travelling Salesman Problem, for example, instances with 10, 20, and 50 cities, respectively. Within each figure, three distinct presentations can be observed: the leftmost section provides a display of the ground truth; the central section showcases a heat map representation, offering insights into the concentration and distribution of the cities; and on the right, we see the predicted TSP tour, which is generated using beam search. From a cursory examination, it becomes evident that the GNN model, combined with the beam search, predicts with remarkable accuracy. This high level of precision in the predictions underscores the efficacy of the GNN model in handling such computational challenges, confirming its robustness and adaptability to problems of this nature.

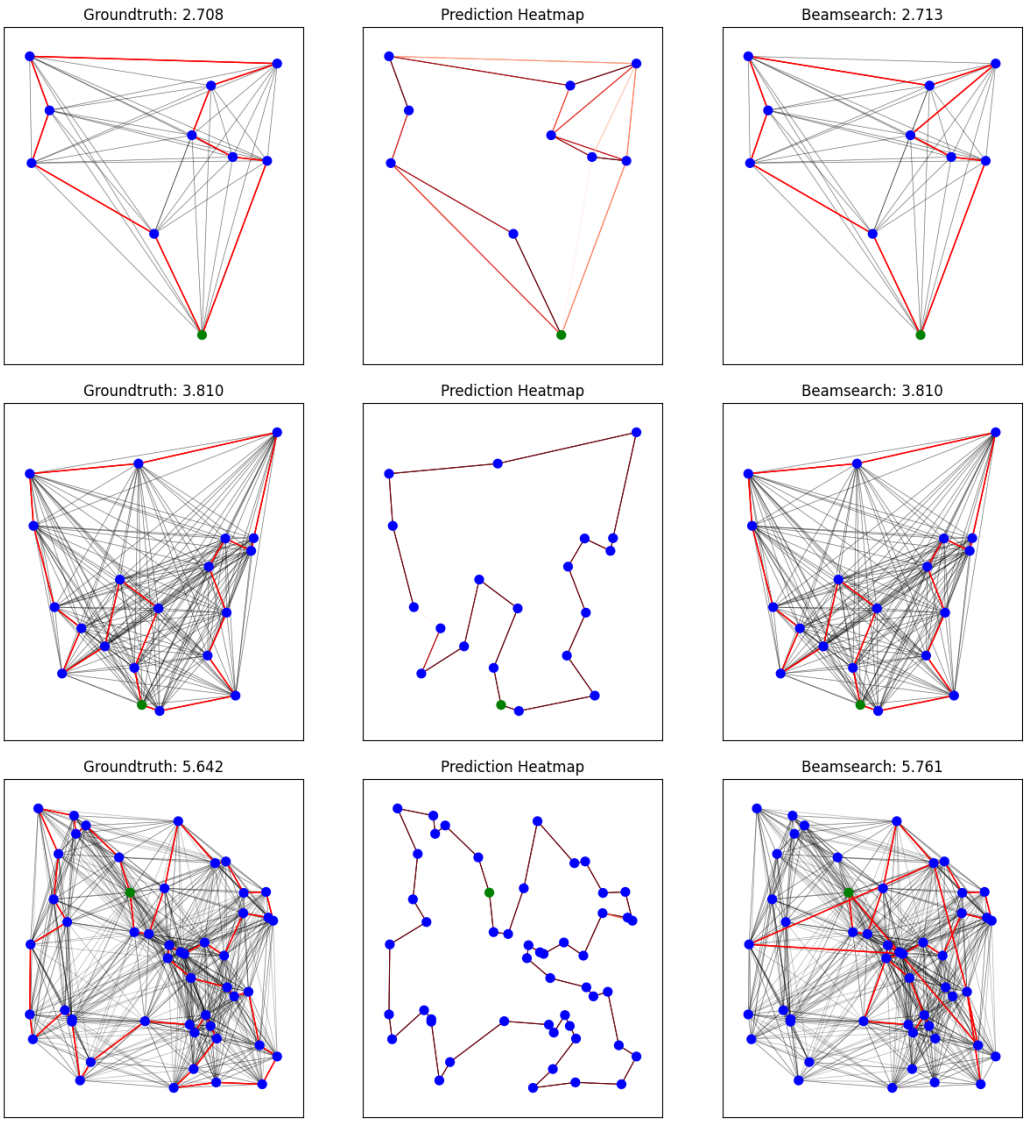

Figure 5: Visualisations of TSP, for example, instances with 10, 20, and 50 cities, respectively. In each figure: (left) displays the ground truth; (middle) shows the heat map representation; and (right) presents the predicted TSP tour using beam search.

TSPLIB comparison results The comparative analysis of the nine selected TSPLIB instances involving the original Concorde output (ground truth, predicted heatmap tour length, and encoded solutions generated by beam search) yielded valuable insights.

Table 4 presents a comprehensive comparison of solution outputs for a range of Travelling Salesman Problem instances. These instances, namely berlin52, bays29, gr96, gr120, lin105, rd100, gr666, ulysses16, and eil101, represent various challenges in TSP optimisation, varying in the number of nodes and the distribution of cities.

The groundtruth column provides the known optimal tour lengths for each instance, serving as a benchmark for evaluating the performance of optimisation techniques. In particular, instances like berlin52 and bays29 exhibit relatively short optimal tours, while gr96 and gr120 pose significantly more complex routing challenges due to their larger node counts.

The Predicted column showcases the tour lengths produced by the GNN model, reflecting its ability to approximate optimal solutions. Although the predictions generally align with the ground-truth values, variations indicate the inherent difficulty of TSP.

The beam search column presents tour lengths generated using a heuristic-based approach after the GNN model, demonstrating the efficiency of the beam search algorithm in quickly approximating solutions. While the beam search solutions are near optimal for some instances, the algorithm faces increased challenges with larger instances like gr666.

The Optimality Gap column quantifies the relative performance of the predictive model and beam search compared to ground truth, highlighting the percentage difference from the optimal solution. Notably, the gaps indicate that, while these approaches offer viable solutions, there is room for improvement, especially for instances with high opt gaps, such as gr96 and gr120.

In summary, the table's results emphasise the complexities and nuances of the Travelling Salesman Problem across various instances. The analysis highlights the trade-offs between solution quality and computational efficiency and illuminates the potential for further optimisation and refinement to tackle these real-world routing and logistics challenges. Notably, the beam search-encoded solutions demonstrate a commendable level of similarity to the original Concorde outputs, indicating the potential of beam search as an effective heuristic for tackling the Symmetric Travelling Salesman Problem. However, it is worth noting that, due to the model's training constraints, which primarily encompass instances with fewer than 50 nodes, the beam search-encoded results exhibit slight deviations in larger instances such as gr96 and gr120. Although the beam search approach appears promising, these results underscore the need for further model training on larger instances to enhance its accuracy and applicability to complex real-world TSP scenarios.

Table 4: Comparative analysis of solution outputs across nine selected TSPLIB instances for the Travelling Salesman Problem. The table showcases the known optimal tour lengths (Groundtruth), the predicted tour lengths from the GNN model (Predicted), the solutions generated using the beam search algorithm (Beamsearch), and the percentage difference from the optimal solution (Opt Gap). The analysis highlights the challenges of TSP optimisation, the effectiveness of the models and algorithms used, and areas for potential improvement.

Instance	berlin52	bays29	gr96	gr120	lin105	rd100	gr666	ulysses16	eil101
Groundtruth	7540,2	8975,9	502,4	11401,1	14379,4	790,8	3919,3	73,8	641,2
Predicted	7922,7	11553,7	852,6	12371,2	16668,0	1137,9	5731,4	84,0	1016,7
Beamsearch	7632,8	9021,1	792,0	12269,1	14917,9	870,9	4517,2	77,1	738,7
Opt Gap	1.23%	0.50%	57.64%	7.61%	3.74%	10.13%	15.25%	4.45%	15.20%

Results in Inland Waterway Transport context As previously mentioned in the data section, our data set comprises polygons representing five distinct harbours, as shown in Figure 6 and a larger representation is provided in 2. To enhance the practicality of our modelling, we transform these polygons into their respective centroids, as illustrated in Figure 7. The geographical layout facilitates seamless navigation between any pair of ports, resulting in each port having four neighbouring points, as shown in Figure 8. The culmination of our analysis is visually presented in Figure 9.

The results of our analysis reveal a notable alignment between the predictions generated by the TSP model and the ground truth. This congruence can be attributed to the relatively limited scale of the dataset, which encompasses only five nodes. Although this inherent simplicity influences the model's accurate predictions, it is essential to emphasise that this demonstration does not detract from the practical value of its application. Even within this constrained scenario, the TSP model effectively showcases its potential, offering valuable insights into optimisation strategies for real-world logistical

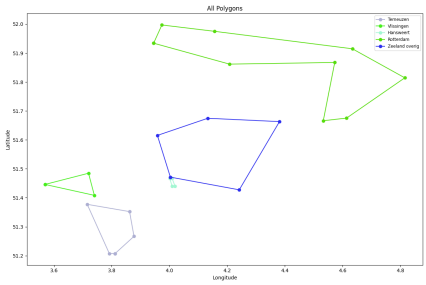

Figure 6: Utilising Teqplay API for port data: mapping the polygonal boundaries of ports on schematic map

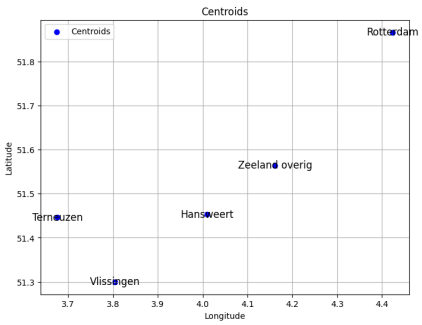

Figure 7: Converting polygons to centroids and simplifying navigation with four neighbouring points

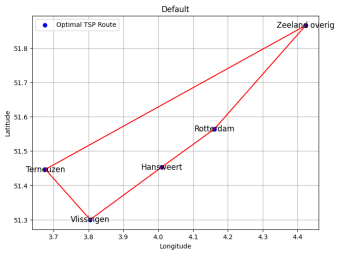

Figure 8: Displaying possible route: geographical layout showcasing all viable edges between ports with four neighbouring points

complexities. This underscores the versatility and adaptability of the model, which can readily extend its utility to more complex and comprehensive settings.

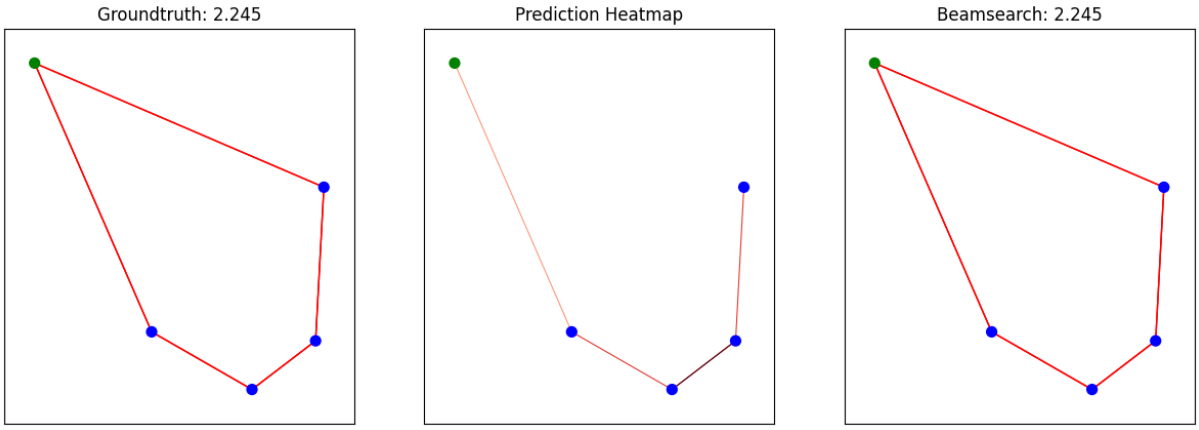

Figure 9: Visualising optimal routing for inland waterway transport extracted for port with Teqplay API. From left to right: Ground truth of the TSP tour, heat map representation, and predicted TSP tour using beam search.

2.8 Conclusion

In the pursuit of solving the Travelling Salesman Problem (TSP) for 2D Euclidean graphs, this thesis explores the potential of Graph Convolutional Networks (GCN). Through a unique blend of deep learning and beam search, the proposed model provides solutions in a non-autoregressive manner, achieving superior results compared to recent autoregressive techniques in solution quality, speed, and sample efficiency.

A comparative evaluation of the GCN's performance against other non-learned baselines reveals its prowess across various TSP instance sizes. For instance, when compared with the Concorde solver, which yields optimal solutions, the GCN model, enhanced with supervised learning and a greedy heuristic, exhibits near-optimal solutions. In particular, the model maintains efficient computational times and achieves minimal optimality gaps, emphasising its accuracy and reliability. When further integrated with a different strategy, with supervised learning and a beam search heuristic, the GCN model's efficiency improves, outclassing other tools in solution accuracy and processing speed. Interestingly, the non-

autoregressive nature of the model, which predicts connections individually, adds an extra computation layer, showcasing a trade-off between sequential and parallel predictions.

Visual representations of the TSP instances reiterate the model's proficiency. Across various cities' instances, the model, paired with beam search, can make accurate predictions, underlining its resilience and adaptability. Further insights are derived from the comparison against nine TSPLIB instances. While the GNN model exhibits promising approximations of optimal solutions, challenges persist, especially for larger instances, highlighting the inherent complexity of TSP. The results illuminate the potential and areas of refinement, emphasising the need for future research and optimisation to address more intricate real-world routing challenges.

The analysis shows that beam search, as a heuristic, holds promise in addressing the symmetric Travelling Salesman Problem. However, it also hints at the need for extended training in larger instances to increase precision, especially in more complex TSP scenarios.

In adapting the Travelling Salesman Problem to the inland waterway networks of the Rotterdam region, the study successfully integrated the complex geographical data of key ports, represented as polygons, by converting them into tractable centroids. While the chosen dataset, consisting of five nodes, may seem simplistic, the congruence between the TSP model's predictions and the ground truth demonstrates the model's potential in this context. The results underscore the versatility of the TSP model, highlighting its applicability in optimising real-world logistical challenges within inland waterway networks, even in more intricate scenarios beyond this study. This research offers a promising avenue to improve logistic efficiency in the vital inland waterway sector.

To address the research subquestion: "How can GNN be tailored and adapted for the Travelling Salesman Problem in the context of inland waterway shipping, and what benefits does this adaptation offer compared to other TSP solvers?", the thesis demonstrates that GNN, when tailored with specific strategies such as supervised learning and beam search, can be adapted efficiently for TSP challenges. In the context of inland waterway shipping, the model's ability to handle data sets representing harbours underscores its practical relevance. The primary benefits of this adaptation include near-optimal solutions, fast computation times, and enhanced accuracy, making it a formidable alternative to traditional TSP solvers.

3 Job-Shop Scheduling Problem

3.1 Intro

Job Shop Scheduling Problem (JSSP) is a famous Combinatorial Optimisation Problem (COP) in computer science and production planning. In JSSP, several jobs with a predefined order of processing sequence must go through a specific machine. To solve JSSP, a schedule has to be produced that minimises the makespan of the production time while also obeying the problem's constraints. Typically, JSSP is solved using a heuristic method such as the Priority Dispatching Rule (PDR) (Haupt, 1989). However, the heuristic method has the drawback of being very costly and time-consuming to develop, requiring a lot of specific knowledge, and becoming less effective when the JSSP size increases. Because of this, DRL has become a hot topic in solving JSSP.

When using the DRL approach, an environment has to be designed to be the agent's playground. The agent is put inside this environment to learn to solve the problem and be rewarded if it can produce a good solution. Therefore, one critical aspect of designing an efficient DRL is having a proper environment design. For JSSP, the environment is created using the disjunctive graph as a base (Balas, 1969). However, to use the graph as input, extra work is required.

There are also attempts to solve the COP using GNN. For instance, the travelling salesman problem, graph optimisation problems, and the satisfiability problem, as already mentioned in the previous chapters. However, there has not been much research on disjunctive graphs. Several studies have used GNN to represent disjunctive graphs (Hameed & Schwung, 2020; C. Zhang et al., 2020). More studies are reviewed in the literature section. However, they did not focus on developing GNN because they mainly tried to solve JSSP with DRL while using GNN as an embedding method for declaring disjunctive graphs. Therefore, the GNN used in those research may not be the most optimal GNN to represent disjunctive graphs. The lack of research on GNN to represent disjunctive graphs of JSSP motivates this thesis to explore the GNN architecture.

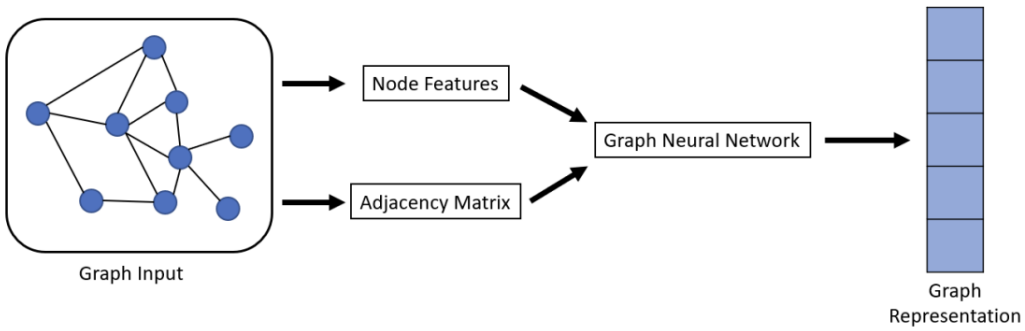

Figure 10: Workflow of a Graph Neural Network transforming node features and adjacency matrix from a graph input into a learned graph representation.

3.2 JSSP formulation

Before presenting the JSSP problem formulation, it is crucial to grasp the concepts of graph types and their distinctive attributes. Subsequently, an illustrative problem instance is provided, leading to a subsequent presentation of the mathematical problem formulation.

3.2.1 Types of graphs

A graph, denoted as $G(\mathbf{V}, \mathbf{E})$, consists of two main elements: a set of nodes $\mathbf{V}$ and a set of edges $\mathbf{E}$. These edges link nodes in the graph G. A specific node is represented as $v_i \in \mathbf{V}$, and the connection between node v_i and node v_j in a graph is represented as an edge e_{ij}, where e_{ij} is the ordered pair (v_i, v_j). The existence and orientation of edges are depicted using the adjacency matrix, $\mathbf{A}$. If $A_{ij} = 1$, this indicates the presence of the edge e_{ij}, whereas $A_{ij} = 0$ indicates its absence.

Every node possesses attributes, which are presented in matrix form as

$$\mathbf{X} \in \mathbf{R}^{n \times d}$$

where n represents the total number of nodes and d is the dimension of the node's features. Similarly, edges can also have features denoted as

$$\mathbf{X}^e \in \mathbf{R}^{m \times c}$$

with m indicating the number of edges and c specifying the dimension of edge attributes.

Graphs can be differentiated in several ways: directed versus undirected, homogeneous versus heterogeneous, and weighted versus unweighted. Detailed explanations for each type of graph follow in subsequent sections.

Directed/undirected graph Graphs can be categorised according to the nature of their edges as directed or undirected. In a directed graph, the edges have a clear starting and ending point, linking one node to another. On the contrary, undirected graphs have edges that connect two nodes without a specific direction. You can represent an undirected edge with two opposing directed edges between the nodes. Directed edges often indicate a flow or direction in the graph, whereas undirected edges suggest a mutual relationship between nodes.

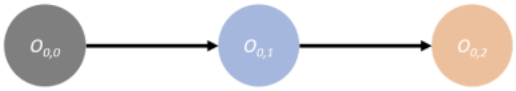

Figure 11: Example of directed and unweighted graph.

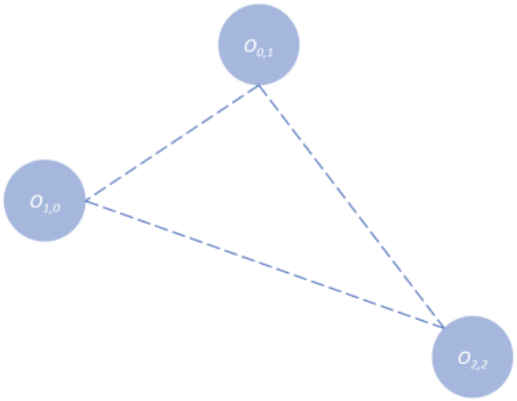

Figure 12: Example of undirected and unweighted graph.

Homogeneous/heterogeneous graph Based on their features, graphs can be classified as homogeneous or heterogeneous. In homogeneous graphs, all nodes and edges share the same type. Conversely, heterogeneous graphs have nodes or edges, sometimes both, of varying types. For instance, a graph with both directional and nondirectional edges falls under the heterogeneous category. Similarly, a graph where nodes have distinct data types, such as numbers and text in another, is also termed heterogeneous.

Figure 13: Example of graph with heterogeneous nodes properties.

Weighted/unweighted graph is a classification to determine whether the edges in a graph are assigned weight. The weight on the edge denotes the cost of using the edge. For instance, in Figure 14 the distance between $O_{0,0}$ to $O_{0,1}$ is 2.0.

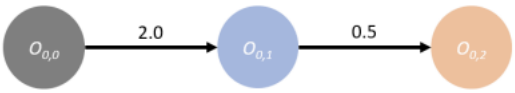

Figure 14: Directed and weighted graph.

3.2.2 Distinctive graphs

As mentioned in the introduction section, JSSPs are commonly visualised using a disjunctive graph as seen in Figure 15. The disjunctive graph is widely used because it can easily describe the constraints of JSSP. The disjunctive graph is considered as a heterogeneous graph because it contains both directed and undirected edges. The directed edges of the graph describe the priority sequence of operation within the same job, whereas the undirected edges describe the relationship of operation that uses the same machine. The start and termination of a disjunctive graph are noted by node S and T, respectively. Both nodes have no time value attached to them, as they are just a landmark. Each operation node O_{ij} contains attributes of machine number and processing time.

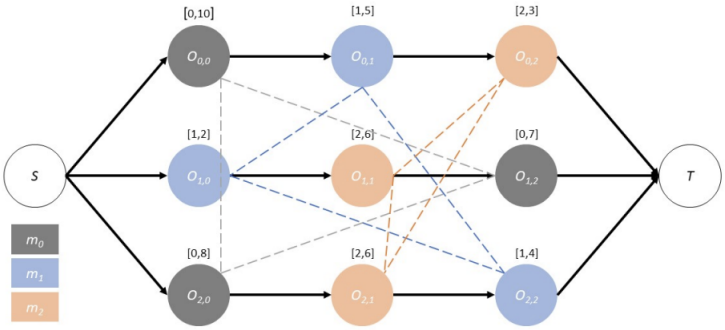

Figure 15: Disjunctive graph for a 3x3 Job Shop Scheduling Problem, illustrating directed and undirected edges to represent operational constraints and sequences.

3.2.3 Problem example

As outlined in Zalzala and Fleming, 1997, the JSSP is a notable combinatorial optimisation task. We consider n tasks $J_j, (1 \leq j \leq n)$ to be processed on m machines $M_r, (1 \leq r \leq m)$. Each task J_j follows a specific sequence of machines, M_r. When J_j is processed on M_r, it is termed operation O_{jr}, with a duration p_{jr}. Operations must follow a set order, and the schedule consists of completion times c_{jr}. The makespan is the overall time to complete all tasks is the makespan, denoted by L. The optimisation goal is to minimise this makespan, as shown in Equation 14.

$$L = \min_{j,r}\{C_{jr} = S_{jr} + p_{jr}\} \tag{14}$$

Table 5: Example of 3x3 JSSP instance example with three jobs and three machines.

job	Machine Number (Processing time in minutes)		
1	1(3)	2(3)	3(3)
2	1(2)	3(3)	2(4)
3	2(3)	1(2)	3(1)

Table 5 presents a 3x3 JSSP example with three jobs, each comprising three operations. Each operation is assigned to a machine with a specific processing time indicated in parentheses. A Gantt chart, shown in Figures 16, effectively illustrates the resulting schedule from solving the JSSP.

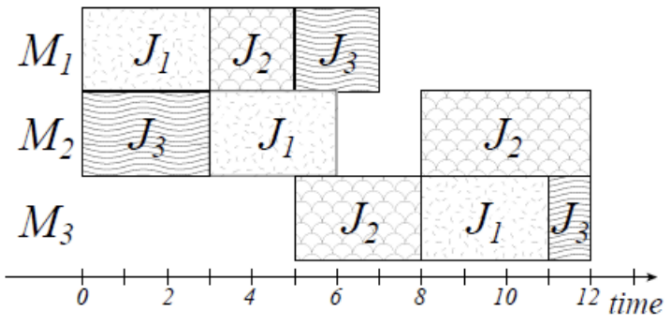

Figure 16: A Gantt chart solution of 3x3 JSSP with time in minutes (Zalzala & Fleming, 1997).

3.2.4 Mathematical Representation of JSSP

Let's examine the mathematical description of JSSP more closely. Define:

- $J = \{1, 2, \ldots, n\}$ as the job set.
- $M = \{1, 2, \ldots, m\}$ as the machine set.
- O_{ij} as the j-th operation for job i.
- p_{ij} as the duration needed for the j-th operation of job i.
- S_{ij} and C_{ij} as the start and finish times of the j-th operation for job i, respectively.
- $x_{ij,k}$ as a binary variable: 1 if the j-th operation of job i utilises machine k, otherwise 0.

The goal is to minimise the total completion time:

$$\min \max_{i \in J} C_{i,n_i} \tag{15}$$

Constrained by:

1. Every operation is executed once:

$$\sum\nolimits_{k \in M} x_{ij,k} = 1 \qquad \forall i \in J, \forall j \in \{1, \ldots, n_i\} \tag{16}$$

2. Job operations must follow their designated order:

$$C_{i,j-1} \leq S_{ij} \qquad \forall i \in J, \forall j \in \{2, \ldots, n_i\} \tag{17}$$

3. Avoiding operation overlap (if the j-th operation of job i precedes the l-th operation of job h on an identical machine, then $C_{ij} \leq S_{hl}$ or its reverse):

$$S_{ij} + p_{ij} x_{ij,k} \leq S_{hl} + M(1 - x_{ij,k}) \qquad \forall i, h \in J, \forall j \in \{1, \ldots, n_i\}, \forall l \in \{1, \ldots, n_h\}, \forall k \in M \tag{18}$$

Here, M represents a sufficiently large constant.

Given this mathematical representation of the JSSP, we can use various analytical techniques and strategies to navigate this complex scheduling task.

3.3 Literature

Our literature review takes a distinctive route in the realm of the Job Shop Scheduling Problem. While a comprehensive literature review remains limited for the JSSP, we focus on a novel approach leveraging unique graph structures within a Graph Neural Network architecture for JSSP resolution. As a result, the literature review dedicated to JSSP remains succinctly listed in a dedicated subsection and is focused on leveraging DL techniques to solve JSSP. Unlike the more expansive literature reviews for the Travelling Salesman Problem and the Resource-Constrained Project Scheduling Problem, our emphasis on integrating distinctive graphs with GNNs for JSSP resolution is our only focus on this challenging optimisation issue.

Various reinforcement learning (RL) techniques have been employed in the context of JSSP and their variations in several research articles. However, it should be noted that our primary focus does not revolve around RL methods for solving JSSPs. Our investigation has revealed that the existing literature primarily emphasises the application of RL methods to distinctive graph-related JSSP rather than directly addressing with GNN.

To the best of our knowledge, the first article that uses the RL method to solve JSSPs is W. Zhang and Dietterich, 1995, in which a combination of an RL method and iterative repair technology is applied to NASA's space shuttle payload processing and scheduling problem. Another successful application of RL in JSSPs is Aydin and Öztemel, 2000, an improved Q-learning method named Q-III is applied to a JSSP to help heuristics avoid falling into local optima. X. Chen and Tian, 2018 proposes an algorithm called neural rewriter to improve the performances on job scheduling and expression simplification problems. On the other hand, for JSSPs in dynamic environments, researchers prefer to use RL methods alone because of their low computational cost. In S. Wang et al., 2007, a Q-learning-based method is proposed to solve a single-machine multi-job scheduling problem in a dynamic environment. Mao et al., 2016, W. Chen et al., 2017 mainly focus on a JSP in data centre networks, and a near-real-time decision-making method is needed because the problem scenario is highly dynamic. Liu et al., 2020 proposes an actor-critic network-based DRL method to solve JSSPs; this method has outperformed constructed heuristics (FIFO, LIFO, SPT, etc.).

However, some problems remain with regard to applying RL to solve JSSPs. Research on how to model JSSPs with RL is still insufficient, and existing RL methods for JSSPs lack generalisation ability, which leads to the resulting RL models being re-trained for JSSPs with different problem scales. In addition, traditional RL methods must handcraft model features, but some domain knowledge is still needed.

As graph-structured models have been used in reinforcement learning domains in Almasan et al., 2022, a graph embedding is given to represent JSSPs with different problem scales.

Furthermore, Hameed and Schwung, 2020 and C. Zhang et al., 2020 only use simple GNN architectures to encode the disjunctive graph. As expressed by Cappart et al., 2022, the limitations of the GNN approaches to COP are the expressive power of GNNs and the generalisation of GNNs. Moreover, each GNN architecture is designed very differently from one another. Thus, their performance and capabilities differ, especially since they are designed to tackle specific problems. Therefore, the GNN used in those research might not be the most optimal GNN for representing disjunctive graphs. The lack of research on GNN in representing disjunctive graphs of JSSP motivates this thesis to explore different GNN architectures to find the most expressive GNN to represent it.

Table 6 summarises the reviewed literature on the Job Shop Scheduling Problem. Again, in light of time constraints and our concentrated focus on the GNN architecture, a comprehensive literature review of the JSSP was not undertaken. Instead, our priority centred on delving into the intricacies of the GNN framework to address this challenge effectively.

Table 6: Literature overview of JSSP literature

Article	Subject
W. Zhang and Dietterich, 1995	RL applied to NASA's space shuttle payload processing and scheduling problem.
Aydin and Öztemel, 2000	Improved Q-learning method (Q-III) applied to JSSPs.
X. Chen and Tian, 2018	Proposal of the neural rewriter algorithm for job scheduling and expression simplification.
S. Wang et al., 2007	Q-learning-based method for solving a single machine multi-job scheduling problem in a dynamic environment.
Mao et al., 2016 and W. Chen et al., 2017	Focus on JSP in data centre networks and near-real-time decision making methods.
Liu et al., 2020	Actor-critic network-based Deep RL (DRL) method for JSSPs.
Almasan et al., 2022	Use of graph embedding to represent JSSPs.
Hameed and Schwung, 2020 and C. Zhang et al., 2020	Utilisation of simple GNN architectures to encode disjunctive graphs.
Cappart et al., 2022	Discussion of limitations and challenges of GNN approaches to Combinatorial Optimisation Problems (COP).

3.4 Data used

This thesis requires data, as shown in Table 5, where there are jobs with a processing sequence that mentions the machine to which it needs to go and the time needed for the process. This thesis uses self-generated data sets of Job Shop Scheduling Problems. There are 3x3x3, 6x6x6, 8x8x8, 10x10x10, and 15x15x15 JSSP and are generated as Samsonov et al., 2021. The dimension of the problem represents

the number of jobs, operations, and machines in the problem. For simpler referencing, the problems are being called $n \times n$ instead of $n \times n \times n$ from this point forward.

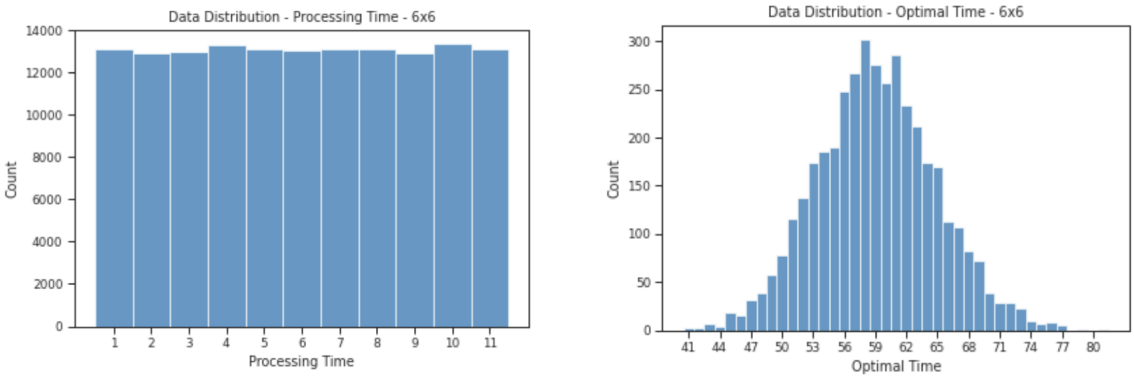

Figure 17: Uniform processing time distribution across machines for a 6x6 Job Shop Scheduling Problem dataset, highlighting randomised machine assignments and time ranges from 1 to 11. Data distribution for the optimal time have a bell-shaped normal distribution.

For data sets 3x3, 6x6, 8x8, 10x10, and 15x15, the processing times of each process range from $[1, 11]$. The processing time counts are evenly distributed, as seen in Figure 17. Each process is assigned to one machine randomly. Besides the data sets mentioned, this thesis also uses L2D data sets generated by (C. Zhang et al., 2020). These L2D data sets have an enormous processing time magnitude compared to the regular dataset, ranging from $[n, 98]$, where n is the size of the JSSP. For instance, 6x6L2D has a range of $[6, 98]$, as seen in Figure 18. Furthermore, the processing time is uniformly distributed and randomly assigned to one machine, similar to regular data sets. In this thesis, two L2D data sets are used, namely 6x6L2D and 15x15L2D. These L2D data sets are used to see if the magnitude of the time changes impacts GNN performance in learning the disjunctive graph structure. All data have the optimal time in a bell-shaped normal distribution. This is a good sign that the data are distributed properly, and with normal distribution, it helps the model to train better to achieve a higher precision.

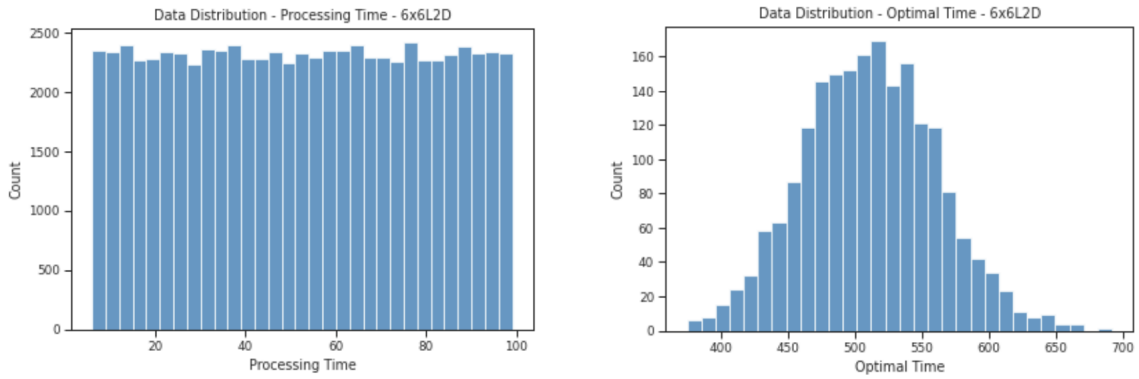

Figure 18: Comparative data distribution in job Shop Scheduling Problems, illustrating even spread in processing times for standard and extended range (L2D) data sets. Data distribution for the optimal time have a bell-shaped normal distribution.

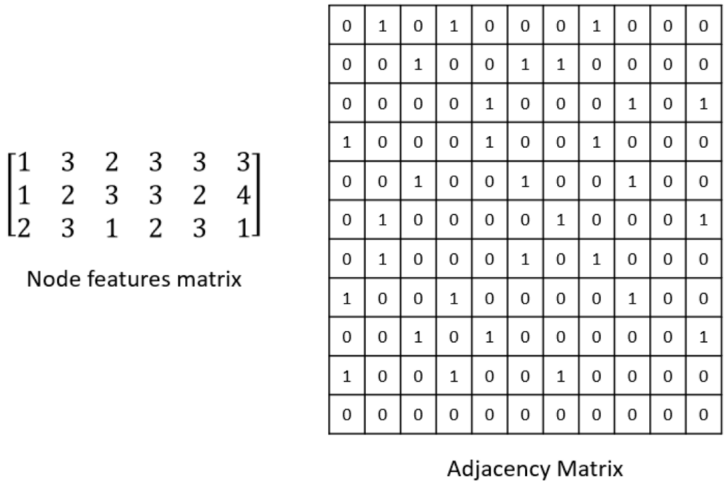

Figure 19: Example of node features matrix and adjacency matrix.

JSSP data sets usually have the form of a matrix or table, like in Table 5. They do not have the problem directly in the structure of a disjunctive graph, but the matrix or table has an order to it so that the sequence of the number has a position value. For this thesis, the generated problems have JSON file formatting that has to be converted before being used for training a model. The data is converted to form a disjunctive graph with a node feature matrix and an adjacency matrix, as seen in Figure 19. Essential features are extracted from the JSON file and are optimal time, job number and operation sequence, machining time, and machine number. These extracted values are then recompiled to Pytorch Geometric (Fey & Lenssen, 2019) data format to run appropriately on GNN models in the library. The data format consists of three things: edges, node features, and a label. The edges show the direction of the edge from the starting node to the target node. Since the disjunctive graph has two types of edges, the undirected edges are converted into two directed edges pointing back and forth of the node pair, as seen in Figure 20. This helps simplify the problem formulation of the graphs because most GNNs cannot process different types of edges. The node features show the machine number n_j and processing time p_{ij} in the format of the array $[n_j, p_{ij}]$. Finally, the label of the data is the optimal time of the disjunctive graph.

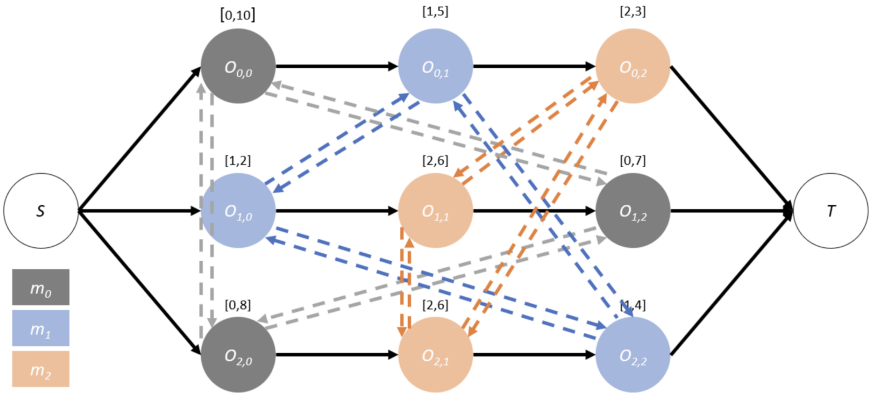

Figure 20: Example of a disjunctive graph with all edges directed, showing the conversion of undirected edges into directed ones to facilitate processing by GNN models. Each node is labelled with machine number and processing time, and each directed edge represents the flow from one operation to the next within job scheduling scenarios.

The node features in the data are normalised and scaled to values between 0 and 1 because normalisation helps the model converge faster, as described in Sola and Sevilla, 1997. In a disjunctive graph, there is also a starting node and a terminating node. With the node features normalised to values between 0 and 1, the machine number for the starting node and the terminating node will be assigned values 2 and 3 to be outside the range of the machine number. As for the processing time, it will be 0 because it is not part of the processing.

For the baseline, it will use the same data sets. However, due to some limitations, the node features are the only input for the baseline, while the edges are not included. Table 7 shows the general information of the data sets. It has to be noted that not all data sets have an equal amount of graphs.

Table 7: Table summarising the dataset statistics, detailing the number of graphs, nodes, and edges for various graph data sets

Datasets	**DG3x3**	**DG6x6**	**DG8x8**	**DG10x10**	**DG15x15**	**DG6x6L2D**	**15x15L2D**
# of graphs	1000	4000	4000	4000	4000	2000	1000
# of nodes	11	38	66	102	227	38	227
# of edges	30	222	520	1010	3390	222	3390

Data used in Inland Waterway Transport context The JSSP problem is well-suited for inland waterway transport scenarios, which can be adapted to the operations within a port call. In such settings, a port services a variety of vessels, each with distinct operational requirements during their berthing period. These ships, which are either bulk carriers or container vessels, demand specific tasks such as bunkering, connecting to shore power, loading, and unloading.

For clarity, we typically encounter either a bulk carrier that requires bulk cargo operations or a container ship that deals with container handling. It is uncommon for a single vessel to switch between these two types of cargo handling during the same port call. Instead, other necessary tasks like bunkering or setting up shore power connections may be required.

For example, Ship A might need to bunker fuel and unload its bulk cargo, while Ship B might connect

to shore power before loading containers onto its deck. Ship C may start with unloading containers followed by a bunkering operation. Each task has a predetermined duration and must adhere to a specific sequence, such as not commencing bunkering before completing the unloading process.

The goal is to find a schedule that minimises the total completion time of all tasks while adhering to the resource constraints and task dependencies, considering the limited availability of resources like tugs, pilots, and dock workers. The JSSP framework provides a foundation for this, treating each ship as a "job" and each operation as a "task", with the objective being to optimise the schedule to ensure efficient turnover while complying with operational constraints.

Unlike the TSP, which is primarily concerned with finding efficient routes, the applicability of the JSSP extends to optimising the intricate scheduling of port operations and ship tasks. In this maritime context, each ship's visit to a port can be likened to a "job", and the tasks required for loading, unloading, and maintenance resemble "operations". Resource constraints, such as the availability of cranes, labour and berths, closely parallel the constraints in the JSSP. By adapting and applying the JSSP framework to the dynamic environment of port operations and inland waterway transportation, stakeholders can allocate resources, sequence tasks, and reduce ship turnaround times, enhancing efficiency in the maritime supply chain. This demonstrates the flexibility and practicality of the JSSP beyond its traditional domains and its potential to contribute to the optimisation of diverse industrial scenarios, including multifaceted operations of port facilities and waterway logistics.

3.5 Methodology

3.5.1 Graph Neural Network

Graph Neural Network enhances traditional machine learning and deep learning models by accommodating graph data. GNN aims to convert graph structures and node features, represented as $\mathbf{x_v}$, into node representations $\mathbf{h_v}$ or overall graph representations $\mathbf{h_G}$. A feature of GNN is its message-passing or neighbourhood aggregation mechanism. Here, every node aggregates information from adjacent nodes to refine its representation. This aggregation process repeats for k iterations, ensuring the node's representation includes details from its surrounding k-hop neighbourhood by the end. Figure 21 shows a visual representation of this.

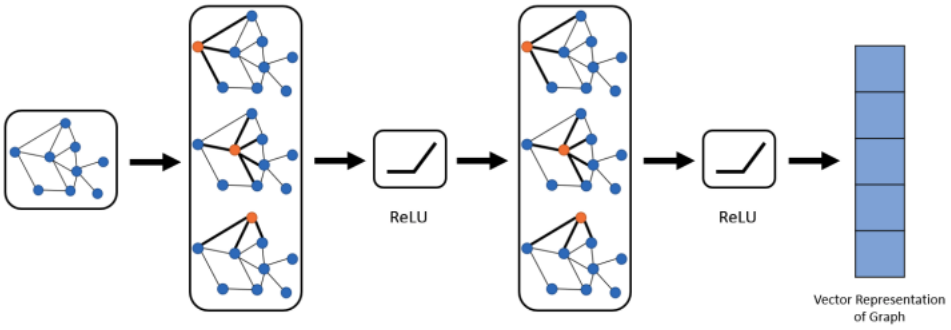

Figure 21: Sequential transformation of a graph through a Graph Neural Network architecture, highlighting feature extraction and the application of ReLU activations, culminating in a compact vector representation of the graph.

$$\mathbf{h}_v^{(0)} = \mathbf{x}_v \tag{19}$$

$$\mathbf{a}_v^{(k)} = AGGREGATE^{(k)}(\{h_v^{(k-1)} : u \in N(v)\}) \tag{20}$$

$$\mathbf{h}_v^{(k)} = COMBINE^{(k)}(h_v^{(k-1)}, \mathbf{a}_v^{(k)}) \tag{21}$$

Equations 19, 20, and 21 describe the message-passing strategy. $\mathbf{h}_v^{(k)}$ is the feature vector of node v at the k-th iteration/layer. Equation 19 is used for initialisation, where $\mathbf{x}_v$ is the initial node features. Equation 20 describes the aggregation of the initial node embedding and its neighbours embedding, denoted by $N(v)$. After obtaining the new embedding, an update is conducted using Equation 21. $AGGREGATE^{(k)}(\cdot)$ and $COMBINE^{(k)}(\cdot)$ are parameterized functions. Choosing both functions is crucial because different functions lead to a different GNN architecture.

For the graph classification problem, GNN will compute the graph embedding $\mathbf{h}_G$ using the readout function Equation 22. This equation uses the final iteration of the node representation $h^{(K)}$ as input. READOUT is a graph-level pooling function such as sum, mean, or max.

$$h_G = READOUT(\{\mathbf{h}_v^{(K)} | v \in G\}) \tag{22}$$

3.5.2 Graph Convolutional Network

Kipf and Welling, 2017 proposed a Graph Convolutional Network inspired by Convolutional Neural Network (CNN) used in image classification problems. Convolution works by applying a spatial filter to the input image and getting a feature map as a result. However, CNN cannot be applied to a graph because, unlike an image, a graph does not have a natural order of nodes, whereas CNN works by stacking the image feature in a specific order. Similarly to CNN, GCN applies a spatially moving filter over the nodes of the graph to obtain the feature representation of each node. Stacking multiple GCN layers, like in CNN, also works to extract high-level node representation.

$$AGGREGATE^{(k)}(\{h_v^{(k-1)} : u \in N(v)\}) = sum_{u \in N(v)} \frac{\mathbf{h}_v^{(k-1)}}{\sqrt{deg(v)deg(u)}} \tag{23}$$

$$COMBINE^{(k)}(h_v^{(k-1)}, \mathbf{a}_v^{(k)}) = \sigma(\mathbf{W}^{(l)}\mathbf{a}_v^k \tag{24}$$

$$\mathbf{h}_v^{(k)} = ReLU(\mathbf{W} \cdot MEAN\{\mathbf{h}_v^{(k-1)} : \forall u \in N(v) \cup \{v\}\}) \tag{25}$$

Kipf and Welling, 2017 used the element-wise mean pooling function for its $AGGREGATE^{(k)}(\cdot)$ as seen in Equation 23. Equation 25 shows the complete form of $AGGREGATE^{(k)}(\cdot)$ and $COMBINE^{(k)}(\cdot)$ in GCN, with ReLU as the choice for its nonlinear activation function.

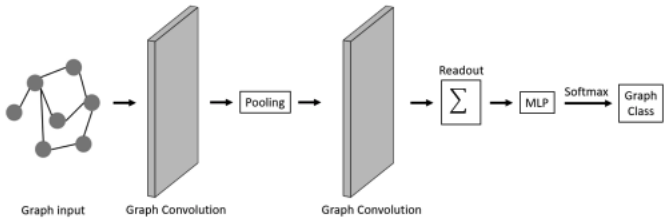

Figure 22: This figure depicts the architecture of a Graph Convolutional Network for graph classification. It demonstrates the process flow from graph input through successive graph convolution layers, followed by a pooling layer to down-sample and capture substructures. Subsequently, a readout layer aggregates node features into a graph representation, which is then passed through a Multilayer Perceptron (MLP) and a softmax function for the final classification of the graph.

The GCN architecture for the graph classification problem can be seen in Figure 22. It combines the GCN layer with the pooling layer and readout layer. The GCN layers are responsible for extracting the node features, and graph-pooling layers are used to down-sample, which helps the model to split a graph into smaller substructures. The readout layer combines node representations into graph representations. After acquiring the graph representation, Multilayer Perceptron (MLP) and softmax can be used to do the classification.

3.5.3 Graph Isomorphism Network

Xu et al., 2018 proposes the Graph Isomorphism Network (GIN). The motivation of GIN comes from trying to solve the graph isomorphism problem. Graph isomorphism refers to graphs having the same number of nodes, edges, and the same adjacency matrix but a different form; that is, if they are drawn, they have a different visual structure. Figure 23 shows an example of an Isomorphic graph, where Figure 23 (a) and (b) have the shape of a pentagon and pentagram, respectively. If traced carefully, the nodes' and edges' connections in both shapes are identical but only visually different.

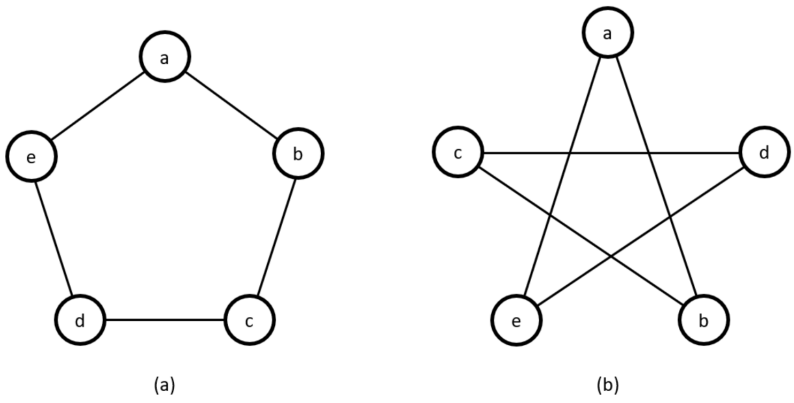

Figure 23: Example of a pair of isomorphic graph.

According to Xu et al., 2018, for an optimal GNN to distinguish non-isomorphic graph structures by mapping them to distinct embeddings in the embedding space, it's crucial to ensure injectivity in both the neighbour aggregation and graph-level readout functions. Equation 26 shows the aggregation method of the sum of its neighbouring nodes. Equation 27 introduces scalar parameter (k) and MLP. Inserting

Equation 26 into Equation 27 create Equation 28.

$$AGGREGATE^{(k)}(\{\mathbf{h}_v^{(k-1)} : u \in N(v)\}) = \sum_{u \in N(v)} \mathbf{h}_v^{(k-1)} \tag{26}$$

$$COMBINE^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)}) = MLP^{(k)}((1 + \epsilon^{(k)}) \cdot \mathbf{h}_v^{(k-1)} + \mathbf{a}_v^{(k)}) \tag{27}$$

$$\mathbf{h}_v^{(k)} = MLP^{(k)}((1 + \epsilon^{(k)}) \cdot \mathbf{h}_v^{(k-1)} + \sum_{u \in N(v)} \mathbf{h}_v^{(k-1)}) \tag{28}$$

For node classification and link prediction problems, GIN can learn the node embeddings and use them to solve them. As for the graph classification problem, a readout function is necessary to produce the embedding of the entire graph from the node embeddings.

$$\mathbf{h}_G = CONCAT(READOUT(\{\mathbf{h}_v^{(k)} | v \in G\}) | k = 0, 1, ..., K) \tag{29}$$

Extending the node embedding learned using Equation 28, Xu et al., 2018 proposes Equation 29 graph-level readout function that process and combines individual embedding of nodes into graph embedding $\mathbf{h}_G$. Equation 29 concatenates learned graph representation from all iterations of the model so that GIN can generalise better. Xu et al., 2018 also suggested using the sum pooling method because it has better expressive power than the mean and max pooling methods, as the research proved.

3.5.4 Experimental Set up

The prepared data sets have disjunctive graphs and optimal time as graph labels. However, before converting the raw data sets into disjunctive graphs, it is interesting to see if another form of graphs can be more suitable for representing JSSP. Research indicates that disjunctive graphs are the only graph structure capable of appropriately representing JSSP (Balas, 1969; Błażewicz et al., 2000). Due to the lack of research on using GNN for disjunctive graphs, exploring different forms of disjunctive graphs and finding alternative structures is necessary. A disjunctive graph comprises a starting node, processing nodes, and a terminating node, as illustrated in Figure 24. However, the starting node and terminate node in the graph do not have any value. Therefore, it will not affect the makespan of the JSSP if removed. Thus, an experiment can be conducted to see the importance of the starting and terminating nodes. The experiment compares the performance of GNN in learning the complete disjunctive graph and in learning the graph without starting and terminating nodes. Another motivation for this experiment comes from trying to adjust the disjunctive graph as a molecular structure (Wieder et al., 2020). The default structure of the disjunctive graph is similar to a flow network, where there is a start node and a terminate node to define the direction of the flow in the graph. However, suppose the start and terminate nodes are removed, leaving behind the processing nodes. In that case, this structure is comparable to a molecule structure because a molecule structure has no starting point or ending point. GNNs are also known to be used for processing molecule graphs for classification problems. Thus, it is interesting to see if disjunctive graphs can be treated this way so that GNN can learn disjunctive graphs better.

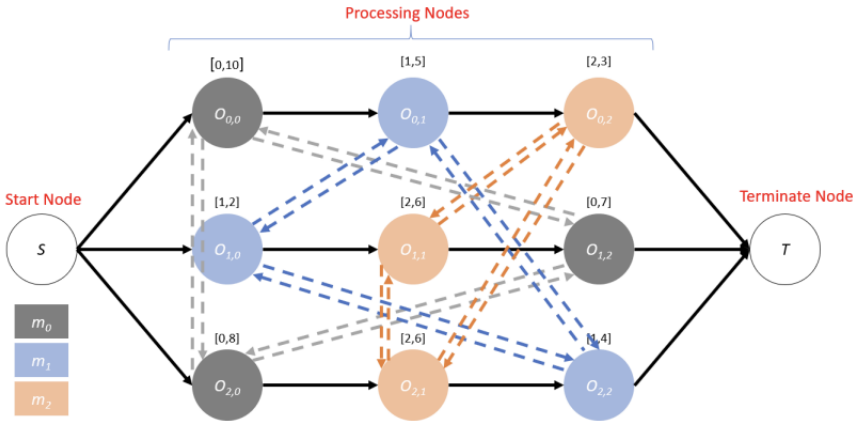

Figure 24: Example of disjunctive graph with starting node, processing nodes, and terminate node. Each node is labeled with machine number and processing time, and each directed edge represents the flow from one operation to the next within job scheduling scenarios.

Besides removing the starting and terminating node in a disjunctive graph, another essential aspect is normalisation for the node features. Several GNN studies highlight their practice of data normalisation prior to training with GNN (Kipf & Welling, 2017; C. Zhang et al., 2020). In contrast, other works do not indicate the normalisation of node features (B. Chen et al., 2020; Hameed & Schwung, 2020). Consequently, the necessity of normalisation remains a topic of debate. Given this uncertainty, it's essential to investigate whether normalising node features is crucial for disjunctive graphs.

After finalising the structure of the disjunctive graphs based on previous experiments, the last experiment measures the model's performance. Because the data sets consist of graphs with a label, a regression task can be performed in which the GNN can learn a vector representation of each graph, and this vector is fed into MLP to predict the makespan of the graph. The expected value is then compared with the actual optimal time to see how the model performed.

In this study, we trained the GCN model with varying numbers of layers, ranging from 1 to 10. Our experimental results indicated that GCN models with layers between 2 and 5 consistently outperformed models with other numbers of layers. Based on this finding, we established the following GCN model setup: the number of layers is constrained to be in the range [2, 5], the hidden dimension can take values from the set 64, 128, and we used global pooling with the mean aggregation method.

For the GIN model, our focus was on the GIN-0 variant, incorporating a learnable parameter ϵ. While earlier investigations (Xu et al., 2018) have highlighted the significance of this parameter, our observations indicate that there is often no noticeable performance enhancement when ϵ is assigned a value of 0; this setup frequently promotes better model generalisation. Therefore, our implementation aligns with the GIN-0 model configuration throughout this thesis. The specific configuration details are: 5 GIN-0 layers, including the input layer, with each layer embedding a two-layer MLP; Possible hidden dimensions are sourced from the set {64, 128}; and batch normalisation is consistently incorporated across all hidden layers.

3.5.5 Evaluation Metrics

This section highlights the criteria used to assess the model's efficacy. This study uses the mean squared error (MSE) and the R2 score to compare the efficiency of the models in the regression analysis.

Mean Squared Error MSE described the approximation accuracy of a regression line with respect to the data points. It quantifies the gap between the model's predictions and the actual values, squaring this difference. An average of these squared differences gives the MSE. The formula for MSE, represented in Equation 30, considers N as the total data points, y_i as the true value and $\hat{y}_i$ as the estimated value.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{30}$$

R^2 score - R2 score is crucial for assessing the performance of a regression model. It quantifies how well the model can explain the variability in the variable. This metric is instrumental in comparing a model with a constant baseline, offering insights into the model's relative performance. The baseline is typically determined by ascertaining the mean of the data and drawing a line corresponding to this mean. Equation 31 shows the computation for the R2 score. Here, SS_{res} signifies the residual sum of squares, while SS_{total} denotes the total sum of squares. In this equation, y_i represents the actual value, $\hat{y}_i$ is the predicted value, and $\overline{y}_i$ indicates the average of the real values.

$$R^2 = 1 - \frac{SS_{res}}{SS_{total}} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y}_i)^2} \tag{31}$$

The potential values for the R2 score span from $[-\inf, 1]$. A negative R2 score may arise from two primary causes: either the model does not align with the baseline's trend, or the model contains a substantial count of outliers, amplifying the numerator's value over the denominator. For instance, an R2 score of 0.6 implies that the model captures most of the data points, signifying their clustering around the regression line. On the contrary, an R2 score of 0.11 suggests that the data points are more dispersed, and the model captures only a fraction of them. Therefore, a diminished R2 score implies a reduced model fit, highlighting the divergence of the data points from the regression line.

3.6 Results

A series of experiments are conducted to understand the impact of design choice for disjunctive graphs towards GNN. As mentioned in the previous methodology section, only a few studies use disjunctive graphs to train a DRL model. However, they do not explore disjunctive graph representation using GNN explicitly (Hameed & Schwung, 2020; C. Zhang et al., 2020). Thus, the structure of disjunctive graphs needs to be researched in more detail to see if any alternatives that GNN better understands are available. Initially, this section explores the impact of node feature normalisation to determine its necessity, given that prior research presents mixed practices on this topic. Subsequently, the focus shifts to the significance of the start node S and terminate node T in disjunctive graphs. This exploration is prompted by the fact that nodes S and T lack intrinsic node feature values and act as endpoints for the graph. The underlying question is whether a disjunctive graph, consisting solely of processing nodes, can enhance the GNN's learning capability. All experiments to understand the graph structures are tested using the GCN architecture to save computational time. GCN is selected here because although GCN is the most basic GNN model, it has enough expressive power to learn from disjunctive graphs. Thus, the GCN can represent the other GNN models in finding the best configuration for the disjunctive graph. The code was implemented in Python within a Visual Studio Code environment. It incorporated several libraries, including Networkx, PyTorch Geometric, and PyTorch, utilising an 8-core Apple M1 CPU @3.20Ghz, 16GB RAM and an 8-core GPU.

3.6.1 Normalizing Node Features

Table 8 shows the loss difference in the test set for normalised and non-normalised node features. The arrow up or down symbols denote larger and smaller preferred values, respectively. In all cases, normalising the node features helps improve GCN's performance in learning disjunctive graphs. The improvement is very minimal in some data sets, but most data sets have a significant improvement, up to 89% for dataset 6x6. Even in datasets where the improvement seems marginal, such as the 3x3 or 15x15L2D with enhancements of 1.75% and 5.05%, respectively, normalising still leads to better results. Finally, the 8x8 and 10x10 datasets display improvements of 65% and 95%, respectively. Similarly, the 15x15 dataset shows a 26% boost, while the 6x6L2D dataset records a 22.52% enhancement.

Based on these results, it can be concluded that the normalisation of node features has to be done to improve the model's performance. Otherwise, there is a risk that the model learns poorly from the disjunctive graph.

Table 8: Table showing the comparison of test loss between normalised and non-normalised node features in a GCN, where normalisation leads to significant improvements (MSE $\downarrow$)$inlearningdisjunctivegraphsacrossvariousdatasets$.

	3x3	**6x6**	**8x8**	**10x10**	**15x15**	**6x6L2D**	**15x15L2D**
Normalized	11.2	14.8	23.4	20.5	23.0	1123.4	1653.6
Not Normalized	11.4	142.8	67.1	389.3	30.9	1449.9	1741.6
Improvement	1.75%	89.64%	65.13%	94.73%	25.57%	22.52%	5.05%

3.6.2 Start Node and Terminate Node

Having demonstrated that normalisation contributes to enhanced model learning, the subsequent analysis investigates the influence of the starting node S and the terminating node T on the architecture of a disjunctive graph. The start node and terminate node do not have any node feature value. Therefore if they are removed, it will not affect the calculation towards the final makespan. In addition, when removed, the structure of the disjunctive graph is similar to a molecule graph. Table 9 presents the losses for the model when learning a disjunctive graph with nodes S and T compared to one that does not have them. For data sets 3x3, 8x8, and 10x10, there is an increase in performance between 23.9% to 35.32%. On the other hand, the rest of the data sets did not show any improvement and performed very similarly for both types of graphs. The losses in some of the data sets are nearly the same. Based on this finding, it is hard to tell if the nodes S and T play a significant role in defining a disjunctive graph's structure. Nevertheless, the advantages of having those nodes outweigh the disadvantage of not having them. From this, it can be said that it is better to have the start and terminate node. The disjunctive graph should be handled as a flow network instead of a molecule graph, as it is proven that the model can learn better from it.

Table 9: Table comparing the test loss of a GCN when learning from disjunctive graphs with and without the start node (S) and terminate node (T). (MSE $\downarrow$)

	3x3	**6x6**	**8x8**	**10x10**	**15x15**	**6x6L2D**	**15x15L2D**
With	7.7	14.8	14.9	15.6	23.2	1135.0	1677.4
Without	11.2	14.7	23.4	20.5	23.0	1123.4	1653.6
Improvement	31.25%	-0.68%	35.32%	23.9%	-0.87%	-1.03%	-1.44%

The above results prove that a normalised complete disjunctive graph is the better input structure for GCN. In the following section, all data sets are normalised and include the starting node S and the terminating node T.

3.6.3 GNN Models Performance

This section evaluates the performance of all models, encompassing both the state-of-the-art and baseline approaches. Table 10 presents the performance metrics for each dataset. Similarly, Table 11 displays the optimal R2-score for every dataset. It has to be noted that not all data sets have an equal amount of graphs. The statistics of the data sets are mentioned in Table 7. The first part of this section shows the result table of the models' performance. The next part of the analysis talks about the different performances of the architectures when processing L2D data sets. After that, the performance of each model is discussed based on the MSE loss and R2-Score simultaneously.

Table 10: MSE loss of various GNN models on test sets across different sizes of Job Shop Scheduling Problem datasets, indicating the performance of each model with highlighted best results.

		3x3	**6x6**	**8x8**	**10x10**	**15x15**	**6x6L2D**	**15x15L2D**
GNN	GCN	8.9 ± 4.1	12.1 ± 5.0	15.1 ± 4.4	17.5 ± 3.1	19.9 ± 5.8	955.4 ± 143.2	1317.8 ± 126.4
	GIN-0	9.8 ± 2.0	14.8 ± 5.4	17.3 ± 2.6	16.4 ± 3.4	27.0 ± 3.8	1533.0 ± 472.5	1487.0 ± 126.4
Baseline	MLP	9.5 ± 5.7	13.06 ± 5.5	16.6 ± 2.9	19.7	-	956.0 ± 31.7	-

Table 11: R^2 scores, representing the accuracy percentages of GNN models across various Job Shop Scheduling Problem test data sets, contrasting the performance with an emphasis on the highest scores for each dataset.

		3x3	**6x6**	**8x8**	**10x10**	**15x15**	**6x6L2D**	**15x15L2D**
GNN	GCN	72.4 ± 1.4	58.9 ± 3.4	56.4 ± 1.0	56.3 ± 1.5	41.8 ± 4.6	58.9 ± 3.5	27.1 ± 2.7
	GIN-0	63.7 ± 6.5	53.6 ± 2.8	51.7 ± 4.6	54.3 ± 1.2	24.5 ± 9.3	47.3 ± 3.8	35.8 ± 4.0
Baseline	MLP	70.7 ± 1.3	59.7 ± 4.4	57.9 ± 0.4	56.7	-	59.1 ± 4.5	-

Standard vs. L2D data sets The difference between standard data sets and L2D data sets is that L2D data sets have an enormous processing time magnitude, as explained before. Nevertheless, all inputs are normalised so that both data sets have the same scale. However, the optimal time of the graphs is not scaled, so the MSE loss can be interpreted easier because it has the same scale as the original input before normalisation. Because of this experiment design choice, when comparing the standard data sets versus the L2D data sets, the MSE loss became useless because L2D data sets have an enormous range of optimal time.

The R2-Score remains the most reliable metric to assess the performance of the GCN and GIN models. When comparing the 6x6 model to the 6x6L2D based on the R2-Score, it's evident that the increased magnitude of processing time doesn't adversely affect the models' performance. GCN has enough expressive power to handle both versions of the 6x6 data sets. In 15x15 data sets, both regular and L2D, there is a decrease in performance for the GCN model. 15x15 dataset is already challenging to process with either GCN or GIN, the difficulties also increase, and most models suffer from it. To sum up, based on the findings, it can be said that GCN and GIN have enough capacity to process 6x6 data sets, both the regular and the L2D version of it. However, when the JSSP size increases to 15x15, all models struggle to learn the graph structure, especially if the processing time scale is increased.

Baselines Results From Table 10, it can be seen that GCN outperformed the baseline MLP. There is a difference in the loss compared to the best GCN of each data set. In the case of MLP, it can be seen that MLP has the highest loss in most of the data sets. Only datasets 15x15 and 15x15L2D reported a good result for MLP. The large MSE loss observed for MLP across numerous datasets can be attributed to its inability to train effectively in certain folds during cross-validation. At least 1 or 2 folds of the data failed to be learnt by MLP and caused the loss value to be tremendous. MLP learned graph input poorly because MLP is not designed to process graph inputs. It failed to understand the correlation of the input because it cannot take advantage of the edge connection between the nodes. This limitation made it challenging for the MLP to discern the constraints essential for determining the makespan of JSSP. Thus, using MLP to learn the representation of disjunctive graphs is not recommended. MLP can potentially approximate the representation of a disjunctive graph, provided it yields consistent outputs. Yet, one should remain cautious, as utilising MLP for graph input learning is inherently unstable. This is evident from its tendency to falter in certain folds of the datasets.

Graph Convolutional Network and Graph Isomorphism Network results The comparative analysis reveals distinct outcomes in evaluating the performance of Graph Convolutional Networks (GCN) and Graph Isomorphism Networks (GIN), specifically GIN-0, where ϵ equals 0. GCN's Mean Squared Error (MSE) loss is commendable across various datasets, as shown in Table 10, consistently outperforming GIN-0. It is particularly notable that GCN was never the least effective and remained quite competitive with GIN-0, except for larger Job Shop Scheduling Problems (JSSPs) like 15x15L2D where its R2-Score falls below 50%, as detailed in Table 11. This suggests GCN's capability to understand the structure of disjunctive graphs effectively. On the other hand, GIN-0 often incurs the largest prediction errors across almost all datasets, with the exception of dataset 15x15L2D where it leads in R2-score; however, the score is still underwhelmingly below 50%. That means that GIN-0 tends to make a big mistake when predicting the optimal time of the disjunctive graph. The main reason why GIN is not performing that well is because of its use of the *SUM* aggregation method. In Xu et al., 2018, it is explained that *SUM* is more expressive than other aggregations such as *MEAN/MAX/MIN*. The expressive power of *SUM* can be seen in Figure 25, where other aggregation methods fail to distinguish graph (a) from graph (b). Although SUM is more expressive than other aggregation methods, a big problem is faced by *SUM*, which is numerical instability. Table 13 shows an example of this. Given that the training set and the test set have the same number of nodes, if the distribution of machining operation time between the train set and the test has a difference, then the *SUM* aggregation messages will have a huge difference. This means that *SUM* aggregation is highly dependent on the distribution of the train set, making it not robust to different graphs. This problem is more severe if the number of nodes and the time magnitude increase, such as in the data set 6x6L2D and 15x15L2D, where both MSE loss and R2-score for GIN-0 have poor performance.

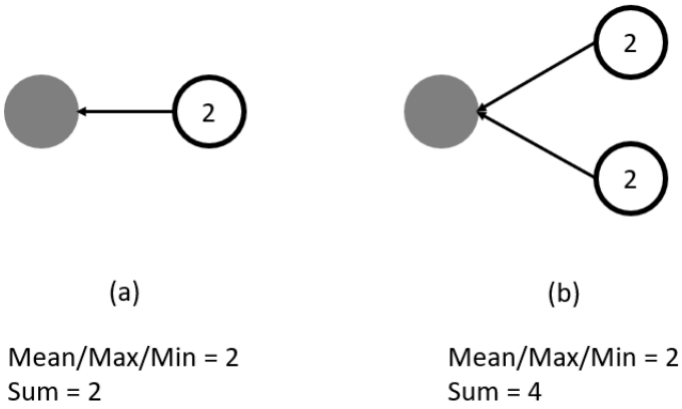

Figure 25: Illustration of the discriminative ability of SUM aggregation, showing its expressive power in differentiating graph structures where other aggregation methods like MEAN, MAX, or MIN may fail.

Table 12: This table presents an example comparison of the distribution of SUM aggregation messages in the training and test sets, illustrating numerical instability in the SUM aggregation method due to variations in the distribution of operation times.

Table 13

Dataset	**Train Set**	**Test Set**
Number of nodes	38	38
Distribution of operation time	[2, 8]	[1, 11]
Distribution of SUM aggregation message	[76, 304]	[38, 418]

3.6.4 Results in Inland Waterway Transport context

Our study focused on optimising port call shipping operations, we leveraged the JSSP framework to enhance scheduling efficiency and resource allocation. By applying this approach to real-world scenarios, we utilised a comprehensive dataset featuring six ships, six different types of tasks, and six distinct resource constraints. This diverse and complex dataset allowed us to test the capabilities of the JSSP in a port environment. Through analysis, we observed that the JSSP model accurately predicted optimal scheduling sequences, effectively minimising time and resource conflicts. The results demonstrated that the JSSP's predictions aligned closely with the time required to solve complex scheduling challenges, showcasing its capacity to streamline port operations and contribute to the Port of Duisburg's position as a global leader in maritime trade efficiency.

We present a visual representation in Figure 26 and Figure 27 to validate our approach further. The regression plot exhibits a tight clustering of data points around the diagonal line, illustrating the close alignment between predicted and actual values. Moreover, the associated error term diagram highlights a pronounced concentration of error terms near zero, underscoring the precision and reliability of our JSSP-based predictions in line with real-world observations. This striking convergence between anticipated and factual outcomes enhances the credibility of our approach, reinforcing its suitability for tackling intricate port scheduling complexities.

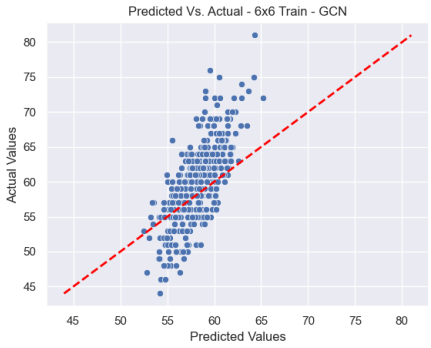

Figure 26: Regression analysis of JSSP predictions for port call shipping operations, exhibiting a strong correlation between the predicted and actual scheduling times, indicative of an effective optimization process.

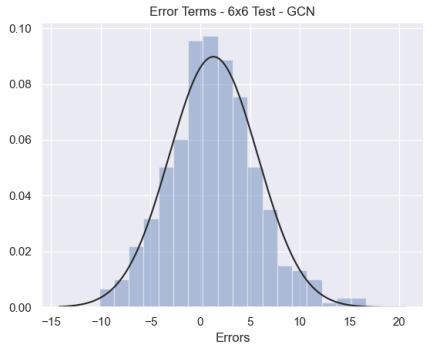

Figure 27: Error distribution for JSSP predictions, showing a high concentration of errors around zero, suggesting a high level of accuracy in the scheduling model's performance.

3.7 Conclusion

The Job Shop Scheduling Problem (JSSP) is widely recognised as a fundamental issue in the field of combinatorial optimisation, especially concerning production planning (Xiong et al., 2022). The key is finding a schedule that reduces the total manufacturing time or makespan. Although the Priority Dispatching Rule (PDR) is the predominant solver for JSSP (Haupt, 1989), its heuristic nature demands intricate domain-specific knowledge. Moreover, it tends to falter when alterations in the problem's environment arise. This underscores the need for a more adaptable solution (Balas, 1969).

In the fascinating domain of JSSP, its representation via a disjunctive graph stands out. Notably, while there have been ventures into incorporating the disjunctive graph within Deep Reinforcement Learning, the act of representing the disjunctive graph itself remained untouched. Enter the era of deep learning, where Graph Neural Networks (GNN) are emerging as a powerful tool explicitly designed to learn from graph data through their distinctive message-passing mechanism. This study investigated an important question: Could Graph Neural Networks accurately capture and represent the inherent characteristics of disjunctive graphs?

Throughout this research exploration, various GNN architectures were meticulously evaluated for their capacity to capture the essence of the disjunctive graph structure. The experiments conducted in this thesis unveiled significant insights. Primarily, normalising node features in the disjunctive graph stands out as a critical factor for boosting model efficiency. It became evident that when node features were normalised, the graph-convolutional network's understanding of disjunctive graphs improved. This enhancement was consistent across datasets, with a remarkable increase of up to 89% observed for the 6x6 dataset.

Another aspect of the research explored the significance of the start (S) and terminate (T) nodes in the disjunctive graph. These nodes, devoid of any feature values, were postulated as potentially redundant. However, empirical evidence showed a mixed bag of results. While certain datasets witnessed performance boosts up to 35.32% when these nodes were included, others remained largely unaffected. This led to the hypothesis that the presence of these nodes morphs the disjunctive graph more toward a flow network, which GNNs seem to resonate better with, rather than likening it to a molecule graph.

When the performance of various GNN models was compared against each other, GCN emerged as a formidable contender compared to MLP and Graph Isomorphism Network (GIN), often outperforming its

counterparts. The models' performance on different data sets, from standard to L2D data sets, exhibited varied results, highlighting the intricate dynamics of the JSSP and the complexity involved in representing it using GNNs.

Utilising the JSSP framework, our study successfully optimised port call shipping operations within the Port of Duisburg. Using a multifaceted dataset, the JSSP model accurately predicted optimal scheduling sequences, showcasing its efficacy in streamlining port operations. Visual data further corroborated the model's precision and reliability, positioning the JSSP as a vital tool for addressing port scheduling complexities.

In response to the research subquestion posed, GNNs, particularly the GCN architecture, have shown promising potential for developing a compact representation of the JSSP through the disjunctive graph structure. Key findings revealed the paramount importance of normalising node features in disjunctive graphs, with GNNs seeing a performance increase of up to 89% on certain data sets when nodes were normalised. The role of the start (S) and end (T) nodes produced mixed results, with some data sets showing enhancements up to 35% when these nodes were included. Regarding GNN model comparison, GCN notably outperformed counterparts like MLP and GIN. Applying these insights, the JSSP framework effectively streamlined port operations at the Port of Duisburg, demonstrating its practical viability. Furthermore, while GNNs show potential, the varied performance across different data sets underscores the need for further fine-tuning and exploration, especially in contexts like inland waterways.

4 Resource Constrained Project Scheduling Problem

4.1 Introduction

The challenge of optimally scheduling projects in the face of limited resources is a cornerstone of project management and operations research, known as the Resource-Constrained Project Scheduling Problem (RCPSP). This complex task involves assigning limited resources to various project activities, adhering to specific constraints, and achieving set objectives, thus reflecting its profound significance in numerous industries ranging from engineering to software development (Brucker et al., 1999).

The RCPSP is centred on organising activities, each with a fixed duration, that must be completed without interruption, in a sequence dictated by precedence relationships—a task represented by a Directed Acyclic Graph (DAG) with activities as nodes and special nodes for project start (source) and end (sink). It involves the strategic use of limited renewable resources like labour or machinery without exceeding their maximum capacity while aiming to minimise the project's total duration, known as the makespan. The complexity of allocating these finite resources efficiently to optimise schedules has led to significant research and the development of advanced algorithms to address the challenges of RCPSP (Nightingale & Demirović, 1999).

The scope of RCPSP extends beyond individual project scheduling to encapsulate a wide range of scheduling problems, standing as a fundamental issue in the domain. It serves as the umbrella for special cases like job-shop, open-shop, and flow-shop scheduling problems, reinforcing its versatility in fields such as manufacturing and construction (Cai et al., 2022; Sprecher et al., 1997).

Recent advancements have broadened the RCPSP paradigm, incorporating diverse models, varying activity conditions, different types of resources, and novel objective functions, addressing both deterministic and stochastic aspects of the problem (Naderi et al., 2022). While simpler scheduling issues without resource constraints can be managed by methods like the Critical-Path Method (CPM), introducing resource limitations renders the problem NP-hard. An extension of RCPSP is the multi-mode RCPSP (MRCPSP), which presents various activity modes, offering different resource-time trade-offs and demanding strategic decisions regarding activity scheduling and resource allocation (Zhu et al., 2006).

The traditional RCPSP model assumes a rigid finish-to-start precedence and a consistent resource demand during the activity's duration. However, real-world scenarios often do not conform to these assumptions, leading to the development of the Generalised Resource-Constrained Project Scheduling Problem (GRCPSP). This more nuanced model accommodates different types of precedence relationships and is tackled by precise and heuristic methods (Bartusch et al., 1988; Herroelen et al., 1998; Schutt et al., 2013). Moreover, the flexible resource profiles version (FRCPSP) further refines the model by allowing for variable resource usage per activity, with heuristic approaches showing promising results, alongside the existence of exact mixed-integer programming (MIP) solutions (Fündeling & Trautmann, 2010; Naber, 2017; Naber & Kolisch, 2014; Tritschler et al., 2017). Practically, projects are often represented using Activity-on-Node (AoN) networks, which illustrate activities as nodes and their precedence relationships as arrows. This visualisation underscores the logical sequence of tasks (Koulinas et al., 2014). An example is provided in Figure 28. The RCPSP integrates these sequences with resource constraints to optimise and reduce the overall project timelines.

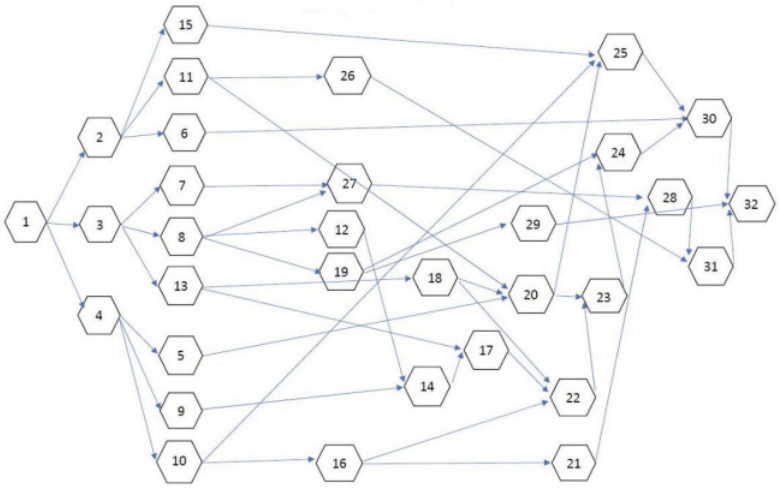

Figure 28: Example of an Activity-on-Node (AoN) network illustrates a project with 32 activities, where activities 1 and 32 are dummy tasks with no duration.

4.2 RCPSP formulation

4.2.1 Definition

The Resource-Constrained Project Scheduling Problem is a combinatorial optimisation problem that aims to schedule activities in a project optimally while considering resource constraints and precedence relations among tasks. This problem, as characterised by Artigues, 2013, is defined by the tuple (V, p, E, R, B, b):

- Activities: A project consists of a set of activities, $V = \{A_0, ..., A_{n+1}\}$. Here, A_0 indicates the start and A_{n+1} indicates the end of the schedule. Including these dummy activities, the project has $A = \{A_1, ..., A_n\}$ tasks.
- Durations: Each activity has a duration, represented by the vector p in N_{n+2}. Due to dummy activities, $p_0 = p_{n+1} = 0$.
- Precedence Relations: Activities follow certain orderings or precedences. If $(A_i,\ A_j)$ is in set E, then activity A_i must be completed before A_j starts. This is visualised by an activity-on-node graph $G(V, E)$. Importantly, A_0 precedes all activities and A_{n+1} follows all.
- Resources: The project relies on renewable resources, represented by set $R = \{R_1, ..., R_q\}$.
- Resource Availabilities: Each resource has a maximum availability, given by vector B in N_q. If a resource, R_k, has an availability of 1, it is unary. Otherwise, it is cumulative.
- Resource Demands: Activities require resources during execution, represented by matrix b. Specifically, b_{ik} tells how much resource R_k is used by activity A_i per time period.

A schedule, S, defines the start times of activities. If S_i is the start time for A_i, then its completion is $C_i = S_i + p_i$. The project starts at $S_0 = 0$. To be valid, a schedule must meet two sets of constraints:

- Precedence Constraints: Activities must respect their ordering:

$$S_j - S_i \geq p_i \in (A_i, A_j) \in E \tag{32}$$

- Resource Constraints: At any time, total resource demand cannot exceed availability:

$$\sum_{A_i \in A_t} b_{ik} \leq B_k \forall R_k \in R, \forall t \geq 0 \tag{33}$$

The goal of the RCPSP is to minimise the makespan, S_{n+1}, which is the start time of the end activity, subject to the constraints 32 and 33. Notably, as activity durations are integers, only integer schedules need to be considered to find an optimal solution.

4.2.2 RCPSP example

An illustration given by Artigues, 2013 is shown below. Table 14 presents an RCPSP instance with ten actual activities ($n = 10$) and two resources ($\mid R \mid = 2$). The availabilities for these resources are $B_1 = 7$ and $B_2 = 4$, respectively.

Table 14: Table detailing an RCPSP instance with ten actual activities and resource requirements, illustrating the duration and resource consumption for each activity.

A_i	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}
p_i	0	6	1	1	2	3	5	6	3	2	4	0
b_{i1}	0	1	0	1	0	1	1	0	2	2	1	0
b_{i2}	0	2	1	3	2	1	2	3	1	1	1	0

The activity-on-node graph in Figure 29 illustrates the precedence constraints connecting the activities $A_i \in V$. Additionally, a schedule with the shortest makespan, $S^*_{n+1} = 12$, is depicted as a two-dimensional Gantt chart in Figure 30. In this chart, the x-axis signifies time, while the y-axis indicates resource utilisation.

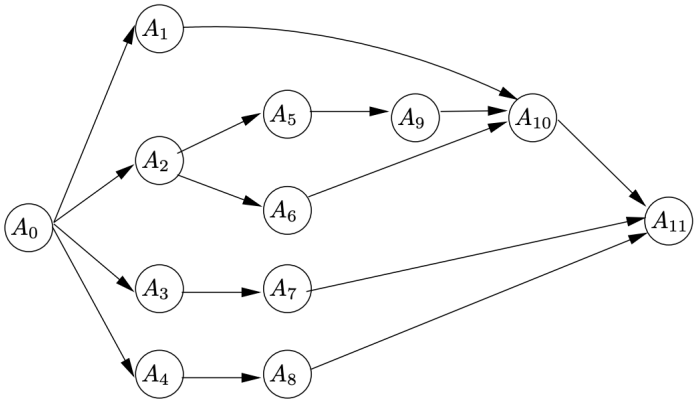

Figure 29: Activity-on-node graph showing the precedence relationships between activities in an RCPSP instance, guiding the project's scheduling constraints.

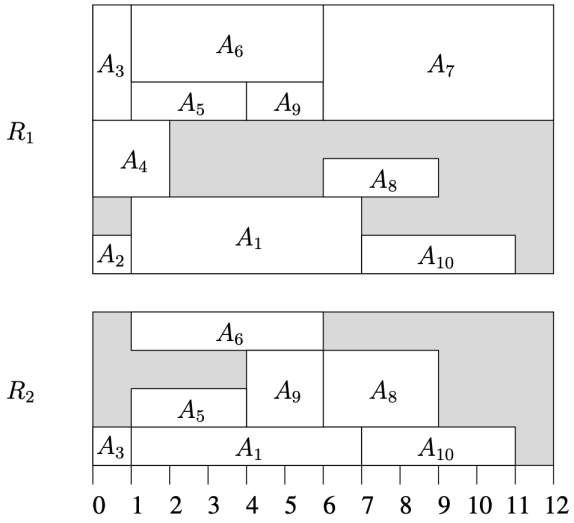

Figure 30: Optimal schedule Gantt chart for an RCPSP instance, demonstrating the shortest makespan and resource allocation over time.

4.2.3 Mathematical problem formulation

In our study, the RCPSP involves scheduling multiple activities across different periods, each with a single mode. Resources are categorised into renewable and non-renewable types. Decisions include determining when to initiate an activity and the amount of renewable resources required, considering initial availability. An activity's duration is also defined. The primary objective is to minimise scheduling lags while making decisions optimally.

The RCPSP is denoted by the set of activities $A = 1, ..., i$, subjected to two primary constraints: resource constraints and precedence relations. The mathematical representation is as follows:

$$\text{minimise } f_n \tag{34}$$

$$\text{s.t.} \quad f_i \leq f_j - d_j, \qquad \forall (A_i, A_j) \in \text{Pred}, \tag{35}$$

$$\sum_{i \in P_t} u_{irt} \leq U_r, \qquad \forall t \in \{1, \ldots, f_n\}, \forall r \in R. \tag{36}$$

$$f_1 = 0,\ d_1 = 0,\ d_n = 0, \tag{37}$$

Equation 34 represents the goal of optimising project duration, accounting for primary constraints. Meanwhile, Equation 35 guarantees adherence to precedence constraints, ensuring that activity i begins only after its direct predecessors are finished. Activities consume renewable resources, depicted by u_{irt} units per time. Equation 36 ensures that resource consumption remains within available limits during project execution(Koulinas et al., 2014). Equation 37 clarifies that activities 1 and n are dummies, essentially serving as project milestones. Table 15 details the components used in standard RCPSP.

Table 15: Elementary definitions in the context of standard RCPSP.

Element	Description
A	The collection of project tasks with duration d_i, numbered as $i = 1, 2, 3, ..., n$. Activities 1 and n serve as placeholders or benchmarks.
R	Represents renewable resources needed for the project, defined as $R = 1, ..., r$.
U_r	Quantity of the renewable resource r that is available for use.
f_i	Completion moment of the ith activity.
f_j	Completion moment of the j-th activity, which directly follows the i-th activity.
Pred	Denotes ordered pairs (A_i, A_j), indicating A_j directly succeeds A_i.
u_{irt}	Resource consumption: the quantity of renewable resource r utilised by activity i at time t.

4.3 Characteristics of RCPSPs

Over the years, numerous classifications have emerged to delineate the variations and nuances of RCPSPs, taking into account the different characteristics of resources and the diverse nature and requirements of the activities involved.

Resource constraint characteristics In project scheduling, activities require various types of resources for their execution. The primary focus is on renewable resources, which are assumed to maintain full capacity throughout each period and are consumed in fixed quantities during activity execution. The specific consumption patterns of these resources depend on the nature of the activities. Within the realm of renewable resources, there is a distinction between those that are readily available and those that are scarce and expensive, often requiring external sourcing for short project durations.

Another resource category is non-renewable resources, which are finite and remain constant throughout the entire project. These include resources such as raw materials and the overall project budget.

Doubly constrained resources combine characteristics of both renewable and non-renewable resources, with limitations imposed both per period and for the project's entirety. Money and energy serve as examples. While doubly constrained resources can be represented using renewable and non-renewable resource models, their unique attributes warrant consideration in the project scheduling study of Hartmann and Briskorn, 2008. Overall, resource allocation plays a crucial role in project management and scheduling, with various types of resources impacting the planning and execution of activities.

Characteristics of activities Over time, the understanding of RCPSPs has evolved, with practical implications. One foundational assumption in the basic RCPSP is the uninterrupted progression of project activities. However, real-world situations often require temporary activity halts. Activities can be allowed to pause and resume, either without added expenses or at designated times, echoing the challenges of real-world scheduling.

Another traditional RCPSP assumption is that activities follow a single method with predefined timeframes and resource consumption. The concept of multi-mode activities challenges this by proposing multiple execution techniques, each with distinct time and resource requirements. Once chosen, an activity's execution mode remains consistent until completion.

RCPSPs typically assume constant resource demand for activities throughout their duration. However, introducing variable resource demand expands this assumption, allowing resource requirements to vary across activity periods and enhancing flexibility.

Reducing project timelines is essential. Techniques such as activity crashing involve reducing critical activity durations at higher costs. Additionally, there are considerations like prohibited intervals when activities cannot proceed due to inherent characteristics, and significant transitions between tasks, such as shifting resources, are necessary. These evolving concepts in RCPSPs reflect the adaptability of scheduling models to real-world complexities and the need to address practical challenges in project management.

4.4 Literature

4.4.1 RCPSP Problems in general

The RCPSP problem is notable for both its relevance in real-world scenarios and the computational intricacies it poses. Over time, the focus of research has been predominantly on forging superior solution techniques, with Zhu et al. (2006) categorising existing strategies for minimising RCPSP makespan into: priority rule-based sequencing heuristics, metaheuristics, and explicit branch and bound methods. Initial research based on standard integer linear programming (ILP) formulations encountered barriers due to a multitude of binary variables in practical-sized problems.

Two primary approaches dominate the stochastic RCPSP landscape: robust scheduling and stochastic scheduling. Robust scheduling preempts uncertainty by embedding time buffers into the schedule. On the other hand, reactive scheduling, which has drawn attention from Goldratt (1997), Herroelen and Leus (2004), and Lambrechts et al. (2008), seeks adjustments in the face of emerging information. Some works, like Bai et al. (2022), Brcic et al. (2019), and Davari and Demeulemeester (2019), synthesise both methods. Stochastic scheduling itself is anchored on a scheduling policy that directs which activity begins at every decision point Igelmund and Radermacher (2010). This classification further bifurcates into open-loop or closed-loop policies H. Li and Womer (2015). While open-loop policies simultaneously lay out the entire project's schedule, their closed-loop counterparts address only the immediate scenario. The popularity of priority rules (PR) in open-loop policies stems from their simplicity and efficiency. Conversely, closed-loop policies, as outlined by Dreyfus and Law (2014), are dynamic, modifying the plan based on real-time data.

Projects without resource constraints can be adeptly handled by the Critical-Path Method (CPM) (Zhu et al., 2006). However, introducing even a single resource pivots the problem to the NP-hard category (Zhu et al., 2006). Meanwhile, Zhu et al. (2006) advocates for a branch-and-cut approach for the multi-mode RCPSP (MRCPSP) and emphasises leveraging precedence constraints.

Meta-heuristics have emerged as potential RCPSP solutions, with Golab et al. (2022) listing related articles. Some innovative meta-heuristic approaches include a self-adaptive and hybrid genetic algorithms. The former uses activity list representation with two decoding strategies, while the latter revolutionises the traditional genetic algorithm structure, incorporating several novel elements.

Challenging the RCPSP's classical assumptions, Bold et al. (2021) introduces two innovative extensions: the generalised RCPSP (GRCPSP) and the flexible resource profile RCPSP (FRCPSP). By integrating both, they present the GFRCPSP, an NP-hard challenge. They propose a mixed-integer programming (MIP) framework combined with a non-greedy genetic algorithm to tackle this.

The RCPSP domain, significant for its real-world applicability, has witnessed several groundbreaking milestones. Initially, the focus was on handling uncertainties, with Kolisch and Hartmann (1999) pioneering the Fuzzy Critical Chain Method. By leveraging fuzzy sets, this approach transformed the conventional critical chain, adapting it to ever-present uncertainties in project dynamics. Following this, the search for optimisation techniques led Pan and Jang (2008) to refine the traditional tabu search. This Enhanced Tabu Search elevated solution quality while reducing computational time.

In metaheuristics, J. Chen and Askin (2009) made a significant leap and pioneered the combination of Genetic Algorithms with Simulated Annealing. By capitalising on both strengths, this fusion ensured a holistic exploration of the solution space. Subsequently, Kyriakidis et al. (2012) introduced Mixed-Integer Linear Programming, broadening the scope to accommodate both single- and multi-state projects. Recognising the unpredictable nature of real-world scenarios, Bruni and Beraldi (2011) integrated random variables to represent uncertain durations, thereby adding a Stochastic Lens to project management.

The next wave of innovations emphasised advanced algorithms. For instance, Bouleimen and Lecocq (2003) optimised the simulated annealing process, infusing it with dual loop techniques for superior convergence. Parallelly, Chakrabortty et al. (2020) tapped into the Variable Neighbourhood Search Metaheuristic, while Tao and Dong (2018) combined the AND-OR Network with Dual-Objective Genetic Methods, both enriching the repertoire of RCPSP solutions.

In more recent advancements, Xie et al. (2021) addressed activity cost uncertainties, crafting an optimisation model to maintain cost equilibrium. As technology progressed, so did the solutions. Saad et al. (2021) delved into quantum computing, unveiling a Quantum-Infused Genetic Algorithm, tapping into quantum computation's unmatched capabilities. Lastly, introducing a mathematical nuance, Bulavchuk and Semenova (2021) based their genetic algorithm on algebraic methods, merging sophistication with elegance.

In summary, the landscape of project scheduling is dotted with a plethora of models and methods, each uniquely contributing to the expansive knowledge base. These scholarly endeavours underscore the essence and evolution of project scheduling, propelling it forward in our quest for optimal solutions. A summary of this literature subsection can be found in Table 16.

Table 16: Literature overview of general RCPSP.

Article	Summary
Zhu et al., 2006	Presents a branch and cut procedure for solving the multi-mode RCPSP directly using the ILP model.
Golab et al., 2022	Discusses meta-heuristics to solve RCPSP, including self-adaptive genetic algorithms and hybrid genetic algorithms.
Bold et al., 2021	Introduces extensions to RCPSP, including the Generalised Resource-Constrained Project Scheduling Problem (GRCPSP) and Resource-Constrained Project Scheduling Problem with flexible resource profiles (FRCPSP).
Kolisch and Hartmann, 1999	Proposes a fuzzy critical chain method for resource-constrained project scheduling under uncertainty.
Pan and Jang, 2008	utilises an improved tabu search method for Resource-Constrained Project Scheduling.
J. Chen and Askin, 2009	Combines genetic algorithms and simulated annealing to solve multi-project scheduling problems.
Kyriakidis et al., 2012	Uses mixed-integer linear programming models to formulate single-state and multistate project scheduling problems.
Bruni and Beraldi, 2011	Investigates Resource-Constrained Project Scheduling with uncertain activity durations.
Bouleimen and Lecocq, 2003	Provides an efficient simulated annealing algorithm for a Resource-Constrained Project Scheduling Problem with different modes.
Chakrabortty et al., 2020	utilises variable neighbourhood search metaheuristic for multi-mode resource-constrained project scheduling.
Tao and Dong, 2018	Presents resource-constrained project scheduling with alternative project structures, using genetic methods and tabu search.
Xie et al., 2021	optimises project scheduling under uncertainty of activity cost, employing a hybrid approach and genetic algorithm.
Saad et al., 2021	Solves the Resource-Constrained Project Scheduling Problem using a quantum-based genetic algorithm.
Bulavchuk and Semenova, 2021	Uses a genetic algorithm based on algebraic methods for RCPSP, considering resource constraints and net present value.

4.4.2 RCPSP Problems solved with Deep-learning techniques

The intersection of deep learning and combinatorial optimisation for the Resource-Constrained Project Scheduling Problem presents a promising avenue for achieving better results in scheduling. Deep learning techniques, particularly Graph Neural Networks (GNNs), have been found to be efficient in capturing dependencies and constraints intrinsic to scheduling problems.

Deep learning has significantly influenced combinatorial optimisation (CO) problems (Bengio et al., 2021). Depending on the training method, solutions based on deep learning can be supervised or reinforcement learning (RL) methods. Although supervised learning offers faster computation, it often falls short of achieving the results of the original algorithm, especially for large-scale CO problems. RL methods, on the other hand, do not depend on precalculated solutions, focussing instead on learning

strategies. The downside is the demanding training time and the complexities in environmental setup Shou, 2005.

GNNs have become notably relevant in this context. Essentially, GNNs function on graph-structured data, capturing information from nodes (entities) and edges (relationships) to extract useful features (embeddings) for subsequent tasks. Given that graphs can represent scheduling problems, several researchers have begun to explore the applications of GNNs to such challenges.

Shou, 2005 pioneered a neurogenetic approach that combined genetic algorithms (GA) and neural networks (NN). The model leverages GA for global search and NN for local search, intertwining the two to extract optimal solutions. The emphasis here was on using Artificial Neural Networks (ANN) to optimise project durations.

Teichteil-Königsbuch et al., 2023 introduced an approach that utilises GNNs for the RCPSP. The research formulates scheduling as a combinatorial optimisation task by representing activities as nodes, precedence constraints as edges, and encoding resource availability within node features. The GNN model is trained using the Proximal Policy optimisation algorithm, and the resulting GNN-based scheduler was found to be superior in both solution quality and computational efficiency compared to traditional methods.

Highlighting the benefits of deep reinforcement learning, Zhao et al., 2022 combines neural networks with reinforcement learning to enhance the solution quality and computational efficiency for RCPSPs. Meanwhile, Adamu and Aromolaran, 2018 utilised machine learning to dynamically choose the most suitable reactive scheduling policy among the predefined ones. However, its application was limited to small-scale projects, and the trained model lacked generalisation for unseen data.

Addressing stochastic versions of the RCPSP, Cai et al., 2022 devised a framework integrating GNN and RL. The process treats scheduling as sequential decision-making, employing a GNN to extract problem features and convert them into probability distributions to drive scheduling decisions. This model was trained using proximal policy optimisation and showcased competitive performance.

In the realm of estimating project times for RCPSP, Ozkan and Gülçiçek, 2015 utilised an artificial neural network (ANN) and achieved a commendable correlation coefficient of 0.70. This result underscores the effectiveness of ANN in predicting project durations under resource constraints.

Providing a holistic overview, Abdolshah, 2014 emphasised the importance of RCPSP in operational research and optimisation. This comprehensive review covered more than 200 articles, offering a systematic presentation of various methods and models described in existing literature, acting as a reference point for practitioners.

Lastly, Ranjbar et al., 2012 approached the RCPSP intending to minimise total weighted resource tardiness penalty costs. Considering renewable resources with specific ready and due dates, the study utilised a branch-and-bound algorithm, employing a graph-based branching scheme. Through this, the research proposed different rules to optimise the RCPSP, focusing on resource constraints.

In conclusion, the realm of RCPSP has witnessed a shift toward integrating deep learning techniques, primarily GNNs, offering enhanced efficiency and solution quality. Building on this foundation, we delve deeper into the intricacies of RCPSP by continuing with the approach presented by Teichteil-Königsbuch et al., 2023. Converging traditional methods with state-of-the-art neural networks provides a forward-looking approach to solving complex scheduling problems. A summary of the literature regarding RCPSP problems solved by deep learning methods can be found in Table 17.

Table 17: Literature overview of RCPSP solved by DL techniques.

Article	Summary
Shou, 2005	Develops a neurogenetic approach for solving RCPSP by combining genetic algorithms (GA) and neural network (NN) approaches.
Teichteil-Königsbuch et al., 2023	Introduces an approach to address the Resource-Constrained Project Scheduling problem using Graph Neural Networks and reinforcement learning techniques.
Zhao et al., 2022	Proposes a deep reinforcement learning approach for the Resource-Constrained Project Scheduling Problem, combining deep neural networks and reinforcement learning algorithms.
Adamu and Aromolaran, 2018	Uses machine learning and reinforcement learning to choose scheduling policies for the RCPSP dynamically, but primarily applies to small projects.
Cai et al., 2022	Introduces a framework for addressing the Stochastic Resource-Constrained Project Scheduling Problem (SRCPSP) using a GNN and RL.
Ozkan and Gülçiçek, 2015	Discusses using an artificial neural network (ANN) for estimating project times in RCPSP, achieving a correlation coefficient of 0.70.
Abdolshah, 2014	Reviews and compiles different methods and approaches used to solve Resource-Constrained Project Scheduling Problems.
Ranjbar et al., 2012	Introduces an optimal solution procedure to minimise total weighted resource tardiness penalty costs in RCPSP with constrained renewable resources.

4.4.3 RCPSP problems in Inland Waterway Transport context

Waterway Ship Scheduling Problem (WSSP) presents unique challenges due to the characteristics of waterway transportation, such as limited resources and the availability of various modes of operation for ships. Hill et al., 2019 focuses on the multi-modal aspect of ship scheduling. Ships in waterways can operate under different modes, each with distinct resource needs. The study reformulates the RCPSP, tailoring it for the WSSP to incorporate resource constraints, precedence relations, and ship operating modes. This reformulation enables more precise scheduling, considering the intricacies of each mode and other factors like resource constraints and activity interdependencies.

On the other hand, Agussurja et al., 2018 focuses on preventing congestion in narrow water areas, using the Singapore Straits as a case study. Given the increased maritime traffic, ensuring efficient vessel scheduling is vital for timely arrivals and safety. The research contributions include: 1) presenting a maritime traffic management problem specific to Singapore waters; 2) modelling this challenge as a variant of RCPSP, with mixed-integer and constraint programming (MIP/CP) formulations; 3) introducing a Combinatorial Benders (CB) approach for enhanced scalability, supplemented by symmetry-breaking constraints and optimality cuts; 4) designing a maritime traffic simulator for Singapore Straits. The results show a reduced traffic density with minimal delays, emphasising the efficiency of the proposed methods.

Integrating the RCPSP-WSSP framework by Hill et al., 2019 with the GNN approach of Teichteil-Königsbuch et al., 2023, our study presents a hybrid solution for the WSSP that leverages the benefits of both methodologies. This combined strategy aims to capture the complexity of ship scheduling while utilising the predictive strength of GNNs, offering a comprehensive and efficient approach to managing waterway transportation challenges. A summary of the RCPSP problems in the waterway context

discussed in this subsection is presented in Table 18.

Table 18: Literature overview of RCPSP in waterway context.

Article	**Summary**
Hill et al., 2019	Reformulates the Waterway Ship Scheduling Problem (WSSP) as a Resource-Constrained Project Scheduling Problem, considering the multi-mode nature of ship scheduling in waterways.
Agussurja et al., 2018	Addresses congestion and hotspot prevention in waterways, specifically in the Singapore Straits, using mixed-integer and constraint programming formulations and a combinatorial Benders approach, leading to reduced traffic density and minimal delays.

4.5 Data description

A standardised data set is crucial for a robust evaluation of different heuristic outcomes in the RCPSP. The PSPLIB repository, introduced by Kolisch and Hartmann, 1999, is a comprehensive benchmark for various RCPSP instances, including multi-mode and single-mode formulations. The multi-mode instances offer tasks the flexibility to choose from various resources or modes, while the single-mode restricts tasks to one specific resource.

The PSPLIB comprises diverse instance types and sizes and is a cornerstone for testing both exact and heuristic methods in RCPSP research. It offers four distinct sizes in the single-mode RCPSP: *j30, j60, j90, and j120*, representing 30, 60, 90, and 120 activities respectively. The respective instance counts for these sizes are 480, 480, 480, and 600. Each instance is crafted with varying parameters, such as network complexity and resource intensity.

For the study's scope, the model is trained on a subset from the PSPLIB, retaining other instances for validation and testing. In total, the PSPLIB provides 2040 varied instances. Of these, 1632 instances (or 80%) spanning different sizes and structures are designated for training, ensuring a comprehensive assessment of the model's capability.

Data used in Inland Waterway Transport context The concepts of RCPSP can find application within the realm of inland waterway transport. To illustrate this, we provide an example of employing RCPSP principles within the context of inland waterways. Consider a real-world scenario where the goal is to optimise the unloading of cargo ships at an inland port situated within a network of interconnected waterways. This scenario mirrors real-world complexities, with each port facing limitations in terms of resources and specific time slots allocated for unloading activities. The overarching goal here is to execute cargo ship unloading operations efficiently while strictly adhering to resource constraints and striving to minimise the overall processing time.

In this intricate web of logistic operations, we encounter several tasks and constraints. First, we have the cargo ships with a predefined unloading time. These times represent the duration it takes for a ship to offload its cargo completely. Then, there are ports, which serve as pivotal hubs in this supply chain network. Each port is equipped with a finite number of berths and unloading equipment, which is essential for efficiently handling cargo. Importantly, ports operate within specific time windows, dictating when cargo unloading activities occur. This temporal aspect adds an additional layer of complexity to the scheduling puzzle.

Let us consider a concrete example to illustrate these concepts: Consider a scenario with multiple cargo ships labelled as Ship A, Ship B, and Ship C. Ship A requires six hours for unloading, Ship B takes eight hours, and Ship C completes unloading in five hours. Simultaneously, we have multiple ports in our network: Port X, Port Y, and Port Z. Port X offers two berths and is available for unloading activities between 8:00 in the morning and 4:00 in the afternoon. Port Y, with three berths, operates from 9:00 a.m. to 6:00 p.m., and Port Z, with two berths, is accessible between 10:00 am and 8:00 p.m.

To effectively address the challenge of unloading cargo ships, we employ a structured solution approach that encompasses task sequencing, resource allocation, berth balancing, and an overarching objective of minimising unloading time and optimising ship arrivals. Task sequencing involves determining the order of ship unloading based on factors like unloading times and resource availability. Resource allocation assigns ships to available berths, respecting berth availability and specified time windows. Berth balancing optimises this allocation for efficient resource usage. The ultimate goal is to reduce unloading time and improve scheduling, ensuring a smooth flow of activities in the logistics network and preventing congestion at ports.

As a result of this comprehensive approach, a potential solution emerges, exemplified by the following schedule: At Port X, Ship A conducts unloading from 8:00 AM to 2:00 PM, followed by Ship C from 2:00 PM to 7:00 PM. Meanwhile, at Port Y, Ship B completes the unloading between 9:00 AM and 5:00 PM. Lastly, at Port Z, Ship C arrives at 10:00 AM and concludes unloading by 3:00 PM, after which Ship A takes over from 3:00 PM to 9:00 PM. This optimised scheduling minimises unloading times for each ship, efficiently utilises resources, and ensures a smooth logistical flow while preventing port congestion. It effectively mitigates congestion at the ports and guarantees the smooth flow of activities within this intricate logistics network. Figure 31 provides a detailed visualisation of the optimised unloading schedule.

However, as with any real-world scenario, this endeavour has challenges. Dynamic conditions, such as changing weather patterns and traffic fluctuations, may require real-time adjustments to established schedules. Additionally, the intricate balance between cargo ship unloading, berth availability, and time windows introduces complex constraints that require careful consideration. Furthermore, the inherent uncertainty in predicting accurate unloading times and cargo ship arrivals remains an ongoing challenge, which calls for innovative and adaptive solutions to address these uncertainties effectively.

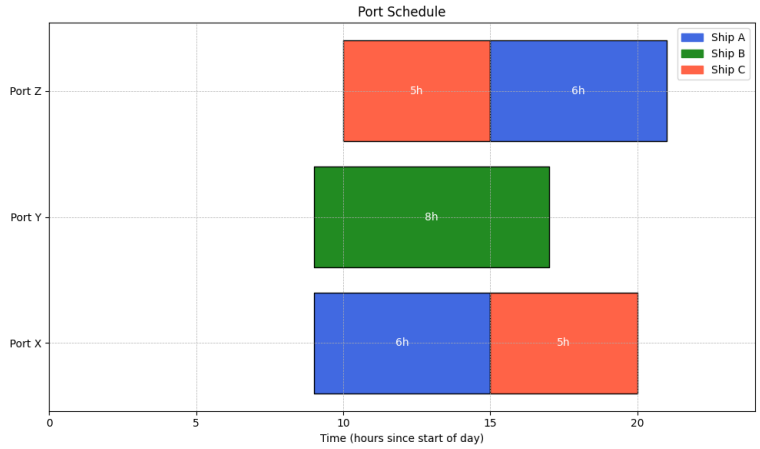

Figure 31: Gantt chart depicting the optimised example schedule for unloading cargo ships at Ports X, Y, and Z. The colours represent different ships, and the horizontal bars indicate the unloading time slots allocated to each ship at the respective ports.

4.6 Methodology

The methodology section, predominantly inspired by the work of Teichteil-Königsbuch et al., 2023, delves deep into the unique approach of addressing the RCPSP problem with the help of GNN.

The approach here is to harness the potential of GNN to mimic and project solutions analogous to an exact CP solver. Interestingly, by using labels from the CP-sat solver's solutions, a supervised GNN training approach consistently yielded better results, leveraging the relationship between feasible solutions and the structure of CP-sat models. GNNs, innately built to cater to data best depicted as a graph, have two standout features. Firstly, the model's output is unfazed by the sequence of nodes. Secondly, a single GNN model seamlessly fits different RCPSP instances regardless of size or structure. Employing GNNs in addressing RCPSPs poses dual challenges. Firstly, the GNN is tailored to recognise the pivotal relationships embedded within RCPSP problems. Secondly, there's the task of handling cases where schedules generated by the GNN might not adhere to feasibility. For the former, the RCPSP problem was rewritten to be GNN-compatible, with specifics discussed in the following subsection. For the latter, strategies were formulated to refine the GNN's predictions, steering them closer to feasibility. These nuances are elaborated on in the interference section.

Outlined in Figure 32 are the principal steps and components of the training and inference processes for the RCPSP problem. The training process constructs batches of GNNs for specific RCPSP problems to collectively refine prediction capabilities through back-propagation. After training, GNN constraint violations are rectified by sequencing tasks based on start dates and utilising the Serial SGS procedure to achieve feasible schedules.

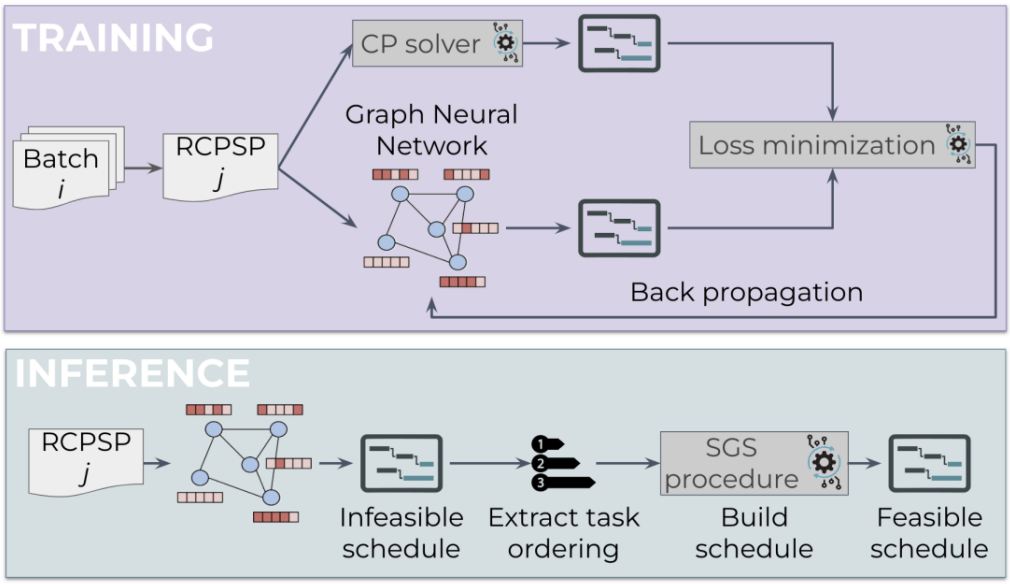

Figure 32: This diagram illustrates the training and inference processes for solving RCPSP problems using a Graph Neural Network. During training, the GNN learns from batches of RCPSP instances with guidance from a CP solver for loss minimization through back-propagation. In the inference phase, the GNN's output is refined through a Serial SGS procedure to extract task ordering from an initially infeasible schedule, resulting in a feasible schedule that aligns with resource constraints and task dependencies (Teichteil-Königsbuch et al., 2023).

4.6.1 Training

GNN Representation of RCPSPs To portray an RCPSP instance, we utilise a GNN graph defined as $G = (T, R, E, \mathbf{V}, \mathbf{E})$. Tasks ($T$) and resources ($R$) are distinct node types in this graph. The graph has three types of edges: precedence edges (E_P), resource demand edges (E_R), and parallel reverse links (E_{rev}). These edges serve to capture relationships, such as precedence constraints and resource demands, between tasks and resources. Edge features, represented by $\mathbf{e_{ij}} \in \mathbf{E}$, help encode resource consumption and differentiate among edge types. For instance, precedence edges have the feature vector $\mathbf{e_{ij}} = [1, 0, 0, 0, 0]$, while resource consumption edges follow the pattern $\mathbf{e_{ki}} = [0, 1, 0, 0, r_{i,k}]$. To represent task durations and available resources, we employ task and resource node features ($\mathbf{v_i} \in \mathbf{V_T}$ and $\mathbf{v_k} \in \mathbf{V_R}$, respectively). A visual representation of the RCPSP is depicted in Figure 33. Unlike edge features, node features evolve with each graph-Transformer layer. The representation outlined above facilitates direct processing using established GNN models and converts an RCPSP into a graph for PyTorch Geometric. It sets up variables for tasks and resources and creates mappings for task dependencies and resource use. The representation effectively organises nodes for resources and tasks and crafts lists for tracking resource consumption and task lengths. It pinpoints start and end tasks and transforms the scheduling details into tensors. Node and edge features are then crafted to differentiate resources from tasks and to detail connections and resource use. The result is a PyTorch Geometric Data object filled with graph details and RCPSP specifics. A CP-sat solver incorporates Existing solutions with start times and project duration.

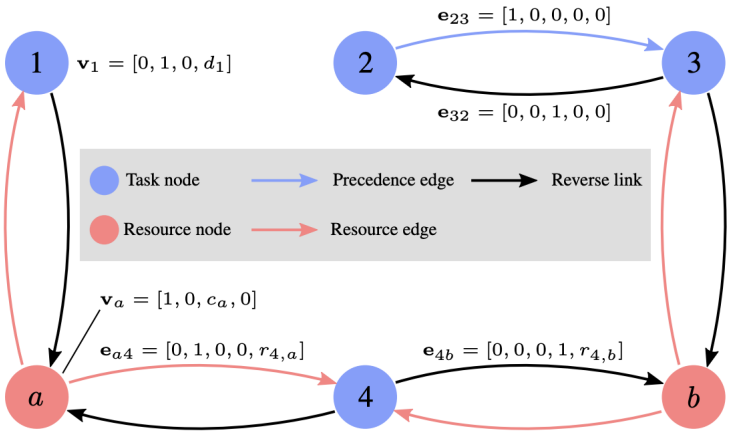

Figure 33: Visual representation of a GNN for Resource-Constrained Project Scheduling Problem, depicting task and resource nodes, along with the various types of edges such as precedence, resource, and reverse links, each annotated with their respective feature vectors to encapsulate the relationships and constraints within the scheduling problem. Note that all edges and nodes are annotated with features, but only some examples are shown here for simplicity (Teichteil-Königsbuch et al., 2023).

GNN architecture The architecture of the GNN employs a multi-layer message-passing mechanism, incorporating the graph Transformer model as described by Shi et al., 2020. The Graph Transformer model integrates the self-attention mechanism of transformers with Graph Neural Networks to process graph-structured data. The process computes the representation of a node by aggregating feature information from its neighbours, weighted by attention scores that reflect the relative importance of each neighbour. This approach enables the model to handle variable-sized input graphs and adaptively capture

local and global graph structures. Graph Transformers uniquely benefit from considering both node and edge attributes in their attention calculations, enhancing their ability to learn from the comprehensive semantics of graph structures.

The ResTransformer class implemented in PyTorch encapsulates a neural network architecture designed for processing graph-structured data using the graph Transformer model. The architecture is notable for its depth, featuring 15 blocks of Transformer-based convolution layers, and it integrates residual learning, where the output of each block is added to its input, thereby facilitating gradient flow and enabling the training of deeper networks. The network uses a hidden layer size of 128 units. We used 15 stacked residual transformer layers to ensure efficient information flow across large graph diameters. These 15 stacked residual transformer layers enhance information flow within the network, potentially mitigating issues related to vanishing gradients.

The input dimensions for nodes and edges are taken from the RCPSP graph representation described before, aligning the network to handle the graph data correctly. The initial layer is a Transformer convolution that adapts the node inputs to the internal hidden size using the provided edge features. Information aggregation at node i from its neighbouring nodes occurs as described in Equation 38. The subsequent layers are organised into residual blocks, each performing a sequence of operations: normalisation, activation via ReLU, and another Transformer convolution, followed by a residual connection.

The forward pass of the model involves logarithmic scaling of inputs, sequential processing through the initial Transformer layer and the residual blocks, and normalisation of the output to stabilise the gradients. The final stage involves a linear transformation to a binary output, which is scaled to produce an integer value. This architecture aims to harness the self-attention mechanism of Transformers to capture complex patterns in graph data. At the same time, the residual connections work to prevent the vanishing gradient problem commonly faced when training deep networks. The attention mechanism, reminiscent of Vaswani et al., 2017, is essential to determine the weights $\alpha_{i,j}$, as seen in Equation 39.

$$\mathbf{v_i'} = \phi(\mathbf{v_i}; \mathbf{V}, \mathbf{E}) = \mathbf{W_1 v_i} + \sum_{j \in N(i)} \alpha_{i,j}(\mathbf{W_2 v_j} + \mathbf{W_5 e_{ji}}), \tag{38}$$

$$\alpha_{i,j} = softmax(\frac{\mathbf{W_3 v_i^T}(\mathbf{W_4 v_j} + \mathbf{W_5 e_{ji}})}{\sqrt{o}}) \tag{39}$$

Transformations rely on the weight matrices $\mathbf{W_i}$. Residual connections for each layer aid in network training, with each layer having distinct weights. Implementing nonlinear transformations ($ReLU(x) := max(0, x)$) and normalisation (ψ), the complete chain of operations in a residual transformer layer follows:

$$\mathbf{v_i'} = \mathbf{v_i} + \phi(ReLU(\psi(\mathbf{v_i})), \mathbf{V}, \mathbf{E}) \tag{40}$$

$$\psi(\mathbf{v_i}) = \frac{\mathbf{v_i}}{C} \quad \text{where} \quad C = \|\mathbf{v_i}\|_2 \tag{41}$$

Training GNN weights for RCPSP The training process for optimising scheduling tasks using a GNN begins with the construction of a batch of GNNs, with each network representing a distinct RCPSP. This collective batch is integral for back-propagation during the weight update phases, which refines the GNN's prediction capabilities as detailed in Algorithm 1. The training algorithm employs this GNN architecture to process the graph-structured data efficiently, concluding with the MSE loss function that calculates the mean squared error loss for each data point and averages it for the batch.

This loss computation is crucial for the iterative learning and fine-tuning of the model through backpropagation. Alongside this, constraints are rigorously checked to validate start times, precedence, and resource constraints, as well as to compute the makespan. The model's parameters are reinitialised before each fold in a K-Fold cross-validation scheme to maintain training integrity and avoid contamination from previous training iterations. Throughout each epoch, metrics such as constraint violations and makespan are tracked, informing the optimisation of the model weights via the optimiser. By archiving the best-performing models, the training function secures the retention of the most efficient schedules, which is essential for achieving the most effective scheduling outcomes in the long run.

Algorithm 1 Training GNN Weights for RCPSP (Teichteil-Königsbuch et al., 2023)

Input: Set P of RCPSP problems with their corresponding known solutions $\chi = (x_P)_{P \in P}$

Output: GNN weights $\mathbf{W}$ trained to fit the solutions of P's problems

1: Build set B_{train} of training batches containing GNN encodings of problems in P and their solutions from χ
2: **for** all $0 \leq$ epoch $<$ nb_epochs **do**
3: **for** all batch $b \in B_{\text{train}}$ **do**
4: Predict starting dates $\hat{x_b}$ from the GNN batch b
5: Compute the mean-squared error loss $\mathcal{L}_{\text{MSE}}$ between $\hat{x}_b$ and the known solutions x_b of the RCPSPs encoded in the GNNs in b
6: Backpropagate $\mathcal{L}_{\text{MSE}}$ and update $\mathbf{W}$
7: **end for**
8: **end for**
9: **return** W

To ensure effective training, we extracted 1632 diverse instances from the PSPLIB, accounting for 80% of the total instances, and conducted training for 20,000 epochs. These instances were labelled using the CP-SAT solver, which operated under a 15-minute timeout. However, it is crucial to note that our methodology is independent of the specific RCPSP solver used for labelling.

As training progresses, the precedence violation percentage escalates due to the GNN's drive to learn and emulate schedules optimised by the CP-sat solver, inadvertently pushing against precedence constraints. However, this does not impact resource constraints, as the GNN continues to better its adherence to those established from the CP-sat solution benchmarks. The inference component is designed to fine-tune the predictions of the GNN, guiding them towards feasibility, as detailed in the following section.

4.6.2 Inference

Initially, we provide an overview of the schedule generation scheme, followed by an in-depth explanation of the inference component.

A schedule generation scheme is a procedural approach to compute feasible schedules from an alternative solution representation. Notably, the methodology adopts two main schemes: the Serial Schedule Generation Scheme (SSGS) and the Parallel Schedule Generation Scheme (PSGS). A more profound contrast between them can be found in Kolisch and Hartmann, 1999. The SSGS is an integral method for creating feasible schedules for RCPSP, ensuring that schedules adhere to resource limitations and task dependencies. It begins with a priority list of tasks, which is established based on heuristic rules or optimisation algorithms. SSGS allocates resources to tasks sequentially, prioritising the task that can be

started immediately, considering resource availability and task interdependencies. It meticulously checks that the allocation does not surpass resource capacities and schedules tasks at the earliest opportunity while satisfying precedence relations. This guarantees the feasibility of the schedule in terms of resource constraints and task dependencies.

In practical application, a system utilising SSGS starts by generating a provisional schedule, also known as a dummy solution, which forms a baseline. The system proceeds to compute a feasible solution following the initial task order. This solution involves recording the makespan and start time details and integrating these findings into a dictionary. Upon completing the batch, the system evaluates the average results, excluding invalid instances. If the batch contains only one instance, a feasibility check is conducted. If valid, a detailed schedule is constructed. The dictionary encapsulates these results, providing a comprehensive view of the scheduling system's efficacy, including makespan and start time for each task. SSGS is designed always to produce a feasible schedule.

Post-training, the GNN predictions exhibit minor constraint violations, hinting at the need for correction to achieve a feasible schedule. To remedy this, the strategy implemented is described in Algorithm 2 and further illustrated in Figure 32.

The algorithm presented delineates a method for generating a solution schedule for RCPSP by utilising the pre-trained weights of GNN. It initiates by constructing a GNN that is tailored to the specific instance of the RCPSP, employing the pre-trained weights to parameterise the network. The GNN, once established, processes the RCPSP instance to predict a potential solution, which primarily consists of inferred starting times for each task within the project.

Following the predictions, the algorithm proceeds to extract an ordering of tasks based on these predicted start times. This ordering is crucial as it establishes the sequence in which tasks should ideally be undertaken. With this sequence at hand, the algorithm then employs SSGS. The Serial SGS methodically converts the task ordering into a feasible schedule, ensuring that each task is allocated resources without violating constraints.

The end result of this process is the production of a solution schedule. This schedule, derived from the predictions of the GNN and refined by the systematic approach of the SGS, represents a feasible schedule for the RCPSP. Given the Serial SGS's capability to derive feasible schedules from existing optimal task orders, we anticipate that the reconstructed schedules from the almost-optimal GNN schedules will maintain a high-quality standard.

Algorithm 2 Scheduling with Trained GNN Weights (Teichteil-Königsbuch et al., 2023)

Input: Given RCPSP P with unknown solution; GNN trained weights $\mathbf{W}$

Output: Solution schedule x of P

1: Build GNN $G_{\mathbf{W},P}$ from P and using weights $\mathbf{W}$
Predict from $G_{\mathbf{W},P}$ the solution $\hat{\mathbf{x}}$ of P
Extract tasks ordering $O_{\hat{\mathbf{x}}}$ from inferred starting dates $\hat{\mathbf{x}}$
Construct solution schedule x^* by running SGS on $O_{\mathbf{x}^*}$
return x^*

4.7 Results

4.7.1 Performance comparison with CP solver

The test data set used for comparison comprises 408 random instances, representing 20% of the total problem instances, while the remaining 80% is utilised for training. These test data set instances are distributed as follows: 22.5% with 30 tasks, 23.5% with 60 tasks, 26.5% with 90 tasks, and 27.5% with 120 tasks. The code is implemented in Python within a Visual Studio Code environment utilising an 8-core Apple M1 CPU @3.20Ghz, 16GB RAM and an 8-core GPU. We used a suite of libraries, including PyTorch, PyTorch Geometric, tensorboard for visualisation, optimisation tools or tools, and scikit-decide. Two main comparisons were drawn between the Graph Neural Network and CP-sat: Computational Efficiency and Solution Quality.

Computational Efficiency: The comparative runtimes of the GNN model's inference and CP-sat's execution in pursuit of equivalent makespans are illustrated in 34 and 35, with corresponding data tabulated in 19. The results from CP-sat are generated internally by our computational processes.

The table presents an in-depth comparison of the computation times for GNN and CP-sat when the quality of the resulting makespan is kept identical across 408 instances. The data reveals that CP-sat exhibits a broad range of computation times, from a minimum of approximately 0.0084 seconds to a maximum of 363 seconds, a median time of 0.14 seconds, and an average computation time of 1.88 seconds. The variability in CP-sat's performance is further evidenced by a standard deviation of 18.22. In stark contrast, the GNN method showcases much tighter computation times, ranging from a minimum of roughly 0.0117 seconds to a maximum of only 0.10 seconds, with a significantly lower median of 0.02 seconds and mean of 0.02 seconds, indicative of its consistent and swift performance, underscored by a minimal standard deviation of just 0.01 seconds.

When comparing relative performance, the statistics presented in boldface in the table are the ones of importance and relevance. Utilising a trained GNN significantly accelerates finding solutions for scheduling problems compared to using the CP-SAT solver. In 330 out of 408 test cases, corresponding to 81% of instances, the GNN model demonstrates superior performance by outpacing CP-sat in computational speed. This is illustrated in Figure 34, where instances lying above the diagonal indicate that GNN outperforms CP-sat in terms of speed. For 40% of the compared instances, CP-sat's computation time was at a minimum ten times, and up to 10,000 times longer than that of GNN, as depicted in Figure 35. Conversely, in instances where CP-sat outperformed, its speed advantage was capped at a maximum of fourfold, indicated by the minimum relative performance value of 0.27.

Table 19: Tabulated comparison between GNN and CP-sat computation times with identical quality. This table provides key metrics such as mean and standard deviation to offer a comprehensive view of how the two computation methods compare across various instances.

	Count	**Minimum Value**	**Maximum Value**	**Median**	**Mean**	**Standard Deviation**
CP-sat (Figure 34)	408	8.4e-3	363	0.14	1.88	18.22
GNN (Figure 34)	408	1.17e-2	0.10	0.02	0.02	1.00e-2
Relative Performance(Figure 35)	-	**0.27**	**9638**	597	1651	2.13e3

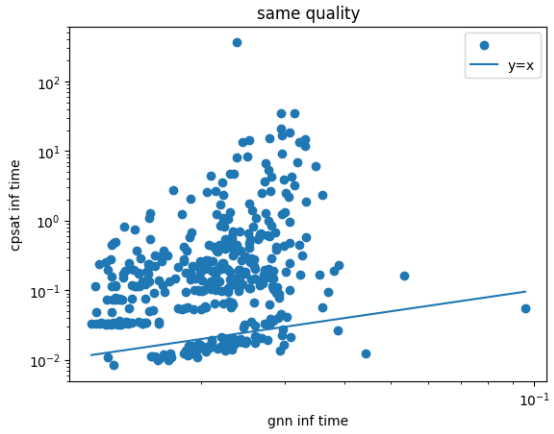

Figure 34: Scatterplot illustrating the comparison between GNN computation time and CP-sat computation time with the same quality. Points on the diagonal indicate identical makespans, while those above the diagonal show the GNN computation time surpassing CP-sat.

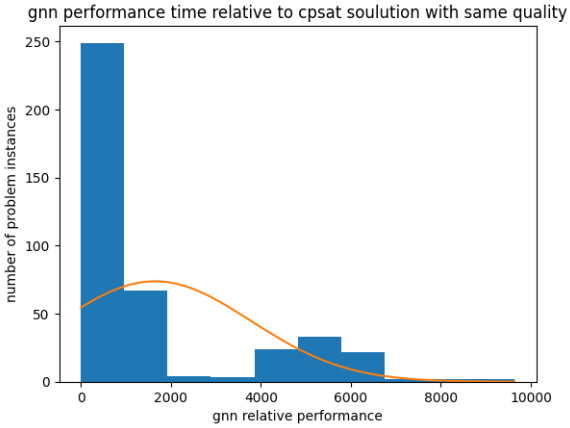

Figure 35: Graphical representation of GNN's relative performance compared to the best CP-sat solution over 408 problem instances.

Solution Quality: When given a maximum runtime of 5 minutes, the quality of the CP-sat solution was compared to that of GNN, as visualised in Figures 36 and 37. Also, a comprehensive overview of these results is provided in Table 20. Once more, the statistics highlighted in bold are of primary significance and relevance. Notably, CP-sat's makespan ranges from a minimum of 35 to a maximum of 357, with a median makespan of 92 and an average of 106, coupled with a standard deviation of 50.64. In comparison, GNN displays a tighter computation makespan distribution, with its values spanning from 35 to 316, a median of 85, and a lower average makespan of 97.60, alongside a standard deviation of 44.89, suggesting better performance in the same runtime. The results, as visualised in Figure 37, reveal that while the Graph Neural Network (GNN) exhibited a marginal dip in performance relative to the CP-sat solver, it nonetheless delivered praiseworthy outcomes. The mean relative performance of GNN stood at 0.93, as emphasised in bold, signifying that, on average, GNN's efficiency closely mirrors that of CP-sat and, in certain instances, surpasses it. This comparative analysis was substantiated by GNN achieving parity with the optimal solutions in nearly half of the cases, specifically 203 out of 408. Moreover, the schedules generated by GNN were, on average, less than 7% longer in makespan compared to those determined by CP, underscoring GNN's commendable performance in computational problem-solving.

Table 20: Statistical summary comparison between GNN makespan and CP-sat makespan with the same runtime. This table provides key metrics such as mean and standard deviation to offer a comprehensive view of how the two computation methods compare across various instances based on their makespan.

	Count	**Minimum Value**	**Maximum Value**	**Median**	**Mean**	**Standard Deviation**
CP-sat (Figure 36)	408	35	357	92	106.41	50.64
GNN (Figure 36)	408	35	316	85	97.60	44.89
Relative Performance(Figure 37)	-	0.68	1.09	0.94	**0.93**	7.48e-2

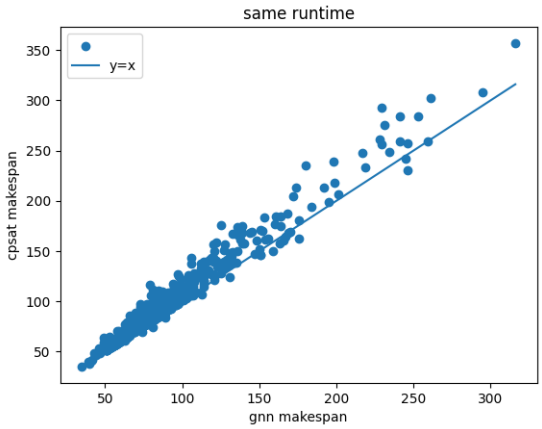

Figure 36: Scatterplot illustrating the comparison between GNN makespan and CP-sat makespan with the same runtime. Points on the diagonal indicate identical makespans, while those above the diagonal show the GNN makespans surpassing CP-sat.

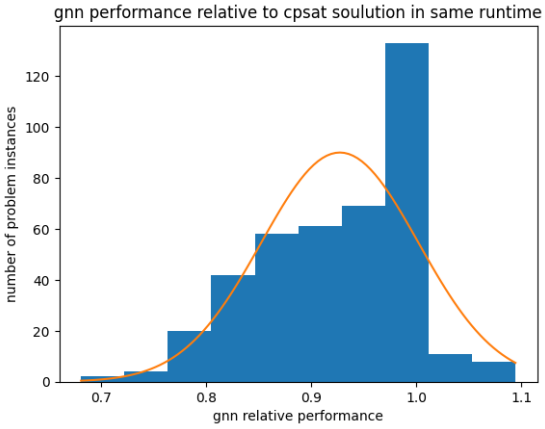

Figure 37: Graphical representation of GNN's relative performance compared to CP-sat solution in the same runtime over 408 problem instances.

4.7.2 Results in Inland Waterway Transport context

In the realm of Inland Waterway Transportation (IWT), we utilised the Resource-Constrained Project Scheduling framework to plan cargo operations across a 30-node port network. We incorporated predictive analysis to create efficient schedules, maximising resource use and reducing processing time. In an IWT context, JSSP and RCPSP deal with scheduling but at different granularities. JSSP sequences individual tasks, such as unloading, for ships at specific ports, often focussing on local resources. In contrast, RCPSP orchestrates cargo activities across a port network, considering task interdependencies and broader resource allocations. Essentially, JSSP optimises within ports, while RCPSP looks at interconnected operations across them.

Our study considered 30 nodes as ports, each with specific constraints and resources. These ports serve as key points along the waterway network where cargo transportation operations occur. We addressed the complexities associated with real-world IWT scenarios.

Our model incorporates four renewable resources that mirror the various assets available in IWT operations. These resources include factors such as berths, cranes, labour, and vessels, which are crucial to ensure efficient cargo handling and transportation. Berths are essential for cargo ships to dock and unload their cargo. Each berth has a limited capacity and can handle only one ship at a time. Cranes are necessary to lift and move cargo from ships to storage areas. Each crane has a certain efficiency rate, and the total time for unloading is affected by the number of available cranes. Labour resources, including dockworkers and warehouse personnel, are needed to assist unloading cargo. The availability and efficiency of labour affect the loading process. The unloaded cargo must be stored temporarily before further processing or distribution. Storage space availability influences the rate at which cargo can be unloaded. By including these resources, our scheduling solution optimises their allocation and usage, thereby enhancing the overall effectiveness of the waterway system.

Furthermore, considering precedence relations within the RCPSP framework is paramount to reflecting the sequential nature of IWT operations. Precedence relations capture the dependencies between tasks, mirroring the requirement that specific tasks must be completed before others can begin. These relations

mirror the sequential loading, transportation and unloading of cargo along the waterway, ensuring the entire process is executed seamlessly. Perishable cargo, such as fresh produce, must be unloaded and stored before non-perishable goods. This precedence relation ensures that time-sensitive cargo is given priority. Certain cargo might require downstream processing, such as quality checks or additional packaging. Tasks for downstream processing must follow the initial unloading process. The availability of resources, such as cranes and labour, can create precedence relations. For example, unloading tasks requiring cranes might need to wait for crane availability.

The outcomes of a workable schedule are presented here in Figure 38, illustrating the approach's effectiveness. Initially, the Graph Neural Network predicts the start times for each task. Subsequently, the schedule is devised through inference and is then cross-validated against the meticulously optimised schedule. The visual representation below illustrates this comparison. Remarkably, the results are remarkably congruent, underscoring the reliability of the model's outputs. This congruence between the GNN-predicted and optimised schedules highlights the model's capacity to generate dependable and accurate outcomes.

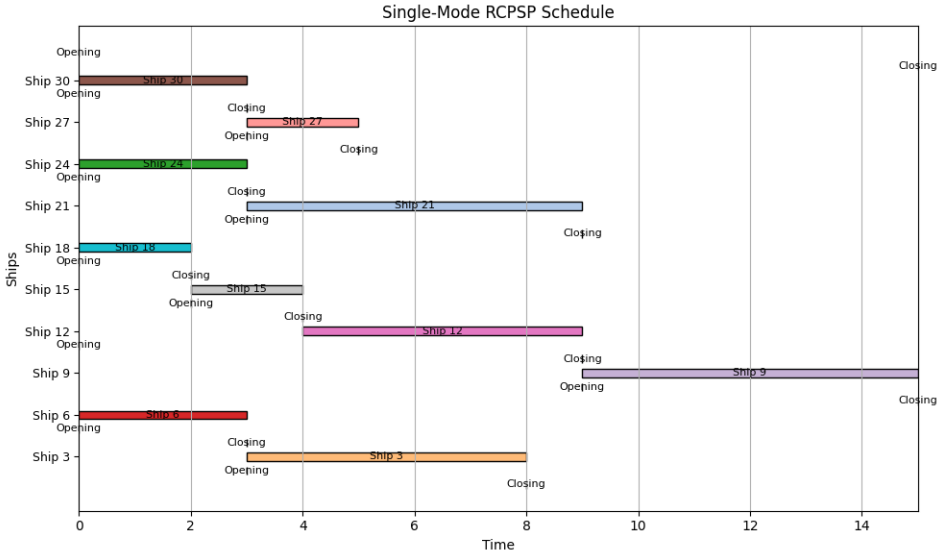

Figure 38: Gantt chart representation of the scheduling window for various ships in a single-mode RCPSP scenario. Each horizontal bar corresponds to a ship's time window, with distinct colours indicating opening and closing phases. This visual schedule facilitates the tracking of operations, ensuring optimal utilisation of available resources over the time horizon.

In conclusion, our adaptation of the RCPSP framework to the domain of Inland Waterway Transportation demonstrates its versatility in addressing the intricacies of scheduling cargo operations within a complex network of ports. Our approach optimises scheduling by incorporating renewable resources, handling precedence relations, and leveraging predictive inference to enhance resource utilisation and adaptability. This framework has the potential to significantly improve the efficiency, reliability, and resilience of current IWT scheduling practices.

4.8 Conclusion

Resource-Constrained Project Scheduling Problems (RCPSPs) resemble the intricacies of the Job Shop Scheduling Problem (JSSP) in the optimisation domain due to their NP-complete nature, making the robust solution of extensive instances a demanding task. Addressing this challenge, a novel approach

leverages Graph Neural Network architecture to imitate the outputs of a Constraint Programming (CP) solver. This strategy is analogous to using distinctive graphs for representing job and machine sequences in JSSP. The objective is to capture the intricate structure of RCPSPs, transforming raw solutions into more efficient and feasible ones. By integrating the Schedule Generation Scheme (SGS) post-processing method, the GNN ensures that its schedules adhere to constraints, even if they initially violate them.

GNN and CP-sat are compared using a test data set comprising 408 random instances. Of these instances, the distribution varied by the number of tasks, ranging from 30 to 120 tasks per instance. The comparison highlighted two key aspects: computational efficiency and solution quality. Impressively, GNN's computational efficiency outperformed CP-sat in more than 80% of the instances, sometimes even being up to 10,000 times faster. On the solution quality front, with a maximum runtime restriction of 5 minutes, GNN's results were slightly suboptimal compared to CP but still noteworthy. GNN matched CP's optimal solutions in almost half the test cases.

The study's real-world application is demonstrated in the Inland Waterway Transportation (IWT) context. Here, the RCPSP framework is adapted to schedule cargo transportation operations across a complex network of 30 port nodes. This application considered intricate aspects such as berth allocations, crane utilisation, labour efficiency, and storage space availability. The RCPSP's precedence relations played a crucial role in ensuring that tasks occurred in the correct sequence, with perishable cargo, for instance, being prioritised over non-perishables. The model outputs were visualised, and the remarkable overlap between GNN-predicted schedules and optimally devised schedules was evident, emphasising the model's robustness and reliability.

In summary, this research underscores the versatility of the RCPSP framework, particularly when adapted to the specialised domain of Inland Waterway Transportation. Through its novel approach to integrating renewable resources, managing precedence relations and utilising predictive inferences, the study argues for an optimal scheduling mechanism that could revolutionise current practices in IWT. This not only promises enhanced resource utilisation but also offers greater adaptability, paving the way for more efficient and resilient IWT operations.

5 Waterway Ship Scheduling Problem

5.1 Introduction

Inland waterway transport is vital for global and regional economies T. Notteboom et al., 2020. Europe extensively uses its vast river and canal network for bulk goods transportation, positioning inland waterways as an alternative to road and rail systems (Hofbauer & Putz, 2020). Maritime operators globally focus on optimising ship operations, making port waiting times a critical efficiency factor. Ports like Shanghai and Hamburg encounter unique challenges like tidal restrictions and width constraints (Du et al., 2015; Ernst et al., 2017; Lübbecke, 2018). In the midst of increasing maritime traffic, ports prioritise reducing ship waiting times to enhance efficiency, reduce emissions, and improve accessibility (Du et al., 2015; Du et al., 2011; Verstichel, De Causmaecker, et al., 2014b).

Waterways function as transport corridors with limited capacity, consisting of navigable waters supported by navigational systems and characterised by particular water conditions. These could be natural or artificial and often have to meet several criteria to be considered viable for shipping. For instance, they must offer sufficient depth to accommodate ships, and their width must be carefully calibrated. While most waterways are two-directional, allowing vessels to pass in opposite directions, the width must exceed the combined breadth of two navigating ships, with an added margin for safety. In certain instances, the waterway might only allow for the isolated passage of individual vessels.

Given these challenges, this thesis aims to develop a comprehensive model suitable for operational decision-making under such complex conditions. The principal objective is to create a methodology that enables optimal planning to minimise ship delays. Despite the growth and complexity of maritime traffic, planning and management have been predominantly reactive and ad hoc in nature. This thesis will explore the potential of advanced planning systems to address these gaps and enhance efficiency in maritime operations.

5.2 WSSP formulation

The Waterway Ship Scheduling Problem (WSSP) aims to optimise ship turnaround times and minimise idle periods within busy port systems. As shown in the example from Hill et al., 2019 and depicted in Figure 39, the process begins when a ship arrives and might face a short wait due to congestion. Ships navigate through the port's inbound and outbound passages of two distinct waterways, where they proceed to port operations. After handling cargo and other activities, vessels may experience a port wait before departing. The flexibility to choose between Waterway 1 and Waterway 2 allows ships to manoeuvre based on real-time traffic conditions and operational feasibility. Environmental factors like tidal patterns and the physical limitations of each waterway further guide such decisions. The diagrammatic representation in the Figure underscores the intricate flow of maritime traffic and the necessity for strategic scheduling to enhance efficiency in port operations and waterway management.

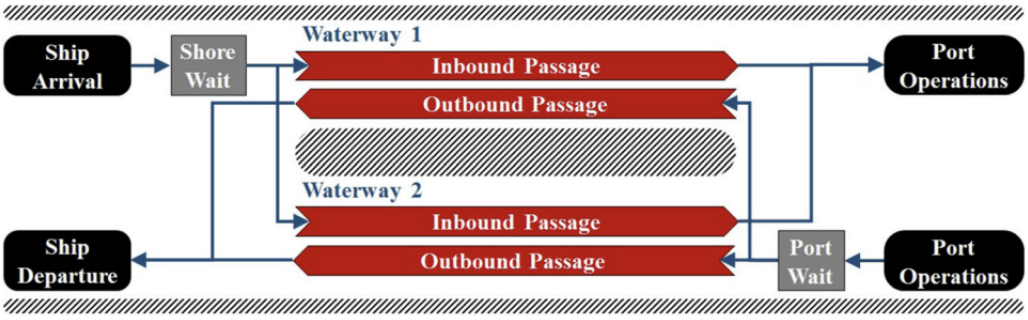

Figure 39: Flow diagram outlining the stages of a ship's journey through a port and waterway system, based on the example from Hill et al., 2019.

Example Consider navigating a complex network of waterways that interconnect four significant ports: Ports A, B, C, and D. The design of these canals is such that their narrow width permits only a single ship to traverse at any given moment. Tidal restrictions further constrain this issue. Among the vessels navigating this complex port network, Ship 1 departs from Port A, cutting through the water with a 7-meter draught and transporting containers that may include hazardous materials requiring particular handling protocols. Ship 2 embarks from Port B with a 5-meter draught, its hold filled with bulk materials essential for various industries. Ship 3, leaving from Port C with a 6-meter draught, is burdened with a diverse array of containers, some demanding electricity for cooling to preserve sensitive contents. Ship 4, with an imposing 8-meter draught, sets off from Port D, its vast capacity taken up by voluminous bulk commodities. Lastly, Ship 5 sails out from Port A as well, its 5.5-meter draught supporting containers that may carry dangerous goods, each adhering to stringent safety standards to ensure secure passage. Time plays a crucial role in their navigation. The high tide lasts from 10:00 AM to 4:00 PM and is favoured by larger ships, especially those with draughts exceeding 6 metres. However, once 4:01 PM strikes, the low tide sets in, lingering until 9:59 AM the following day. These ships also have to contend with a series of constraints. The limited width of the canal means that only one ship can pass at any moment. Tidal variations dictate that vessels with draughts over 6 metres must move during high tide. Each port has its specific hours during which ships can enter, contingent on the particular waterways. Moreover, contractual obligations have mandated that Ships carrying hazardous materials receive priority. Amidst these challenges, the main objective is to ensure that all ships traverse this intricate network in the shortest time possible while adhering to the constraints.

Definitions The Waterway Ship Scheduling Problem seeks to effectively organise the movement of ships using available waterways within a set time frame. According to Hill et al., 2019, the problem is described as:

Entities Involved:

1. Ships ($V = \{1..., v\}$): Comprised of both incoming ($V_1 = \{1, 2..., v_1\}$) and outgoing ($V_2 = \{1, 2..., v_2\}$) ships.
2. Waterways ($W = \{1..., w\}$): Channels facilitating ship movements.
3. Time Steps ($H = \{1..., h\}$): Denoting specific intervals within a two-day horizon.

This daily operational challenge employs a discrete and equidistant time horizon $T = t_0, t_0 + d, ..., t_0 + h \cdot d$,

where each time step $t \in T$ represents the period $[t, t + d]$ and d its time extension.
Waterway Attributes:

- time variations, alongside a fixed width, $B(w)$.
- A waterway can be accessed in two distinct manners: 'in' for entry and 'out' for exit.
- The maximum number of ships that can enter simultaneously through the waterway w is defined by $Q_{in}(w)$.
- Conversely, waterway w can facilitate the exit of up to $Q_{out}(w)$ ships simultaneously.

Ship Attributes:

- Each ship $v \in V$ is characterized by certain specifications: width ($b(v)$), length ($l(v)$), and depth ($d(v)$).
- Inbound vessels $v_{in} \in V_{in}$ have an Estimated Time of Arrival (ETA), $a_{in}(v_{in})$, and the latest permissible time entry, $a'_{in}(v_{in})$.
- Outbound ships $v_{out} \in V_{out}$ have a predetermined last departure time, $a'_{out}(v_{out})$. Their earliest possible exit, typically after their port operations, is $a_{out}(v_{out})$.
- Direction-oriented travel times are denoted as $z_{in}(v_{in}, w)$ for incoming ships and $z_{out}(v_{out}, w)$ for those departing, through waterway w.
- Furthermore, each ship is allocated a specific time slot, T_w, marking its availability.

Scheduling Prerequisites:
To solve the WSSP efficiently, it is imperative to assign an entry waterway and corresponding entry time for every inbound ship, and for every outbound ship, designate a departure waterway with an appropriate departure time. The following conditions are mandatory:

- Concurrent waterway usage by multiple ships should not exceed prescribed limits.
- Ships are permitted to access a waterway solely during operational hours.
- Inbound scheduling should not precede the ship's ETA.
- The waterway's depth should be compatible with the ship's draught at any given time.
- Outbound scheduling should not be before the ship's prescribed departure.
- The waterway width should accommodate the aggregate widths of ships using it simultaneously.

Mathematical Formulation Though seemingly distinct from our GNN methodology, the classic mathematical model offers critical insights into the problem's core structure. The model presented by Hill et al., 2019 succinctly captures the scheduling constraints and objectives. It centres on reducing the total of specific scheduling parameters while adhering to constraints like resource limits and time windows. Typically, such problems draw solutions from Mixed-Integer Linear Programming (MILP) foundations, as seen in the approach adopted by Lalla-Ruiz et al., 2018. The study by Hill et al., 2019 serves as a reference and comparative baseline later. While our GNN method diverges from this linear perspective, understanding this traditional formulation equips us with insights into the inherent intricacies of the WSSP. This foundational grasp aids in tailoring neural network structures efficiently, ensuring the

GNN encapsulates essential problem elements, harnessing the versatility and non-linear prowess of neural networks. The problem's mathematical formulation, derived from Hill et al., 2019, is presented below:

$$\text{minimise} \quad \sum_{j\in J}\sum_{m\in M_j}\sum_{t\in T} t - \text{EST}_j x_{j,m,t} \tag{42}$$

subject to

$$\sum_{m\in M_j}\sum_{t\in T} x_{j,m,t} = 1 \qquad \forall j \in J, \tag{43}$$

$$\sum_{j\in J}\sum_{m\in M_j}\sum_{t'<t\leq t+d_m} u_{m,r}x_{j,m,t'} \leq q_{r,t} \qquad \forall r \in R, t \in T, \tag{44}$$

$$\text{EST}_j \leq \sum_{m\in M_j}\sum_{t\in T}(t-d_m)x_{j,m,t} \leq \text{LST}_j \qquad \forall j \in J, \tag{45}$$

$$x_{j,m,t} \in \{0,1\} \qquad \forall j \in J, m \in M_j, t \in T. \tag{46}$$

As outlined in Equation 42, the goal is to minimise the difference between task completion times and release times. The provision in Equation 43 ensures each ship (or job) is allocated a schedule, and precisely one waterway (or mode) is selected. The capacity stipulations in Equation 44 prevent resource overutilisation in any given time frame. For each resource and timeframe, a knapsack limitation is set. Although all modes are up for consideration, they will only utilise resources if selected. As for the WSSP, Equation 44 maintains consistent adherence to waterway constraints, such as traffic and spatial limitations, throughout the planning span. The changing resource availability across time, especially pertaining to varying waterway depths, is captured by the right-hand side of Equation 44. The constraints in Equation 45 dictate the permissible start times for tasks, corresponding to the timeframes in which ships are allowed to access a waterway. The model introduces a unique set of decision variables, as seen in constraint Equation 46. Specifically, a binary variable $x_{j,m,t}$ is used, which becomes 1 when task j concludes at time t in mode m.

5.3 Literature

5.3.1 Waterways Problems in general

Research targeting specific issues in inland waterways is relatively scarce. Currently, the Vessel Traffic Service (VTS) staff, guided by port navigation rules, manually schedules ship docking at ports. This method can lead to inefficiencies and prolonged ship waiting times (Praetorius et al., 2010). The introduction of time window booking systems for ships could offer a solution. Allowing ships to reserve specific time slots for docking ensures a smoother flow of traffic and minimises congestion. There is an increasing necessity for adopting such systematic, theoretically backed scheduling approaches to navigate complex traffic scenarios.

In addressing these needs, several scholars have proposed methods to tackle the intricate ship scheduling dilemma. In particular, methodologies such as the Lagrangian relaxation algorithm (Jia et al., 2019) and the column generation algorithm (S. Li & Jia, 2019) have emerged. The relevance of anchorage staging to reduce ship traffic congestion, especially in tidal seaports with bidirectional channels, was explored by Jia et al., 2019. They introduced a strongly proven NP-hard MILP model and for the WSSP variant, a time-efficient Lagrangian relaxation heuristic. This approach was further advanced by S. Li and Jia, 2019, who presented a column generation technique with considerable time-saving advantages over the earlier heuristic.

Nevertheless, precise algorithms have demonstrated extended computational times, making them impractical even for modest problems. Therefore, there has been a pivot towards heuristic solutions (Hongxing et al., 2018; Lalla-Ruiz et al., 2018; Meisel & Fagerholt, 2019; X. Zhang et al., 2017; X. Zhang et al., 2016).

Among these studies, X. Zhang et al., 2016 improved the widely adopted FCFS strategy, focussing on channel-berth coordination in a single-direction port. Their method integrated a simulated annealing approach and a multi-population genetic algorithm. Similarly, Lalla-Ruiz et al., 2018 and Hill et al., 2019 explored the unique challenges of two-way channels in Shanghai Port, incorporating factors such as vessel draught limits and tidal impacts on water levels into their models.

Existing optimisation strategies, though promising, present theoretical gaps. An outstanding issue remains in the parameter selection for optimisation algorithms, directly affecting the solution quality. Meisel and Fagerholt, 2019 and X. Zhang et al., 2017 considered vessel dynamics such as variable speeds, waterway width limitations, and consistent velocities, providing innovative perspectives on two-way waterway management.

Furthermore, other efforts such as Günther et al., 2011; Lübbecke et al., 2019 focused on the bidirectional traffic of the Kiel Canal, while Yang et al., 2014 worked on the optimisation of the Yangtze River container liner network. Both of these studies prioritised minimising passage and operational times. Lastly, the Istanbul Strait's unique traffic patterns were investigated by Ulusçu et al., 2009, leading to a tailored algorithm for its unique challenges.

In conclusion, Table 21 presents a comprehensive overview of the literature findings related to optimisation problems in waterways. The focus of this thesis is primarily aligned with the studies presented by Lalla-Ruiz et al., 2018 and Hill et al., 2019. These articles serve as a foundation due to their specific attention to the Waterway Ship Scheduling Problem. The problem was initially introduced and explored in Lalla-Ruiz et al., 2018. Subsequently, its translation into the Resource-Constrained Project Scheduling Problem was detailed in Hill et al., 2019. Given the complexities and challenges identified in the general literature, using the insights and methodologies from these two pivotal studies offers a strategic avenue to advance the understanding and solutions related to the WSSP.

Table 21: Overview of WSSP general literature

Article	Summary
Jia et al., 2019	Proposed a MILP model to mitigate ship traffic congestion in tidal seaports using staging anchorages; demonstrated the problem's strong NP-hard nature and introduced a Lagrangian relaxation heuristic.
S. Li and Jia, 2019	Extended the study of Jia et al., 2019 and introduced a column generation algorithm based on a set-partitioning model, outperforming the previous heuristic in computational time.
X. Zhang et al., 2016	Focused on channel-berth coordination in one-way ports; improved the performance of the current FCFS strategy using a simulated annealing method combined with a multi-population genetic algorithm.
Lalla-Ruiz et al., 2018	Introduced the Waterway Ship Scheduling Problem for two-way channels in Shanghai port using a mixed-integer linear model to minimise ship transit time, considering channel constraints and tidal impacts.
Hill et al., 2019	Enhanced the model proposed by Lalla-Ruiz et al., 2018 using a project scheduling technique.
Hong-xing et al., 2018	Presented a model incorporating real-world constraints like time slot allocation, traffic flow conversion, and coordination.
Meisel and Fagerholt, 2019	Developed a model for the Kiel Canal that considers variable vessel speeds and siding segment capacities for two-way vessel traffic scheduling.
X. Zhang et al., 2017	Introduced a multi-objective model for vessel traffic scheduling on a two-way waterway, focussing on safety intervals and assuming uniform vessel velocity.
Günther et al., 2011, Lübbecke et al., 2019	Addressed bidirectional ship traffic on the Kiel Canal, presenting a mathematical formulation to design liner routes and shipping networks aiming to minimise total passage time, including lock and siding waiting times.
Yang et al., 2014	Presented an integer programming model for optimising container liner networks on the Yangtze River, with the aim of reducing transportation costs and considering uniform operation times in each port.
Ulusçu et al., 2009	Developed an algorithm for scheduling incoming ships in the Istanbul Strait, considering its unique characteristics.

5.3.2 Waterways Problems with ML

Research into machine learning applications for the Waterway Ship Scheduling Problem remains nascent, with sparse literature available. A contribution comes from C. Wang et al., 2017, who employed a backpropagation (BP) neural network to predict vessel flow within waterway networks. Their rigorous residual analysis informed the training and data sampling methods. Given this early stage of machine learning developments, there is a clear need for further investigation and innovation to harness the capabilities of machine learning fully to tackle the complex challenges posed by the Waterway Ship Scheduling Problem.

5.4 Methodology

In this section, we aim to address the WSSP problem using GNN. We achieve this by converting the WSSP problem into a single-mode RCPSP problem, drawing inspiration from the work of Hill et al.,

2019. We then employ the GNN model, detailed in the RCPSP section and rooted in the findings of Teichteil-Königsbuch et al., 2023, to tackle the problem. Our approach begins by directly translating variables from WSSP to single-mode RCPSP. Subsequently, we delineate the necessary modifications to these variables to ensure compatibility with the single-mode RCPSP, such as introducing vessel time windows, defining the horizon, and establishing precedence relations. We further discuss the datasets utilised—randomly generated and those based on marine traffic. The section ends with a case study centred on the Port of Duisburg.

5.4.1 Translating WSSP to RCPSP

Single mode RCPSP translation This paragraph provides a detailed translation of the Waterway Ship Scheduling Problem into a single-mode Resource-Constrained Project Scheduling Problem. This translation is distinctly bifurcated into properties pertaining to the waterways and those related to the ships. In terms of waterway properties, waterways, represented by W, with time-varying depths $D(w,t)$ and a constant width $B(w)$, have been encapsulated as separate resources, considering both their width and depth. It is noteworthy that the time-dependent variability of depth has not been incorporated into the model. The two-directional nature of these waterways has been addressed by allocating distinct resources for each direction, namely incoming and outgoing. Constraints that limit the simultaneous entry, $Q_{\text{in}}(w)$, and exit, $Q_{\text{out}}(w)$, of ships from the harbour are translated as specific resource capacities for incoming and outgoing vessels, respectively. Referring to ship properties, each ship, characterised by its measurements such as width $b(v)$, length $l(v)$, and depth $d(v)$, is depicted as an individual task in the RCPSP model. These characteristics of the vessel are then represented by appropriating the relevant resources for each individual task. The travel durations, dependent on the direction of movement, are integrated directly as task durations. A noteworthy aspect is the acknowledgement of time windows T_w, which determine when ships can traverse a particular waterway; however, this element has not been translated into the RCPSP model. Overall, the summary below presents a methodical plan for moving from the complexities of WSSP to the organised single-mode RCPSP framework centred on waterway and vessel attributes:

Waterway Properties

1. Original: Waterways $W \in W$ with time-dependent $D(w,t)$ and constant width $B(w)$.
   - Translation: Represented as total resource availability for each waterway separated into three resources: one for width (shared by both directions) and two for depth (one for each direction). Time-dependence is not implemented.
2. Original: 2-direction waterways.
   - Translation: Each waterway is divided into two resources, one for incoming and another for outgoing traffic.
3. Original: Maximum $Q_{\text{in}}(w)$ ships can enter the harbor simultaneously.
   - Translation: Added as a new resource capacity for incoming ships.
4. Original: $Q_{\text{out}}(w)$ is the maximum number of ships that can leave the harbour simultaneously.
   - Translation: Added as a new resource capacity for outgoing ships.

Ship Properties

5. Original: For each ship $v \in V$, dimensions are defined by $b(v)$, $l(v)$, and depth $d(v)$

- Translation: Each ship is modelled as a task, and these dimensions are specified using the required resources for each task.

6. Original: V_{in} number of arriving ships.
   - Translation: Each arriving ship is added as a separate task with a dummy duration to represent time windows with zero resource usage.
7. Original: Number of departing ships V_{out}.
   - Translation: The required resources are allocated for each departing ship's task.
8. Original: Direction-dependent travel times.
   - Translation: The necessary travel time is added as task durations.
9. Original: Each waterway $w \in W$ has a contiguous availability time window $T_w \subseteq T$ in which ships can pass.
   - Translation: This feature is not implemented.

Modifications for single mode RCPSP In addition to translation, we must also implement changes for the single-mode RCPSP. For constraints related to the arrival time or ETA, denoted as $a_{\text{in}}(v_{\text{in}})$, they have been added as separate tasks for every ship. These tasks incorporate dummy durations specifically designed for time windows with zero resource availability. This method ensures that arrivals or departures only occur after the earliest designated arrival or departure time, effectively capturing the precedence relations. Similarly, constraints regarding the utmost permissible time to enter an inbound waterway, indicated as $a'_{\text{in}}(v_{\text{in}})$, and the earliest and latest times that a ship can leave the port, represented by $a_{\text{out}}(v_{\text{out}})$ and $a'_{\text{out}}(v_{\text{out}})$ respectively, have been integrated as separate tasks for each vessel. These tasks also use dummy durations in time windows where resource availability is null, ensuring that the constraints are enforced adequately in the single-mode RCPSP model.

In this approach to scheduling, each task is linked to its dummy counterpart through precedence relations to regulate its timings. Specifically, a "start-to-start" relation is established between a task and its dummy to dictate the earliest start time, while a "finish-to-finish" relation ensures adherence to the latest start time. This methodology is similarly applied to set limits for departure times. However, this strategy of introducing time windows to the RCPSP using extra tasks and precedence relations is not without its drawbacks. It can notably amplify the complexity of the problem, especially as the number of tasks increases. This increased complexity can strain computational resources, prolong solution times, and make identifying feasible or optimal schedules more elusive. Consequently, heuristic algorithms, which may sacrifice solution optimality for speed, often become the go-to. Furthermore, the rigid nature of this model can hinder adaptability in dynamic scheduling scenarios, and its representation of time windows may not fully encapsulate real-world uncertainties. On a separate note, a simplifying assumption can be made for tasks by considering them homogeneous. This means treating all ships identically regarding their direction (incoming or outgoing), draught, and waterway width requirements. This assumption negates the need for diverse resource constraints tailored to different ship types. Below are the adjustments that have been made to align the WSSP constraints with the single-mode RCPSP framework:

Original: Arrival time/ETA is denoted as $a_{\text{in}}(v_{\text{in}})$.

- Modification: Added as a separate task for each ship with dummy durations for time windows with zero resource availability and precedence relations to ensure arrival/departure after the earliest possible time.

Original: The latest time it can enter an inbound waterway is $a'_{\text{in}}(v_{\text{in}})$.

- Modification: Added as a separate task for each ship with dummy durations for time windows with zero resource availability and precedence relations to ensure arrival/departure after the earliest possible time.

Original: The earliest time it can leave the port is $a_{\text{out}}(v_{\text{out}})$.

- Modification: Added as a separate task for each ship with dummy durations for time windows with zero resource availability and precedence relations to ensure arrival/departure after the earliest possible time.

Original: The latest port departure time for the ship $v_{\text{out}} \in V_{\text{out}}$ is denoted by $a'_{\text{out}}(v_{\text{out}})$.

- Modification: Added as a separate task for each ship with dummy durations for time windows with zero resource availability and precedence relations to ensure arrival/departure after the earliest possible time.

5.4.2 Definitions

Table 22 provides a detailed account of the resource constraints applicable to maritime navigation in environments with either two or four waterways. For the two-waterway scenario, the constraints are a more condensed set of constraints specifically tailored to fit the particular characteristics and capacities of Waterways A and B. On the other hand, for scenarios incorporating four waterways, the table meticulously specifies individual constraints for Waterways A through D. These include the total traffic volume, navigational breadth, and the maximum draft of vessels permitted for safe passage in and out of the harbour. These stipulated constraints play a pivotal role in ensuring maritime traffic's smooth and secure management, which becomes increasingly complex with more waterways.

Expanding the framework to encompass six waterways, with the addition of Waterways E and F, has introduced ten new resource constraints. These are akin to the constraints for Waterways C and D. This enhancement increases the total number of constraints to thirty, thus augmenting the table's capability to manage resources for an extended range of maritime traffic situations. The comprehensive approach ensures scalability and adaptability, allowing for precise control and optimisation of maritime operations across an increasing number of waterways. The table provides a comprehensive view of the resource distribution required for various maritime navigation scenarios, tailored according to the specific needs of either two or four waterway systems. Note that the limitation on the total number of incoming or outgoing ships in a waterway will be referred to as "Qlimit" henceforth.

Table 22: A comprehensive overview of resource constraints for maritime navigation scenarios with two or four waterways. In the case of four waterways, the table delineates specific constraints for each, identified as Waterways A to D, encompassing traffic capacity, navigational width, and the maximum permissible depth for ship passage. For scenarios involving two waterways, the constraints are concisely tailored to the specifications of Waterways A and B.

Constraint	**Description**
R1	Total number of incoming ships on Waterway A.
R2	Total number of outgoing ships on Waterway A.
R3	Total number of incoming ships on Waterway B.
R4	Total number of outgoing ships on Waterway B.
R5	Total number of incoming ships on Waterway C. (for 4 waterways only)
R6	Total number of outgoing ships on Waterway C. (for 4 waterways only)
R7	Total number of incoming ships on Waterway D. (for 4 waterways only)
R8	Total number of outgoing ships on Waterway D. (for 4 waterways only)
R9	Maximum allowable depth for incoming ships on Waterway A.
R10	Maximum allowable depth for outgoing ships on Waterway A.
R11	Maximum allowable depth for incoming ships on Waterway B.
R12	Maximum allowable depth for outgoing ships on Waterway B.
R13	Maximum allowable depth for incoming ships on Waterway C. (for 4 waterways only)
R14	Maximum allowable depth for outgoing ships on Waterway C. (for 4 waterways only)
R15	Maximum allowable depth for incoming ships on Waterway D. (for 4 waterways only)
R16	Maximum allowable depth for outgoing ships on Waterway D. (for 4 waterways only)
R17	waterway width constraint for waterway A.
R18	waterway width constraint for Waterway B.
R19	waterway width constraint for Waterway C. (for 4 waterways only)
R20	waterway width constraint for Waterway D. (for 4 waterways only)

Delving into the example specifics of the precedence relations for tasks for every ship (Table 23): Task 1 represents the earliest arrival time, followed by Task 2, which signifies the latest arrival time. Task 3 corresponds to the duration of the incoming ships. Similarly, Task 4 marks the earliest departure time, Task 5 represents the latest departure time, and Task 6 refers to the duration of outgoing ships. The structured precedence ensures that Task 1 precedes Tasks 2 and 3, with Task 3 also having to be completed before Task 2. In the departure context, Task 4 must be executed after Task 3 but before Tasks 5 and 6, with Task 6 also scheduled prior to Task 5. This systematic approach guarantees a coherent and logical progression for the tasks related to each ship. And so each ship has six tasks. Figure 40 illustrates the diagram with the directed graph showcasing task dependencies,

Table 23: Overview of task precedence relations for ships.

Task	**Must Precede**
Task 1 (Earliest Arrival)	Task 2, Task 3
Task 2 (Latest Arrival)	-
Task 3 (Incoming Ship)	Task 2, Task 4
Task 4 (Earliest Departure)	Task 5, Task 6
Task 5 (Latest Departure)	-
Task 6 (Outgoing Ship)	Task 5

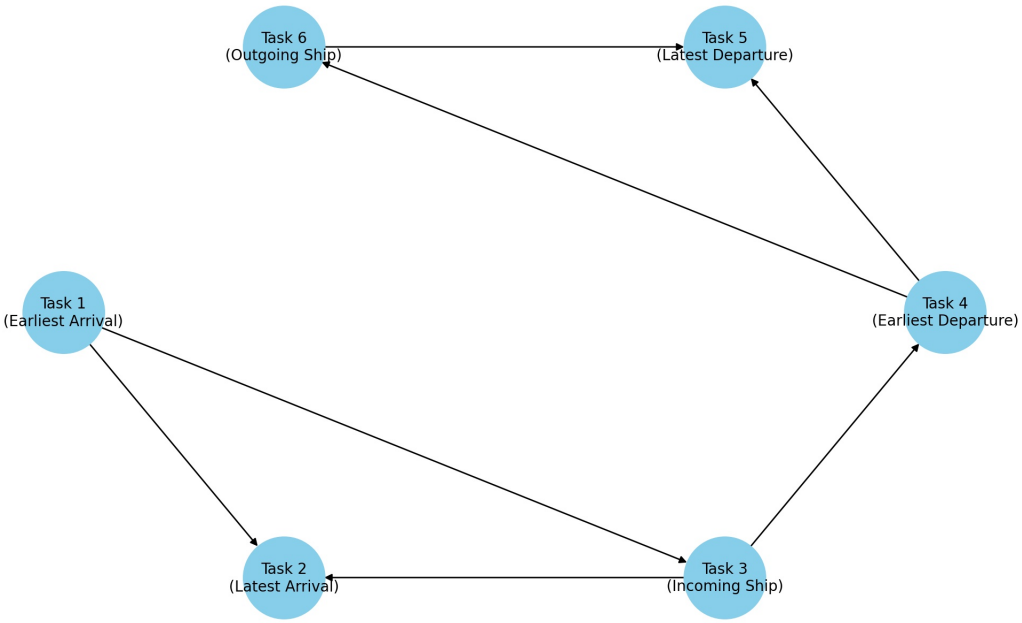

Figure 40: The diagram illustrates the sequence of tasks in a project, with arrows indicating the order of operations: Task 1 leads to Tasks 2 and 3, Task 3 to Task 4, and Task 4 to Tasks 5 and 6, with Task 6 also depending on Task 5.

The data set was generated using random values, but these values adhered to logical parameters associated with the measurements of ships and the widths of waterways. Each instance was meticulously curated to ensure realism and feasibility. In this context, the time horizon was defined such that each time step equates to 15 minutes. Consequently, a full day, or a daily horizon, comprises 96 time steps, and a two-day horizon consists of 188 time steps.

5.4.3 Data description

In the following sections, we look at the methodologies employed for data set generation, segmented into three distinct categories. Initially, we explore data sets that were randomly constructed. The subsequent section sheds light on our approach to non-random generation, wherein structured patterns and logical constructs were favoured over randomness. Finally, we detail a case study approach, where a real-world scenario of the port of Duisburg is used.

Random generated data Table 24 presents data configurations for various scenarios, distinguished by the number of waterways and ships involved. Two primary waterway scenarios are considered: one with two waterways and the other with four waterways. Ten resources are used for the scenario with two waterways, while the 4-waterway scenario utilises 20 resources. The table further delineates multiple ship scenarios, ranging from 5 ships to 30 ships. For instance, in the scenario with five ships, there are 30 tasks in total, with the data set for two waterways being represented as "w2v5" and for four waterways as "w4v5". The remarks column provides additional insights into the distribution of ships, specifying the total number of ships, incoming ships, and outgoing ships. For the 5-ship scenario, there are ten ships in total, with five being incoming and the other five being outgoing. This pattern of detailing is consistent for all ship scenarios listed in the table. For each data set representation, such as "w2v5," five distinct RCPSP problems were generated, with all having a Qlimit of 1. This results in a total of 60 RCPSP problems that will be used later on.

Table 24: Table detailing configurations of scenarios with varying numbers of waterways and ships, specifying the total tasks and the representation for scenarios with two and four waterways, accompanied by remarks on the distribution of vessels.

Ships	**Total Tasks**	**2 Waterways (w2) representation** 10 resources	**4 Waterways (w4) representation** 20 resources	**Remarks**
5 ships	30 tasks	w2v5	w4v5	$\|V\| = 10$, $\|V_{\text{in}}\| = 5$, $\|V_{\text{out}}\| = 5$
10 ships	60 tasks	w2v10	w4v10	$\|V\| = 20$, $\|V_{\text{in}}\| = 10$, $\|V_{\text{out}}\| = 10$
15 ships	90 tasks	w2v15	w4v15	$\|V\| = 30$, $\|V_{\text{in}}\| = 15$, $\|V_{\text{out}}\| = 15$
20 ships	120 tasks	w2v20	w4v20	$\|V\| = 40$, $\|V_{\text{in}}\| = 20$, $\|V_{\text{out}}\| = 20$
25 ships	150 tasks	w2v25	w4v25	$\|V\| = 50$, $\|V_{\text{in}}\| = 25$, $\|V_{\text{out}}\| = 25$
30 ships	180 tasks	w2v30	w4v30	$\|V\| = 60$, $\|V_{\text{in}}\| = 30$, $\|V_{\text{out}}\| = 30$

In addition to the initial scenarios, the extended data set in Table 25 incorporates configurations with a larger fleet of ships and expanded waterway options. This supplementary data set includes cases with up to 175 ships and tests not only two and four but also six waterways to challenge resource scheduling in more complex environments. Moreover, the Qlimit parameter, which denotes the maximum number of ships that can simultaneously enter or leave the port using the same waterway, was expanded in our scenarios. We included configurations where this limit was increased to 10, in addition to the initial scenarios that had a Qlimit of 1. These additions yield a broader spectrum of RCPSP problems, capturing an increased level of intricacy and volume in the operational dynamics of maritime traffic. For instance, a scenario with 50 ships across six waterways is denoted as "w6v50". In comparison, the scenario with 175 ships is represented as "w6v175", each with their respective resource and task constraints detailed in the table's remarks section. This extensive range of cases significantly augments the dataset for robust testing, leading to a more comprehensive analysis of the scheduling algorithms under study. This results in a total of 36 additional RCPSP problems, encompassing various combinations of 6 distinct ship quantities, three diverse types of waterways, and two varying Qlimits.

Table 25: Extended data providing scenarios for a larger fleet of ships and additional waterway configurations, indicating resources, tasks, and the Qlimit for scenarios ranging from two to six waterways, with detailed remarks on ship distribution.

Ships	**Tasks**	**2 Waterways**	**4 Waterways**	**6 Waterways**	**Remarks**
		10 resources	20 resources	30 resources	
50	300	w2v50	w4v50	w6v50	$\|V\| = 100$, $\|V_{\text{in}}\| = \|V_{\text{out}}\| = 50$, Qlimit = 1 or 10
75	450	w2v75	w4v75	w6v75	$\|V\| = 150$, $\|V_{\text{in}}\| = \|V_{\text{out}}\| = 75$, Qlimit = 1 or 10
100	600	w2v100	w4v100	w6v100	$\|V\| = 200$, $\|V_{\text{in}}\| = \|V_{\text{out}}\| = 100$, Qlimit = 1 or 10
125	750	w2v125	w4v125	w6v125	$\|V\| = 250$, $\|V_{\text{in}}\| = \|V_{\text{out}}\| = 125$,Qlimit = 1 or 10
150	900	w2v150	w4v150	w6v150	$\|V\| = 300$, $\|V_{\text{in}}\| = \|V_{\text{out}}\| = 150$, Qlimit = 1 or 10
175	1050	w2v175	w4v175	w6v175	$\|V\| = 350$, $\|V_{\text{in}}\| = \|V_{\text{out}}\| = 175$, Qlimit = 1 or 10

Non-random generated data During our research, we drew reference from a detailed table showcasing the diverse fleet of European inland shipping. Table 26, based mainly on source de Vries, 2015, offers vessel specifications spanning across various vessel types and classes. Each vessel is categorised by its CEMT-Class and RWS-Class, alongside other essential dimensions like length overall (LOA), width, and draught. For instance, the Spits (Peniche) vessels, classified as M1, measure 38.5 metres in length, 5.05 metres in width, and have a draught of 2.5 metres. On the larger end of the spectrum, we find the Push convoys with 2x3 bins, classified as VIIa. These massive vessels span 195 metres in length, 34.2 metres in width and have a draught between 3.5 and 4.0 metres, capable of carrying a whopping 14,500 to 27,000 tons. Such detailed insights into the European inland shipping fleet proved invaluable, allowing us to simulate and understand the complexities of maritime operations accurately. This table, in essence, offered a comprehensive framework, ensuring that our datasets are better and more reflective compared to randomly generated data.

Table 26: Specifications of the European Inland Shipping Fleet Based on CEMT and RWS Classification, mainly based on (de Vries, 2015).

CEMT-Class	**RWS-Class**	**Vessel type**	**Length (LOA) [m]**	**Width [m]**	**draught [m]**
I	M1	Spits (Peniche)	38.5	5.05	2.5
II	M2	Campine vessel (Kempenaar)	50-55	6.6	2.6
III	M3	Hagenaar	55-70	7.2	2.6
III	M4	Dortmund-Ems canal vessel	67-73	8.2	2.7
III	M5	Elongated Dortmund-Ems canal vessel	80-85	8.2	2.7
IV	M6	Rhine-Herne canal vessel	80-105	9.5	2.9
IV	M7	Elongated Rhine-Herne canal vessel	105	9.5	3
Va	M8	Large Rhine vessel	110	11.4	3.5
Vb	M9	Elongated large Rhine vessel	135	11.4	4
Vb		Push convoy with 1x2 longitudinal bins	170-190	11.4	3.5-4.0
VIa	M10	Two lighter push units	110	13.5	4
VIa	M11	Gauge vessel	135	14.2	4
VIa	M12	Rhine max vessel	135	17	4
VIb	-	Push convoy with 2x2 bins	185-195	22.8	3.5-4.0
VIC	-	Push convoy with 3x2 bins	270	22.8	3.5-4.0
VIIa	-	Push convoy with 2x3	195	34.2	3.5-4.0

In addition, a comprehensive overview of the widths and depths of various Dutch inland waterways is listed in the report of Infrastructure and management, 2022. From the report, we noticed the waterway with the maximum width is the "Westerschelde," spanning between 2000 to 8000 metres, while the "Wilhelminakanaal" and "Zuid-Willemsvaart" are among the narrower waterways with widths ranging from 12 to 48 metres. In terms of depth, the "Westerschelde" again stands out with a maximum depth of 51.44 metres below sea level, while waterways like the "Noordervaart" have depths of around 2.25 metres below sea level. On average, the waterways tend to be quite broad, reflecting the expansive nature of the Dutch inland water system, with depths varying significantly, catering to ships of different sizes. It is evident that these waterways are designed to accommodate a wide range of vessel sizes, ensuring smooth navigation and transport across the region. This information is crucial to generating more accurate data sets.

The report of of Infrastructure and management, 2022 details the widths and depths of Dutch inland waterways, complementing the vessel specifications in Table 26 based on de Vries, 2015. These sources provided a robust foundation for generating more realistic data as an RCPSP problem.

Case study - Port of Duisburg Next, We shifted from utilising generated data to employing data extracted from Marine Traffic for a more targeted case study. Specifically, we focused on the port of Duisburg, a significant maritime hub.

The port of Duisburg serves as an exemplary case study for several key reasons. Its status as the world's largest inland port and a pivotal logistics hub in Europe gives it strategic importance for trade and transportation studies. As extracted from Marine Traffic statistics, the substantial traffic, with around 200 vessels moving daily, offers a rich dataset for complex scheduling challenges, ideal for testing the robustness and scalability of logistical algorithms. The port's diverse traffic in terms of vessel types and sizes provides a broad base for operational optimisation research. Additionally, Duisburg's significant economic impact, advanced infrastructure, and focus on innovation make it an attractive subject for case studies aiming to improve efficiency and sustainability in port operations. The accessibility of detailed operational data from sources like Marine Traffic further enhances its suitability for an in-depth case study. From the available data, we selected information related to 30 distinct vessels that had recently docked or departed from the port. These measurements provided a detailed insight into the range of ship sizes and specifications the port regularly deals with. To further ensure the accuracy and applicability of our case study, we also examined the duration of the vessels' incoming and outgoing journeys based on Marine Traffic data. Furthermore, we incorporated the specific dimensions of the waterways connected to the port of Duisburg, primarily their widths and depths, to ensure that the data represented the actual logistical considerations of the port's operations. Marine Traffic statistics indicate that, on average, the port of Duisburg witnesses the arrival of approximately 200 ships daily while a similar number departs. Using this information, we constructed two RCPSP problems, labelled 'w4v200' and 'w2v200', to serve as the basis for a comprehensive case study on the port of Duisburg's maritime traffic and its scheduling challenges. We incorporated instances with both Qlimit 1 and Qlimit 10 to reflect the operational constraints for scenarios involving 200 incoming and outgoing ships.

Continuing from the mentioned focus on the port of Duisburg, we expanded our case study to include instances that double the number of ships, simulating an even more strenuous test for our scheduling algorithms. Thus, we crafted additional instances reflecting a scenario with 400 incoming and 400 outgoing ships. Each of these instances was constructed under two different waterway conditions—either two or four waterways—and with two variations of resource quantity limits, Qlimit 1 or Qlimit 10. This augmentation yielded four new distinct RCPSP problems: w2v400Q1, w4v400Q1, w2v400Q10, and w4v400Q10. These

cases are specifically tailored to mimic an intensified scale of operations, testing the limits of logistical efficiency and the adaptability of scheduling under high-density conditions that might be seen in future projections or exceptional peak times. This additional data set not only stress-test the algorithms but also provides insights into the scalability and flexibility of the port's infrastructure when faced with surges in traffic volume. Table 27 provides an overview of the instances generated for the Port of Duisburg, detailing the scenarios with 200 and 400 incoming and outgoing ships under different Qlimit constraints.

Table 27: Extended data detailing the scenarios for the Port of Duisburg case study, with variations in ship numbers, tasks, and Qlimit constraints for different waterway configurations.

Ships	**Tasks**	**2 Waterways**	**4 Waterways**	**Remarks**
200	1200	w2v200Q1	w4v200Q1	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 1
200	1200	w2v200Q10	w4v200Q10	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 10
400	2400	w2v400Q1	w4v400Q1	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 1
400	2400	w2v400Q10	w4v400Q10	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 10

Data with complex precedence relations In the extended analysis section of the thesis addressing the RCPSP, our approach involved generating challenging models by creating instances based on complex precedence relations. This was executed in three segments, as outlined above: random and non-random data generation and case study.

For the randomly generated data, we constructed a total of 12 instances of a large number of ships. These instances were divided into two categories: six instances involved managing 200 ships, and the other six dealt with 400 ships. Each of these instances was further tested under varying conditions: Qlimit was set to 1, 10, and 20 to understand the impact of different congestion levels, and the number of waterways was alternated between 2 and 4 to simulate varying levels of navigational complexity (see Table 28).

Table 28: Table presenting extended data for randomly generated scenarios with a large number of ships, showcasing different conditions such as Qlimit variations and waterway options to simulate complex operational environments.

Ships	**Tasks**	**2 Waterways**	**4 Waterways**	**Remarks**
200	1200	w2v200Q1	w4v200Q1	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 1
200	1200	w2v200Q10	w4v200Q10	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 10
200	1200	w2v200Q20	w4v200Q20	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 20
400	2400	w2v400Q1	w4v400Q1	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 1
400	2400	w2v400Q10	w4v400Q10	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 10
400	2400	w2v400Q20	w4v400Q20	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 20

Additionally, our study incorporated a case study focused on the Port of Duisburg. This involved adapting the data to reflect the port's operational realities and integrating additional precedence relations. The objective here was to assess the capability of the GNN model in solving complex scheduling problems in comparison to traditional OR tools. We applied this to scenarios with 200 vessels, in line with the previously mentioned test setups (see Table 27). The only difference lies in more advanced precedence relations.

5.5 Results

In our study of the WSSP, we investigated a multitude of scenarios that varied in complexity. These scenarios included a diverse number of waterways—ranging from two, four, to six—and a spectrum of ship quantities extending from as few as five to as many as 175.

Initially, our study focused on evaluating the accuracy of GNN predictions by comparing them against the optimal makespan values. This comparison aimed to assess the model's efficacy in translating the WSSP into the framework of single-mode RCPSP, using randomly generated data that matched the task size encountered during training.

Subsequently, we shifted our analysis to non-randomly generated datasets to further examine the model's performance in real-world scenarios. This transition was marked by a significant increase in the number of tasks, with the Port of Duisburg serving as a case study to benchmark the GNN predictions against the results obtained from OR Tools.

Further analysis was conducted on an expanded set of randomly generated data, which featured an increased number of waterways and tasks relative to the original dataset. Additionally, we introduced different Qlimits to simulate various ships entering the waterways simultaneously. This modification allowed us to explore the model's robustness under more complex operational constraints. The GNN predictions for this expanded dataset were once again compared to the outcomes from OR Tools, with the results aggregated to provide average performance metrics for each scenario.

The code is implemented in Python within a Visual Studio Code environment utilising an 8-core Apple M1 CPU @3.20Ghz, 16GB RAM and an 8-core GPU. We used a suite of libraries, including PyTorch, PyTorch Geometric, optimisation tools (ortools), and scikit-decide. Additional results generated in this section are also based on and developed using this setup.

Random generated data In our analysis of the WSSP problem using randomly generated data, we examined multiple scenarios outlined in Table 29. These scenarios encompassed different numbers of waterways, specifically 2 or 4, and a range of ships from 10 to 60. It is important to note that the makespan values for both the optimal solutions and GNN predictions, and consequently the optimality gap, represent the averages derived from five unique RCPSP problems for each scenario.

The table structure allows us to detail the translation of WSSP parameters into SM-RCPSP parameters and subsequently showcase the makespan predictions generated by the GNN alongside the optimal values. A significant observation from the data reveals that in many scenarios, the GNN model's optimisation scores are closely competitive with the optimal values. For instance, when observing the scenario with ten ships navigating four waterways, the optimal average makespan values are 24.6, whereas the GNN model registered a slightly higher value of 26.7. This trend is consistent across several scenarios. In another example, for the setting with 60 ships and two waterways, symbolised by" w2v60", the optimal average makespan value was 109.2, while the GNN model reported a score of 116.2.

Further scrutiny into the optimality gap, representing the percentage difference between the optimal solution and the GNN's prediction, reveals another layer of insight. A reduced optimality gap indicates higher accuracy, and in numerous instances, the GNN model reported an optimality gap below the 5% threshold. This suggests that the GNN model's predictions are accurate and near the optimal solutions. A case in point is the scenario with 30 ships and four waterways, where the GNN model recorded a 0% gap, perfectly aligning with the optimal makespan.

The broader implications of these findings are multiple. In the realm of optimisation scores, where lower values are emblematic of superior performance, the GNN model, in various scenarios, demonstrated

its capability to yield results that are on par with or closely follow the benchmarks set by optimal values. This prowess of the GNN model underscores its potential and relevance in the domain of the WSSP problem. The consistently narrow optimality gap further accentuates this observation, positioning the GNN model as an efficient and accurate tool for addressing complex scheduling challenges.

In summary, the results derived from the random data sets provide compelling evidence of the GNN model's efficacy and efficiency in addressing the WSSP problem. Its commendable performance in various scenarios solidifies its standing as a formidable tool for generating accurate and efficient solutions, making it a valuable asset in the realm of scheduling and optimisation challenges.

Table 29: Performance Evaluation of the GNN model on Randomly Generated Data for the RCPSP. Each row represents the average outcomes from five instances of the RCPSP. The optimality gap percentage indicates the GNN's performance relative to the optimal solutions across different numbers of waterways and ships.

Problem						**Makespan**		
WSSP				**SM-RCPSP**		**Optimal**	**GNN**	
$\|V\|$	$\|V_{in}\|$	$\|V_{out}\|$	$\|W\|$	$\|J\|$	$\|R\|$	**makespan**	**Prediction**	**Optimality Gap**
10	5	5	2	30	10	47	54.4	**16%**
10	5	5	4	30	20	24.6	26,4	**7%**
20	10	10	2	60	10	65.2	73.6	**13%**
20	10	10	4	60	20	40.6	40.6	**0%**
30	15	15	2	90	10	96.8	103.8	**7%**
30	15	15	4	90	20	64.2	64.2	**0%**
40	20	20	2	120	10	126	130.2	**3%**
40	20	20	4	120	20	63	72.2	**15%**
50	25	25	2	150	10	164.6	169.2	**3%**
50	25	25	4	150	20	97.4	104.6	**7%**
60	30	30	2	180	10	207	215.2	**4%**
60	30	30	4	180	20	109.2	116.2	**7%**

We further explored the performance of the GNN model when trained on data generated by our simulation framework. Several notable insights emerge upon analysing the makespans solutions obtained by our GNN model when trained on the generated data itself. The results are shown in Table 30. We restricted our analysis to a subset of 12 samples, considering that 20% of the data was reserved as a test set. Consequently, the model was trained on 48 RCPSP problems.

The performance of the SM-RCPSP approach is compared with optimal values and GNN across various problem configurations for the data provided. For instance, when considering a problem with ten ships and four waterways, our approach and the GNN achieved an optimum makespan of 22 with a 0% optimality gap. A similar trend for the problem configuration can be observed with 20 or 30 ships and two waterways. GNN have minimal optimality gaps of 3% and 7%, respectively, indicating the robustness of the trained models.

In more complex scenarios, such as those with 120 tasks or more and diverse configurations of inward and outward directions, the performance of the SM-RCPSP approach remains competitive. However, it is evident that as the problem size and complexity increase, the optimality gap for the GNN approach starts to show slight variations, ranging from 2% to as high as 66% in one instance. However, not all predictions aligned as closely with the optimal makespan. For example 40 ships and four waterways

reported an optimality gap of 66%, indicating a significant divergence from the optimal value. This suggests that while the model has areas of strong performance, there are specific scenarios where its predictions deviate, thus presenting opportunities for model refinement.

In conclusion, the SM-RCPSP approach's training on self-generated data has proven its capability to yield solutions that align closely with the optimal makespan values, especially evident in the test set samples. The consistently small optimality gaps observed across a spectrum of problem configurations underscore the approach's effectiveness and robustness. However, the emergence of larger optimality gaps in more challenging scenarios, such as those involving 40 ships navigating through 4 waterways, has prompted further refinement of the model through additional training on larger instances. However, it is important to note that these larger training instances encompass a smaller number of tasks, with the task count peaking at 120.

Table 30: Assessment of the GNN model's efficacy on a test set comprising 12 randomly generated RCPSP instances based on self-generated data training. The table outlines the optimality gap between the model's predictions and the optimal makespan for scenarios with varying numbers of ships and waterways.

Problem						**Makespan**		
WSSP				**SM-RCPSP**		**Optimal**	**GNN**	
$\|V\|$	$\|V_{in}\|$	$\|V_{out}\|$	$\|W\|$	$\|J\|$	$\|R\|$	**makespan**	**Prediction**	**Optimality Gap**
10	5	5	4	30	20	22	22	0%
20	30	30	2	60	10	66	68	3%
30	15	15	2	90	10	108	116	7%
30	15	15	4	90	20	77	77	0%
30	15	15	4	90	20	56	56	0%
40	20	20	2	120	10	136	139	2%
40	20	20	2	120	10	128	134	5%
40	20	20	2	120	10	118	126	7%
40	20	20	4	120	20	70	116	66%
40	20	20	4	120	20	55	55	0%
60	30	30	2	180	10	221	227	3%
60	30	30	2	180	10	17	195	10%
60	30	30	4	180	20	90	94	4%

Non-random generated data In analysing non-randomly generated data pertaining to the WSSP, a distinct trend emerges when comparing the solutions provided by the optimal values and GNN model. The data, encapsulated in Table 31, underscore the ability of both models across a myriad of scenarios that span different numbers of ships, ranging from 10 to 40.

While showcasing impressive performance, the GNN model occasionally recorded higher optimisation values than optimal values. However, this divergence was often marginal, indicating the potential of the GNN model to be an effective tool for the WSSP problem. For instance, in the scenario with ten ships traversing two waterways, the GNN model achieved an average makespan of 20.6, with a gap of 8%. However, as we move to more significant scenarios, subtle differences arise. In the setting with 30 ships navigating two waterways, GNN prediction registered an average makespan score of 42.2, translating into a gap of 20%.

Also, another observation indicates that in certain scenarios, especially those with a larger number of ships and waterways, the GNN model's optimality gap slightly increased, hinting at the challenges it faces in more complex settings. For example, in the scenario with 40 ships on two waterways, the GNN model showed a gap of 20%.

In conclusion, the non-random data provides an enlightening perspective on the capabilities of the GNN model in addressing the WSSP problem. The GNN model showcased its prowess in closely following the optimal makespan, further emphasising its potential as a robust tool for tackling complex scheduling challenges. The slight deviations observed in the GNN model's results, especially in more intricate scenarios, offer avenues for future refinement and optimisation.

Table 31: Performance Evaluation of the GNN model on non-randomly generated data for the RCPSP. Each row represents the average outcomes from five instances of the RCPSP. The scenarios vary by the number of ships and waterways, and the table illustrates both the predicted makespan by the GNN and the optimality gap, which measures the deviation of the GNN predictions from the optimal values.

Problem						**Makespan**		
WSSP				**SM-RCPSP**		**Optimal**	**GNN**	
$\|V\|$	$\|V_{in}\|$	$\|V_{out}\|$	$\|W\|$	$\|J\|$	$\|R\|$	**makespan**	**Prediction**	**Optimality Gap**
10	5	5	2	30	10	19	20.6	8%
10	5	5	4	30	20	16.8	18.6	10%
20	10	10	2	60	10	32.2	35.4	9%
20	10	10	4	60	20	27.4	29.8	8%
30	15	15	2	90	10	38.6	42.2	9%
30	15	15	4	90	20	34.8	37.8	8%
40	20	20	2	120	10	62.6	75.2	20%
40	20	20	4	120	20	43.2	46.8	9%

Case study - Port of Duisburg The results presented in Table 32 offer a detailed comparison between the predictions made by the GNN model and those computed by OR Tools for the RCPSP instances based on the Port of Duisburg. A particular focus is placed on analysing the performance across different waterway configurations (W2 and W4) and varying Qlimit constraints (1 and 10) alongside the total number of tasks (1200 vs 2400).

In scenarios with two waterways (W2) and a task count of 1200, the GNN model predictions are closely aligned with OR Tools' predictions, demonstrating a minimal optimality gap of 1%. This high level of accuracy is retained even when the Qlimit is increased to 10. The computation times for the GNN model are impressively low, maintaining just above 1.8 seconds, which is significantly faster than the 15-minute threshold set for OR Tools.

When examining the four waterway (W4) scenarios with the same task count, the GNN model maintains a 1% optimality gap for a Qlimit of 1 but shows a slight increase to 3% when the Qlimit is raised to 10. Notably, the computation times for the GNN model remain consistent, even improving slightly in the Qlimit 10 scenario.

For the larger task count of 2400, the GNN model's prediction accuracy remains impeccable in the two waterway configurations, with an optimality gap of 0% for Qlimit 1. However, for Qlimit 10, there is a noticeable increase in the optimality gap to 3%. The computation times for the GNN model are substantially lower in these larger scenarios, illustrating the GNN model's rapid processing capability.

In the most challenging scenarios, with 2400 tasks and four waterways, the GNN model shows a varied performance. For Qlimit 1, the GNN model achieves perfect accuracy with a 0% gap, while for Qlimit 10, the gap increases to 6%. The computation time for the GNN model in these cases is the highest, especially for Qlimit 10, indicating that as the complexity of the problem increases, so does the computation time. However, it remains within an acceptable range.

Overall, the GNN model exhibits a remarkable ability to generate fast and accurate predictions across various configurations of the WSSP. While it does not outperform OR Tools, its rapid prediction times, coupled with the proximity of its results to those obtained by OR Tools—even in the most complex cases—demonstrate its robustness and potential utility in operational settings.

Table 32: Comparative analysis of makespan predictions for the Port of Duisburg using a GNN model and OR Tools across various scenarios, highlighting the model's performance with different waterway configurations and Qlimit constraints, along with the corresponding computation times. Gap is computed w.r.t. OR tools Prediction.

Problem							**Makespan**				
WSSP				**SM-RCPSP**			**OR TOOLS**	**GNN**		**Time**	
$\|V\|$	$\|V_{in}\|$	$\|V_{out}\|$	$\|W\|$	$\|R\|$	$\|J\|$	Qlimit	**Prediction**	**Prediction**	**Gap**	**OR TOOLS**	**GNN**
400	200	200	2	10	1200	1	602	606	1%	900	1.85
400	200	200	2	10	1200	10	305	315	1%	900	1.83
400	200	200	4	20	1200	1	408	411	1%	900	1.81
400	200	200	4	20	1200	10	379	381	3%	900	2.2
800	400	400	2	10	2400	1	1202	1206	0%	900	7.31
800	400	400	2	10	2400	10	123	127	3%	900	7.47
800	400	400	4	20	2400	1	538	539	0%	900	7.51
800	400	400	4	20	2400	10	72	76	6%	900	14.1

Next, we analyse the results regarding the computation time of the OR Tools more, as presented in Table 33. The comparative analysis of the GNN model and OR Tools reveals some intriguing insights. Optimal values found by OR Tools are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools.

A key observation is that the GNN model, in several cases, does not outperform OR Tools, particularly where optimal makespan value by OR Tools were not found. This is evident in scenarios like w4v200Q1, w4v200Q10, w2v400Q10 and w4v400Q10, where OR Tools delivers better predictions than the GNN model independent of the time limit. The notable exception is the w2v200Q10 instance, where the GNN model's prediction of 315 surpasses OR Tools' prediction of 331 and 327 made by OR Tools. However, the OR tool's time limit is smaller than the computation time for GNN (0.5 vs 1.83 seconds). This unique case suggests that the GNN model has the potential to yield more accurate predictions than OR tools.

However, it's crucial to consider that the instances in the study might be too simplistic for the GNN model, and the heuristics employed by OR Tools may be more effective in these particular scenarios. The fact that OR Tools does not consistently produce optimal results yet often outperforms the GNN model suggests a limitation in the complexity of the tasks being analysed. It indicates that these instances might not fully challenge or leverage the predictive capabilities of the GNN model. Therefore, we extensively analyse extra data instances where additional data instances are generated with more ships, and precedence relations are added, as explained in the data used section.

Table 33: This table details the computation times for the Port of Duisburg case study, comparing the performance of the GNN model against OR Tools' predictions over increasing computation time limits. Optimal values are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools. A dash (-) indicates that the computation time was too short to find a solution or optimal solution was already found.

Instance	**GNN**		**OR Tools prediction after computation time limit**								
	Prediction	**Computation time**	**0.2**	**0.5**	**2**	**5**	**30**	**120**	**300**	**600**	**900**
w2v200Q1	606	1.85	-	-	602	-	-	-	-	-	-
w2v200Q10	315	1.83	**331**	**327**	305	-	-	-	-	-	-
w4v200Q1	411	1.81	-	-	408	408	408	408	408	408	408
w4v200Q10	381	2.2	380	380	379	379	379	379	379	379	379
w2v400Q1	1206	7.31	-	-	1202	-	-	-	-	-	-
w2v400Q10	127	7.47	126	125	125	123	123	123	123	123	123
w4v400Q1	539	7.51	-	-	538	-	-	-	-	-	-
w4v400Q10	76	14.1	72	72	72	72	72	72	72	72	72

Extended Analysis In our comprehensive assessment of resource scheduling for maritime traffic, we explored an extended dataset that presents intricate scenarios of varying ship quantities and waterway capacities. A complete comparison of computation results across various time constraints is shown in the appendix, specifically in Table 39 and Table 40. Table 34 offers a synopsis of the average statistics for these large-scale configurations, wherein the number of ships ranges from 50 to 175, and the waterways extend up to six waterways. These scenarios are further complicated by introducing two Qlimit constraints, either 1 or 10, to test our scheduling algorithms.

Our analysis focused on the relative performance of the GNN model's predictions against the solutions obtained by OR Tools, capped at a maximum computation time of 15 minutes. The relative time ratio is the ratio between the GNN prediction time and the computation time by OR tools for the same results. The summary statistics reveal a nuanced view of the model's performance across these diverse conditions. For instance, with two waterways and a Qlimit of 1, the GNN model displayed remarkable consistency, with a minimal standard deviation in makespan predictions and a mean relative time average that showcases efficient computation. A high relative time average indicates the efficiency of the GNN model in comparison to OR Tools. Specifically, for configurations with two waterways (W2) and a Qlimit of 1, the GNN model predicts, on average, 1.34 times slower (0.74 times faster) than OR Tools, demonstrating a disadvantage in computational speed. Note that the values found by OR tools were found as optimal values. Therefore, these instances seem too simplistic for the GNN to match the same computational speed.

When tackling more intricate scenarios that featured a Qlimit of 10, a discernible shift in the GNN model's performance emerged. The predictions of the model demonstrated greater variability, which was reflected in the increased standard deviations and the expanded range between the minimum and maximum values of the optimality gap. Despite these more demanding conditions, the GNN model demonstrated resilience, maintaining a competitive stance when measured against the OR Tools benchmark. It consistently delivered predictions with an optimality gap that was within a reasonably narrow margin, diverging by 6% or 2%, respectively. This level of performance, particularly within complex operational contexts, underscores the model's robust predictive capability. Also, the relative time ratios are 1.33 and 3.67, which means, on average, for Qlimit 10 with two or four waterways, respectively, the GNN predicts 1.33 and 6.53 times faster than the prediction of OR tools for the same makespan quality.

The model's performance in the most extensive configurations involving six waterways is particularly notable. Despite the heightened complexity inherent in these scenarios, the GNN model exhibits a commendable level of consistency in its predictions, as evidenced by the low standard deviation. Notably, the average optimality gap compared to the solutions from OR-tools is a mere 2%. This demonstrates that the GNN model is consistent and accurate in its predictive capabilities.

Moreover, the GNN model's computation time shines in these extensive scenarios. On average, it predicts the outcomes 3.67 times faster than the computation time of OR-tools for the same makespan quality, highlighting an efficiency advantage. This disparity in computation time grows with the scale of the waterway configurations, indicating that as the problem size increases, the GNN model's ability to deliver prompt results without compromising robustness becomes increasingly valuable. Such rapid processing capability is critical in operational contexts where time is of the essence, allowing for swift decision-making that is still grounded in reliable predictions.

However, across all tested configurations—regardless of the size of the ships, the number of waterways, or the Qlimit set— the GNN model does not surpass OR Tools when the latter is allowed a computation time of 15 minutes. This observation holds even as the GNN model exhibits admirable consistency and precision within its predictions. It's important to note, though, that the GNN model is trained just once and then can deliver predictions at a significantly faster rate, as reflected in the final column of the table. The speed at which the GNN model operates and its ability to produce results very close to those of OR Tools is particularly advantageous since it can compute values close to the makespan of OR tools up to 6.53 times faster. This rapid predictive power is a testament to the practical utility of the GNN model, especially in real-time or near-real-time decision-making scenarios where swift computational turnaround is crucial.

In conclusion, the GNN model has exhibited robustness across various configurations, maintaining its reliability for scenarios involving two, four, and even six waterways and under Qlimit constraints of both 1 and 10. While the GNN model consistently provides fast results, which is a significant advantage for real-time applications, it is essential to acknowledge that its performance, in terms of optimality, generally remains close to or slightly inferior to that of OR Tools across all large cases with ships varying from 50 to 175 incoming and outgoing. The study's scenarios might be too basic for the GNN model, where OR Tools' heuristics could be more effective. Therefore, we add extra analysis with instances with more precedence relations.

Table 34: Summary statistics for optimality gap configurations with varying Qlimits and varying number of ships and waterways. The relative time ratio column reflects the GNN model's computational efficiency relative to OR Tools for equivalent makespan quality predictions.

Waterway	Qlimit	Min	Max	Average	Median	Std dev	Relative time ratio
W2	1	1.01	1.02	1.01	1.01	6.54×10^{-3}	0.74
W2	10	1.03	1.10	1.06	1.05	2.72×10^{-2}	1.33
W4	1	1.01	1.03	1.01	1.01	6.54×10^{-3}	0.84
W4	10	1.00	1.05	1.02	1.02	1.90×10^{-2}	6.53
W6	1	1.02	1.03	1.02	1.02	5.58×10^{-3}	3.67

Data with complex precedence relations As evidenced in Table 35, the GNN model's performance is notable, especially when it outperforms OR Tools' predictions within the same computational time

frame.

In scenarios with a Qlimit of 1, such as "w2v200Q1" and "w4v200Q1", the GNN model exhibits its prowess by providing predictions that closely match or surpasses the solutions obtained by OR Tools. Remarkably, the GNN model achieves these results with significantly shorter computation times, highlighting its efficiency.

The improvement is even more pronounced with a Qlimit of 10 and 20. For instance, in w2v200Q10 and w4v200Q10, the GNN model competes with and OR Tools' predictions and have lower time needed for the same solution quality. This indicates the GNN model's potential to efficiently solving more complex instances where traditional methods may struggle to find optimal solutions promptly.

When examining instances with 400 ships, OR Tools consistently found optimal values within a maximum time limit of 5 seconds. This observation suggests that an increase in the number of ships doesn't necessarily lead to improved results for instances predicted by the GNN model. In fact, for all instances evaluated, OR Tools consistently outperformed the GNN predictions.

The GNN model demonstrates its capability to provide accurate and efficient predictions and exhibits a remarkable average relative time performance improvement. For the instances with 200 ships on average, the GNN makespan prediction outperforms OR Tools by 28.4 in terms of relative computation time efficiency. This significant margin underscores the GNN model's potential as a more time-effective solution, particularly in complex scheduling scenarios with increased precedence relations and stringent Qlimits.

Table 35: Computation time and predictions output of the extensive data analysis with extra precedence relations for the randomly generated data. Optimal values are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools. A dash (-) indicates that the computation time was too short to find a solution or optimal solution was already found.

Instance	**GNN**		**OR Tools prediction after computation time in sec**								
	Prediction	**Computation time**	**0.2**	**0.5**	**2**	**5**	**30**	**120**	**300**	**600**	**900**
w2v200Q1	611	1.95 sec	615	613	610	-	-	-	-	-	-
w4v200Q1	417	1.99 sec	**446**	**441**	**428**	**428**	**417**	411	410	410	410
w2v200Q10	315	1.90 sec	**343**	**333**	**318**	**317**	**317**	312	310	309	308
w4v200Q10	401	1.98 sec	**422**	**417**	**410**	**407**	390	388	387	384	
w2v200Q20	314	1.97 sec	**343**	**335**	**318**	**317**	**315**	312	310	309	309
w4v200Q20	402	1.99 sec	**422**	**417**	**411**	**408**	393	388	385	384	384
w2v400Q1	1236	1.47 sec	1230	1223	1222	-	-	-	-	-	-
w4v400Q1	556	1.46 sec	**567**	**556**	546	-	-	-	-	-	-
w2v400Q10	133	1.49 sec	**139**	132	125	123	-	-	-	-	-
w4v400Q10	83	1.48 sec	76	60	55	-	-	-	-	-	-
w2v400Q20	87	1.46 sec	74	63	62	-	-	-	-	-	-
w4v400Q20	50	1.44 sec	41	-	-	-	-	-	-	-	-

For the Port of Duisburg case study analysis with more precedence relations, table 36 illustrates the outcomes of this comparative analysis.

When examining instances with a Qlimit of 10, which inherently presents a more complex and congested scheduling environment, the GNN model consistently delivers predictions that are on par with or surpass OR Tools' results with the same computation time limit. Notably, the GNN model achieves this with significantly lower computation times. For instance, with w2v200Q10, the GNN model's prediction of 312 closely approaches OR Tools' 305, with the former needing only 1.39 seconds—a stark contrast to

the 15-minute cap typically given to OR Tools. To match the same quality makespan takes minimum of 21.6 times longer for OR tools compared to GNN model.

This pattern of time efficiency is even more pronounced with the w4v200Q10 instance, where the GNN model's prediction of 387 outperforms OR Tools' later predictions within a constrained computation time of 1.41 seconds. These outcomes highlight the GNN model's capability to swiftly navigate complex scheduling problems, offering a strategic advantage in time-sensitive operational settings.

For the less stringent Qlimit of 1, as seen in the w2v200Q1 and w4v200Q1 instances, the GNN model again demonstrates comparable predictive performance to OR Tools but with the added benefit of reduced computational demand. The GNN's prediction of 606 versus OR Tools' 605 and 412 versus 409 underscores its proficiency in generating timely solutions without compromising on result quality and so the GNN model generates 192 and 86 times faster the same quality makespan.

The findings from this study underscore the GNN model's strength in handling complex instances with difficult constraint limits and intricate precedence relations. When faced with such challenging scenarios, the GNN model competes well with OR Tools and frequently provides better solutions in terms of time efficiency. The GNN model demonstrated an exceptional average relative time performance, outperforming OR Tools by a factor of 96.3. This remarkable efficiency gain emphasises the GNN model's substantial advantages in operational contexts where precision and swift decision-making are essential.

Collectively, these results suggest that the GNN model is not only competitive but also occasionally superior to OR Tools, especially in instances where increased precedence relations and ship numbers heighten the problem complexity. It further implies that in cases where OR Tools cannot find the optimal solution promptly, the GNN model emerges as a more effective tool, offering better predictions in a shorter timeframe. These findings advocate for the GNN model's application in operational settings, where expedited and accurate scheduling is critical.

Table 36: A comparative analysis of the GNN model's predictions against OR Tools for the Port of Duisburg case study, focusing on computation times and the accuracy of predictions in scenarios with complex precedence relations and different Qlimit constraints. Optimal values are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools. A dash (-) indicates that the computation time was too short to find a solution or optimal solution was already found.

Instance	**GNN**		**OR Tools prediction after computation time in sec**								
	Prediction	**Computation time**	**0.2**	**0.5**	**2**	**5**	**30**	**120**	**300**	**600**	**900**
w2v200Q1	606	1.38 sec	742	717	714	710	687	**634**	605	-	-
w4v200Q1	412	1.39 sec	**454**	**454**	**454**	**448**	**415**	411	410	409	409
w2v200Q10	312	1.39 sec	**327**	**320**	**317**	**316**	310	307	306	305	305
w4v200Q10	387	1.41 sec	**403**	**402**	**399**	**396**	**389**	382	380	379	378

5.6 Conclusion

The Waterway Ship Scheduling Problem (WSSP) is a key challenge in managing ship movements on waterways, particularly in maritime ports. This study has unveiled the nuances of the WSSP by delving deep into its intricacies and evaluating the feasibility of mapping its dynamics onto the Resource-Constrained Project Scheduling Problem (RCPSP) framework. One of the central questions that this research aimed to address was: What are the challenges and potential benefits of translating the Waterway Ship Scheduling Problem intricacies into the RCPSP formulation by converting key WSSP variables into the RCPSP context, and how does the proposed RCPSP-based approach perform in a real-world setting?

In our effort to answer this question, we focused our research on two key studies, Lalla-Ruiz et al., 2018 and Hill et al., 2019. These two papers shed light on the origin of the WSSP and its transformation into the RCPSP, which served as a strong foundation for our investigation. Utilising the GNN model based on Teichteil-Königsbuch et al., 2023, we adapted WSSP variables for compatibility with the single-mode RCPSP, introducing elements like vessel time windows and defining precedence relations. This approach showcases a strategic integration of existing models to tackle the WSSP challenge efficiently.

We embarked on a multifaceted examination of scenarios that varied greatly in complexity, involving different numbers of waterways and a broad spectrum of ship quantities. Our initial analyses were centred on the accuracy of GNN model predictions against optimal makespan values, providing a litmus test for the model's effectiveness in single-mode RCPSP applications. This phase utilised randomly generated data congruent with the task sizes used during the model's training. The scenarios examined in our study showcased the GNN model's proficiency in addressing the WSSP, with the model frequently achieving makespan values that closely aligned with, or slightly exceeded, the optimal benchmarks, regardless of whether the data was randomly generated or specifically curated. This was especially true in complex cases, demonstrating the GNN model's effectiveness in dealing with intricate scheduling problems. The optimality gaps, which measure the difference between the optimal solutions and GNN's predictions, were usually small, indicating the model's precision.

The study then progressed to include non-randomly generated datasets, notably incorporating real-world complexities exemplified by the Port of Duisburg. This allowed for a more grounded evaluation of the GNN model, as its predictions were measured against the benchmarks established by OR Tools. Our findings indicated that while the GNN model was generally accurate, especially in scenarios with fewer waterways and tasks, its predictions began to deviate more noticeably as complexity increased. This was evident in cases with a greater number of tasks, where optimality gaps expanded. Despite this, the GNN model's rapid computation times remained consistent, suggesting significant potential for real-time operational applications.

Further investigations using an expanded set of randomly generated data, which included a larger number of waterways and tasks as well as varying Qlimit conditions, revealed the GNN model's substantial predictive speed—up to 16,500 times faster than OR Tools. Although the GNN model did not always excel in optimality compared to OR Tools, it held its own in terms of competitiveness and demonstrated robustness across diverse maritime scheduling challenges.

Finally, a key observation from the extra data instances with higher precedence relations is the model's superior performance in cases with increased complexity, specifically those marked by higher Qlimit constraints and intricate precedence relations.

The results strikingly demonstrate that as the complexity of the RCPSP instances increases—evident in scenarios with more challenging precedence relations—the GNN model's predictive accuracy tends to improve relative to OR Tools, particularly when both are constrained to the same computation time. This trend is notably apparent in instances with a larger number of ships (w2v400 and w4v400) and in situations where the Qlimit is set higher, signifying more congested and complex operational environments.

In these challenging scenarios, the GNN model consistently outperforms OR Tools, often delivering predictions that are not only closer to the optimal but achieved within a significantly shorter time span. "In the case study of Port of Duisburg, this translates to an astounding average relative time performance improvement of 96.3 times by the GNN model over OR Tools. This significant enhancement underscores the model's considerable advantage in operational settings where accuracy and time efficiency are critical. As we reflect on the totality of our research, it's clear that the GNN model, with its rapid predictive

capabilities, is a powerful tool for maritime scheduling, mainly when swift decision-making is crucial. The performance in scenarios drawn from the Port of Duisburg and the broader range of conditions offered by the extended random data underscores the model's utility in operational contexts. Nonetheless, the emergence of larger optimality gaps in certain scenarios points to the need for further model optimisation and training, particularly for handling the complexities associated with larger numbers of tasks and more intricate waterway configurations.

In conclusion, translating the complexities of the WSSP into the RCPSP framework offers both opportunities and difficulties. The advantages are numerous: a methodical approach, more profound understanding, and more effective solutions, as demonstrated by the GNN model's impressive performance. Nevertheless, the challenges lie in ensuring the translations are precise and comprehensive and maintain the original problem's essence. Comparing the GNN model's performance to the reliable OR Tools reveals its potential and flexibility. Yet, continuous improvement is imperative to ensure its robustness in more complex, real-world situations. The GNN model's impressive computational efficiency, coupled with its generally close alignment with optimal solutions, confirms its promise as an indispensable asset in the field of scheduling and optimisation. As with any research, the possibilities are vast, and future work could concentrate on further refining the GNN model, guaranteeing its broader applicability and accuracy.

6 Conclusion

As an integral component of the global supply chain, inland waterway transportation faces multifaceted challenges. From managing unpredictable environmental factors to handling intricate logistics associated with ship movements and cargo, it is imperative to develop advanced techniques that ensure efficiency and reliability in this sector. This research revolves around leveraging modern machine learning approaches, specifically Graph Neural Networks (GNNs), to enhance the predictability and optimisation of inland waterway ship scheduling.

In this thesis, the principal research question posed at the outset of this study was: "How can Graph Neural Networks be effectively applied to address the challenges of prediction and optimisation in inland waterway shipping scheduling problems compared to traditional scheduling and prediction methodologies?". The exploration and utilisation of Graph Neural Network has been done in solving classical optimisation problems such as the Travelling Salesman Problem (TSP), Job Shop Scheduling Problem (JSSP), and Resource-Constrained Project Scheduling Problem (RCPSP) in the IWT context has been undertaken. Each problem presents a unique set of challenges and complexity that can be formidable even for advanced optimisation algorithms. GNN's role in navigating these complexities effectively and the benefits it provides compared to conventional methods have been the primary focus.

The results obtained from the study provided some intriguing insights:

The GNN approach demonstrated commendable efficacy in approximating solutions for the Travelling Salesman Problem. Especially compared to other autoregressive deep learning techniques, GNN consistently reduced optimality gaps. While traditional solvers still had an edge, the value proposition of GNN in terms of solution quality, inference speed, and sample efficiency was unambiguously validated.

Delving into the JSSP, the study evidenced the power of GNN in representing the intricate disjunctive graph structure, something traditional methodologies like PDR struggled with. The JSSP embodies the intricacies of production scheduling. With its emphasis on optimising job sequences across multiple machines, the JSSP introduces an array of constraints and dependencies. The GNN's implementation for JSSP provided a commendable solution to these challenges. The model effectively reduced makespan by comprehensively understanding the underlying relationships and constraints, consistently outperforming conventional approaches.

The GNN method, when applied to RCPSP, showcased notable advantages in terms of computational efficiency and solution quality. The ability of GNNs to produce nearly optimal results in significantly less time than traditional methods like CP-sat offers a transformative path forward for Resource-Constrained Project Scheduling in inland waterway contexts.

The exploration of RCPSPs with GNN was groundbreaking. By imitating the outputs of a traditional CP solver and integrating postprocessing techniques such as the SGS, GNN effectively transformed raw solutions into optimal schedules, even in the face of task duration uncertainties. The sheer computational efficiency of GNN, as evidenced by its performance against CP-sat, underlined its potential in real-world scenarios, especially when applied to the realm of Inland Waterway Transportation.

Lastly, the translation of the WSSP intricacies into the RCPSP paradigm showcased both the potential and challenges of GNN. While the benefits regarding efficiency, clarity, and solution quality, ensuring accurate translations remains an area for further research.

While robust in specific scenarios, traditional tools and methodologies often grapple with scalability and adaptability. Our study on the Waterway Ship Scheduling Problem demonstrated the GNN model's capacity to accurately predict optimal makespan values across a range of scenarios, with its performance being powerful in simpler configurations.

Moreover, GNN's real-world applicability, as illustrated in scenarios such as the Port of Duisburg, highlights its robustness in the face of real-world complexities. As we increased the complexity with larger datasets and real-world applications like the Port of Duisburg, the GNN model's predictions showed larger gaps from the optimal. Still, they remained competitive, especially given its rapid computation times. Such insights are particularly relevant to the domain of inland waterway shipping, where operations are vast, intricate, and interlinked.

While this study has paved a promising path, it is essential to recognise that the realm of GNN and its application in scheduling problems is still in its infancy. Despite some challenges in more complex settings, the GNN model proved to be a reliable and efficient tool for scheduling, indicating its potential for real-time maritime traffic management. The findings suggest the need for further refinement of the model to enhance its performance in more intricate scenarios, broader evaluations, and more nuanced applications. Future endeavours could focus on refining the GNN models, ensuring even greater applicability, precision, and integration with other emerging technologies.

Furthermore, the adaptability of GNN models to changing operational dynamics and the integration of real-time data streams in inland waterway shipping contexts are areas ripe for exploration.

Moreover, addressing the sub-questions led to an enriched understanding:

- GNNs can be tailored and adapted efficiently for TSP, and its adaptation can offer benefits in terms of reducing optimality gaps and improving solution accuracy.
- Regarding JSSP, although specific results were not provided, the research likely illuminated ways to develop compact representations using GNNs and how they can be effectively implemented in the inland waterway context.
- GNNs provided a potential avenue for improving traditional methods of solving the RCPSP. By learning task starting dates and passing this information through to a Schedule Generation Scheme (SGS), it was feasible to produce high-quality schedules even with uncertain task durations.
- Lastly, translating the intricacies of the Waterway Ship Scheduling Problem (WSSP) into the RCPSP formulation offered a fresh lens through which the challenges and solutions of inland waterway shipping could be viewed. This nuanced approach provided methodological robustness and offered practitioners a pragmatic toolset to address real-world scheduling concerns.

In conclusion, the landscape of inland waterway shipping is undergoing a transformative change. As traditional methods grapple with the increasing complexities of modern-day shipping dynamics, emerging techniques such as GNNs offer a beacon of hope. By successfully applying GNNs to various scheduling problems, this research contributes significantly to the field, promising more reliable, efficient, and environmentally sustainable inland waterway transportation systems in the future. The implications of these findings are profound and potentially reshape how waterway transport is viewed and managed in the global logistics ecosystem.

Future work Several prospects for future research in inland waterway shipping scheduling using GNNs have manifested:

- **TSP**: Deepening the exploration into real-world routing predicaments could encompass advanced heuristics or hybrid methodologies. There is also potential in translating JSSP and TSP problems into WSSP contexts. Enhancing the non-autoregressive nature of the GNN model for TSP might lead to superior predictions.

- **JSSP**: An emphasis on amplifying GNN's generalisability across diverse data sets and further experimentation with the GCN architecture could pave the way for more sophisticated models. Prospects include exploring alternative GCN models for JSSP, examining varied data sets, and transposing JSSP scenarios into WSSP contexts.
- **RCPSP**: In spite of its evident potential, developing neural architectures and venturing into other logistical impediments within the IWT realm using GNN-driven techniques may be fruitful.
- **WSSP**: Delving into optimising the translation of WSSP to the RCPSP framework, with a pivot towards potential multi-mode scenarios rather than solely the single-mode RCPSP, is promising. This would allow different modes to be assigned to each waterway, potentially enhancing the flexibility and efficiency of the scheduling process. Additionally, considering stochastic elements in RCPSP could introduce a layer of realism by accounting for the unpredictability inherent in maritime logistics. A broader literature examination and incorporating of environmental factors like tides and draught could provide comprehensive insights. Probing varying operational conditions and scouting other case studies could further fine-tune the model's real-world applicability.

Expanding to larger data sets and weaving in real-time data can further enhance GNN capabilities. The horizon is replete with opportunities to revolutionise the inland waterway shipping industry, optimising it for unparalleled efficiency, adaptability, and resilience.

References

Abdolshah, M. (2014). A review of resource-constrained project scheduling problems (rcpsp) approaches and solutions. *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, *5*(4), 253–286.

Adamu, P. I., & Aromolaran, O. (2018). Machine learning priority rule (mlpr) for solving resource-constrained project scheduling problems.

Agussurja, L., Kumar, A., & Lau, H. C. (2018). Resource-constrained scheduling for maritime traffic management. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1). https://doi.org/10.1609/aaai.v32i1.12086

Ali, I., Essam, D., & Kasmarik, K. (2020). A novel design of differential evolution for solving discrete traveling salesman problems. *Swarm and Evolutionary Computation*, *52*, 100607. https://doi.org/10.1016/j.swevo.2019.100607

Almasan, P., Suárez-Varela, J., Rusek, K., Barlet-Ros, P., & Cabellos-Aparicio, A. (2022). Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Computer Communications*, *196*, 184–194.

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2006). *The traveling salesman problem: A computational study*. Princeton university press.

Arnesen, M. J., Gjestvang, M., Wang, X., Fagerholt, K., Thun, K., & Rakke, J. G. (2017). A traveling salesman problem with pickups and deliveries, time windows and draft limits: Case study from chemical shipping. *Computers Operations Research*, *77*, 20–31. https://doi.org/https://doi.org/10.1016/j.cor.2016.07.017

Artigues, C. (2013). The resource-constrained project scheduling problem - iste. https://www.iste.co.uk/data/doc_dtalmanhopmh.pdf

Aydin, M. E., & Öztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, *33*(2-3), 169–178.

Bai, Q., Xu, J., & Zhang, Y. (2022). The distributionally robust optimization model for a remanufacturing system under cap-and-trade policy: A newsvendor approach. *Annals of Operations Research*, *309*, 731–760. https://doi.org/10.1007/s10479-020-03642-4

Balas, E. (1969). Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. *Operations research*, *17*(6), 941–957.

Bartusch, M., Möhring, R., & Radermacher, F. (1988). Scheduling project networks with resource constraints and time windows. *16*, 199–240.

Bauk. (2004). View of modeling ship's route by the adaptation of hopfield-tank tsp neural algorithm. https://www.jmr.unican.es/index.php/jmr/article/view/21/19

Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, *290*(2), 405–421.

Błażewicz, J., Pesch, E., & Sterna, M. (2000). The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, *127*(2), 317–331.

Bold, M., Boyaci, B., Goerigk, M., & Kirkbride, C. (2021). The generalised resource-constrained project scheduling problem with flexible resource profiles. *Book of Extended Abstracts*, 70.

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, *149*(2), 268–281.

Brcic, M., Katic, M., & Hlupic, N. (2019). Planning horizons based proactive rescheduling for stochastic resource-constrained project scheduling problems. *Eur. J. Oper. Res.*, *273*, 58–66. https://doi.org/10.1016/j.ejor.2018.07.037

Bresson, X., & Laurent, T. (2017). Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, *112*, 3–41. https://doi.org/10.1016/S0377-2217(98)00204-5

Bruni, M. E., & Beraldi, P. F. (2011). A heuristic approach for resource-constrained project scheduling with uncertain activity durations. *Computers & Operations Research*, *38*(9), 1305–1318.

Bulavchuk, A. M., & Semenova, D. V. (2021). Genetic algorithm based on idempotent algebra methods for rcpsp. *Proceedings of the 2021 IEEE 15th International Conference on Application of Information and Communication Technologies (AICT)*, 1–4.

Cai, H., Bian, Y., & Liu, L. (2022). Learning to schedule srcpsp with uncertain resource capacity based on graph neural network and reinforcement learning. *Available at SSRN 4156352*.

Cappart, Q., Chételat, D., Khalil, E., Lodi, A., Morris, C., & Veličković, P. (2022). Combinatorial optimization and reasoning with graph neural networks.

Chakrabortty, R., Abbasi, A., & Ryan, M. J. (2020). Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic. *International Transactions in Operational Research*, *27*(1), 138–167.

Chen, B., Bécigneul, G., Ganea, O.-E., Barzilay, R., & Jaakkola, T. (2020). Optimal transport graph neural networks. *arXiv preprint arXiv:2006.04804*.

Chen, J., & Askin, R. G. (2009). Project selection, scheduling and resource allocation with time-dependent returns. *European Journal of Operational Research*, *193*(1), 23–34.

Chen, W., Xu, Y., & Wu, X. (2017). Deep reinforcement learning for multi-resource multi-machine job scheduling. *arXiv preprint arXiv:1711.07440*.

Chen, X., & Tian, Y. (2018). Learning to progressively plan.

Cook, W. J. (2012). *Mathematics at the limits of computation*. Princeton University Press. https://doi.org/doi:10.1515/9781400839599

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., & Song, L. (2018). Learning combinatorial optimization algorithms over graphs.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, *2*(4), 393–410. Retrieved June 17, 2023, from http://www.jstor.org/stable/166695

Davari, M., & Demeulemeester, E. (2019). The proactive and reactive resource-constrained project scheduling problem. *J. Sched.*, *22*, 211–237. https://doi.org/10.1007/s10951-017-0553-x

Davendra, D., & Bialic-Davendra, M. (2020). Introductory chapter: Traveling salesman problem - an overview. In D. Davendra & M. Bialic-Davendra (Eds.), *Novel trends in the traveling salesman problem*. IntechOpen. https://doi.org/10.5772/intechopen.94435

Defryn, C., Golak, J. A. P., Grigoriev, A., & Timmermans, V. (2021). Inland waterway efficiency through skipper collaboration and joint speed optimization. *European Journal of Operational Research*, *292*(1), 276–285.

de Vries, K. (2015). *Waardevol transport: De toekomst van het goederenvervoer en de binnenvaart in europa*. Bureau Voorlichting Binnenvaart.

Dobrilovic, D. (2022). Uav route planning in urban and suburban surveillance scenarios. In T. A. Kovács, Z. Nyikes, & I. Fürstner (Eds.), *Security-related advanced technologies in critical infrastructure protection* (pp. 217–227). Springer Netherlands.

Dong, X., & Cai, Y. (2019). A novel genetic algorithm for large scale colored balanced traveling salesman problem. *Future Generation Computer Systems*, *95*, 727–742. https://doi.org/10.1016/j.future.2018.12.065

Dorigo, M., & Gambardella, L. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66. https://doi.org/10.1109/4235.585892

Dreyfus, S., & Law, A. (2014). *The art and theory of dynamic programming*. Academic Press, Inc. https://doi.org/10.2307/3009654

Du, Y., Chen, Q., Lam, J. S. L., Xu, Y., & Cao, J. X. (2015). Modeling the impacts of tides and the virtual arrival policy in berth allocation. *Transportation Science*, *49*(4), 939–956.

Du, Y., Chen, Q., Quan, X., Long, L., & Fung, R. Y. (2011). Berth allocation considering fuel consumption and vessel emissions. *Transportation Research Part E: Logistics and Transportation Review*, *47*(6), 1021–1037.

Ernst, A. T., Oğuz, C., Singh, G., & Taherkhani, G. (2017). Mathematical models for the berth allocation problem in dry bulk terminals. *Journal of Scheduling*, *20*, 459–473.

Fagerholt, K., & Christiansen, M. (2000). A travelling salesman problem with allocation, time window and precedence constraints — an application to ship scheduling. *International Transactions in Operational Research*, *7*(3), 231–244. https://doi.org/https://doi.org/10.1016/S0969-6016(00)00021-6

Fan, S., Blanco-Davis, E., Yang, Z., Zhang, J., & Yan, X. (2020). Incorporation of human factors into maritime accident analysis using a data-driven bayesian network. *Reliability Engineering & System Safety*, *203*, 107070.

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric.

Fündeling, C.-U., & Trautmann, N. (2010). A priority-rule method for project scheduling with work-content constraints. *European Journal of Operational Research*, *203*, 568–574. https://doi.org/10.1016/j.ejor.2009.09.019

Ghasemi-Sardabrud, M., Zarkandi, M., & Mahmoodjanloo, M. (2019). A ship routing and scheduling problem considering pickup and delivery, time windows and draft limit. *2019 15th Iran International Industrial Engineering Conference (IIIEC)*, 165–170. https://doi.org/10.1109/IIIEC.2019.8720627

Giusti, R., Manerba, D., Bruno, G., & Tadei, R. (2019). Synchromodal logistics: An overview of critical success factors, enabling technologies, and open research issues. *Transportation Research Part E: Logistics and Transportation Review*, *129*, 92–110. https://doi.org/https://doi.org/10.1016/j.tre.2019.07.009

Golab, A., Sedgh Gooya, E., Alfalou, A., & Cabon, M. (2022). Investigating the performance of an artificial neural network for solving the resource constrained project scheduling problem (rcpsp), 12. https://doi.org/10.1117/12.2618499

Goldratt, E. (1997). *Critical chain*. North River Press.

Günther, E., Lübbecke, M. E., & Möhring, R. H. (2011). Challenges in scheduling when planning the ship traffic on the kiel canal. *Maxim Sviridenko, IBM Watson Research Center Steef van de Velde, Erasmus University Rotterdam Gerhard Woeginger, Technische Universiteit Eindhoven Guochuan Zhang, Zhejiang University*, 23.

Gutierrez-Franco, E., Mejia-Argueta, C., & Rabelo, L. (2021). Data-driven methodology to support long-lasting logistics and decision making for urban last-mile operations. *Sustainability*, *13*(11). https://www.mdpi.com/2071-1050/13/11/6230

Hameed, M. S. A., & Schwung, A. (2020). Reinforcement learning on job shop scheduling problems using graph networks. *CoRR*, *abs/2009.03836*. https://arxiv.org/abs/2009.03836

Hartmann, S., & Briskorn, D. (2008). A survey of deterministic modeling approaches for project scheduling under resource constraints. *European journal of operational research*, *207*, 1–14.

Haupt, R. (1989). A survey of priority rule-based scheduling. *Operations-Research-Spektrum*, *11*(1), 3–16.

Herroelen, W., & Leus, R. (2004). Robust and reactive project scheduling: A review and classification of procedures. *Int. J. Prod. Res.*, *42*. https://doi.org/10.1080/00207540310001638055

Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers Operations Research*, *25*(4), 279–302. https://doi.org/https://doi.org/10.1016/S0305-0548(97)00055-5

Hill, A., Lalla-Ruiz, E., Voß, S., & Goycoolea, M. (2019). A multi-mode resource-constrained project scheduling reformulation for the waterway ship scheduling problem. *Journal of scheduling*, *22*(2), 173–182.

Hofbauer, F., & Putz, L.-M. (2020). External costs in inland waterway transport: An analysis of external cost categories and calculation methods. *Sustainability*, *12*(14), 5874.

Hong-xing, Z., Bao-li, L., Chun-yuan, D., & Pan-pan, F. (2018). Ship scheduling optimization in one-way channel bulk harbor. *Operations Research and Management Science*, *27*(12), 28.

Hougardy, S., Zaiser, F., & Zhong, X. (2020). The approximation ratio of the 2-opt heuristic for the metric traveling salesman problem.

Hu, Y., Zhang, Z., Yao, Y., Huyan, X., Zhou, X., & Lee, W. S. (2021). A bidirectional graph neural network for traveling salesman problems on arbitrary symmetric graphs. *Engineering Applications of Artificial Intelligence*, *97*, 104061. https://doi.org/https://doi.org/10.1016/j.engappai.2020.104061

Hudson, B., Li, Q., Malencia, M., & Prorok, A. (2022). Graph neural network guided local search for the traveling salesperson problem.

Igelmund, G., & Radermacher, F. (2010). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, *13*, 1–28. https://doi.org/10.1002/net.3230130102

Jia, S., Li, C.-L., & Xu, Z. (2019). Managing navigation channel traffic and anchorage area utilization of a container port. *Transportation Science*, *53*(3), 728–745.

Jonkeren, O., Rietveld, P., & van Ommeren, J. (2007). Climate change and inland waterway transport: Welfare effects of low water levels on the river rhine. *Journal of Transport Economics and Policy (JTEP)*, *41*(3), 387–411.

Jonkeren, O., Rietveld, P., van Ommeren, J., & Te Linde, A. (2014). Climate change and economic consequences for inland waterway transport in europe. *Regional Environmental Change*, *14*, 953–965.

Joshi, C. K., Laurent, T., & Bresson, X. (2019). An efficient graph convolutional network technique for the travelling salesman problem.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks.

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In J. Weglarz (Ed.), *Project scheduling: Recent models, algorithms and applications* (pp. 147–178). Springer US. https://doi.org/10.1007/978-1-4615-5533-9_7

Kool, W., van Hoof, H., & Welling, M. (2019). Attention, learn to solve routing problems! *International Conference on Learning Representations*. https://openreview.net/forum?id=ByxBFsRqYm

Koulinas, G. K., Kotsikas, L., & Anagnostopoulos, K. P. (2014). A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Inf. Sci.*, *277*, 680–693.

Krose, B. (1996). *An introduction to neural networks*.

Kumar, D.-K. (2012). *Traveling salesman problem (tsp): A comparative analysis*.

Kyriakidis, T. S., Kopanos, G. M., & Georgiadis, M. C. (2012). Milp formulations for single- and multi-mode resource-constrained project scheduling problems. *Computers & Chemical Engineering*, *36*, 369–385.

Lalla-Ruiz, E., Shi, X., & Voß, S. (2018). The waterway ship scheduling problem [Special Issue on Traffic Modeling for Low-Emission Transport]. *Transportation Research Part D: Transport and Environment*, *60*, 191–209. https://doi.org/https://doi.org/10.1016/j.trd.2016.09.013

Lambrechts, O., Demeulemeester, E., & Herroelen, W. (2008). Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *J. Sched.*, *11*, 121–136. https://doi.org/10.1007/s10951-007-0021-0

Larranaga, P., Kuijpers, C., Murga, R., Inza, I., & Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, *13*, 129–170. https://doi.org/10.1023/A:1006529012972

Li, H., & Womer, N. (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *Eur. J. Oper. Res.*, *246*, 20–33. https://doi.org/10.1016/j.ejor.2015.04.015

Li, S., & Jia, S. (2019). The seaport traffic scheduling problem: Formulations and a column-row generation algorithm. *Transportation Research Part B: Methodological*, *128*, 158–184.

Liu, C.-L., Chang, C.-C., & Tseng, C.-J. (2020). Actor-critic deep reinforcement learning for solving job shop scheduling problems. *Ieee Access*, *8*, 71752–71762.

Lübbecke, E. (2018). On-and offline scheduling of bidirectional traffic. *Operations Research Proceedings 2016: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Helmut Schmidt University Hamburg, Germany, August 30-September 2, 2016*, 9–15.

Lübbecke, E., Lübbecke, M. E., & Möhring, R. H. (2019). Ship traffic optimization for the kiel canal. *Operations Research*, *67*(3), 791–812.

Mahmoodjanloo, M., Chen, G., Asian, S., Iranmanesh, S. H., & Tavakkoli-Moghaddam, R. (2021). In-port multi-ship routing and scheduling problem with draft limits. *Maritime Policy & Management*, *48*(7), 966–987. https://doi.org/10.1080/03088839.2020.1783465

Malaguti, E., Martello, S., & Santini, A. (2018). The traveling salesman problem with pickups, deliveries, and draft limits. *Omega*, *74*, 50–58. https://doi.org/https://doi.org/10.1016/j.omega.2017.01.005

Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. *Proceedings of the 15th ACM workshop on hot topics in networks*, 50–56.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*, 115–133.

Medress, M. F., Cooper, F. S., Forgie, J. W., Green, C., Klatt, D. H., O'Malley, M. H., Neuburg, E. P., Newell, A., Reddy, D., Ritea, B., et al. (1977). Speech understanding systems: Report of a steering committee. *Artificial Intelligence*, *9*(3), 307–316.

Meisel, F., & Fagerholt, K. (2019). Scheduling two-way ship traffic for the kiel canal: Model, extensions and a matheuristic. *Computers & Operations Research*, *106*, 119–132.

Miranda, P. A., Blazquez, C. A., Obreque, C., Maturana-Ross, J., & Gutierrez-Jarpa, G. (2018). The bi-objective insular traveling salesman problem with maritime and ground transportation costs. *European Journal of Operational Research*, *271*(3), 1014–1036. https://doi.org/https://doi.org/10.1016/j.ejor.2018.05.009

Naber, A. (2017). Resource-constrained project scheduling with flexible resource profiles in continuous time. *Computers Operations Research*, *84*, 33–45. https://doi.org/https://doi.org/10.1016/j.cor.2017.02.018

Naber, A., & Kolisch, R. (2014). Mip models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, *239*(2), 335–348. https://doi.org/https://doi.org/10.1016/j.ejor.2014.05.036

Naderi, S., Vaez-ghasemi, M., & Sobhani, F. (2022). Optimizing resource-constrained project scheduling problem considering the reliability function. *Discrete Dynamics in Nature and Society*, *2022*. https://doi.org/10.1155/2022/7711383

Nelson, K., Camp, J., & Philip, C. (2015). Navigable inland waterway transportation modeling: A conceptual framework and modeling approach for consideration of climate change induced extreme weather events. *SMART RIVERS*.

Nightingale, P., & Demirović, E. (1999). CSPLib problem 061: Resource-constrained project scheduling problem (rcpsp) (C. Jefferson, I. Miguel, B. Hnich, T. Walsh, & I. P. Gent, Eds.).

Notteboom, T., Yang, D., & Xu, H. (2020). Container barge network development in inland rivers: A comparison between the yangtze river and the rhine river. *Transportation Research Part A: Policy and Practice*, *132*, 587–605.

Notteboom, T. E. (2006). The time factor in liner shipping services. *Maritime Economics & Logistics*, *8*, 19–39.

Nur, F., Marufuzzaman, M., & Puryear, S. M. (2020). Optimizing inland waterway port management decisions considering water level fluctuations. *Computers & Industrial Engineering*, *140*, 106210.

of Infrastructure, R. M., & management, W. (2022). Vaarwegenoverzicht - informatie en data. https://www.rijkswaterstaat.nl/water/vaarwegenoverzicht/

Ozkan, O., & Gülçiçek. (2015). A neural network for resource constrained project scheduling programming. *Journal of Civil Engineering and Management*, *21*. https://doi.org/10.3846/13923730.2013.802723

Pan, M.-J., & Jang, W.-Y. (2008). Determinants of the adoption of enterprise resource planning within the technology-organization-environment framework: Taiwan's communications industry. *3e Journal of Computer Information Systems*, *48*(3), 94–102.

Pandiri, V., & Singh, A. (2019). An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem. *Applied Soft Computing*, *78*, 481–495. https://doi.org/10.1016/j.asoc.2019.03.001

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Courier Corporation.

Praetorius, G., Lützhöft, M., & Bruno, K. (2010). The context matters: Maritime safety in the vessel traffic service (vts) domain. *Reliability, Risk and Safety: Back to the Future*.

Prates, M. O. R., Avelar, P. H. C., Lemos, H., Lamb, L., & Vardi, M. (2018). Learning to solve np-complete problems - a graph neural network for decision tsp.

Ranjbar, M., Khalilzadeh, M., Kianfar, F., & Etminani, K. (2012). An optimal procedure for minimizing total weighted resource tardiness penalty costs in the resource-constrained project scheduling problem. *Computers Industrial Engineering*, *62*(1), 264–270. https://doi.org/https://doi.org/10.1016/j.cie.2011.09.013

Reinelt, G. (1991). https://pubsonline.informs.org/doi/abs/10.1287/ijoc.3.4.376

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton project para*. Cornell Aeronautical Laboratory.

Saad, H. M. H., Chakrabortty, R. K., Elsayed, S., & Ryan, M. J. (2021). Quantum-inspired genetic algorithm for resource-constrained project-scheduling. *IEEE Access*, *9*, 38488.

Salman, R., Ekstedt, F., & Damaschke, P. (2020). Branch-and-bound for the precedence constrained generalized traveling salesman problem. *Operations Research Letters*, *48*(2), 163–166. https://doi.org/10.1016/j.orl.2020.01.009

Samsonov, V., Kemmerling, M., Paegert, M., Lütticke, D., Sauermann, F., Gützlaff, A., Schuh, G., & Meisen, T. (2021). Manufacturing control in job shop environments with reinforcement learning. *ICAART (2)*, 589–597.

Schutt, A., Feydy, T., & Stuckey, P. J. (2013). Scheduling optional tasks with explanation. In C. Schulte (Ed.), *Principles and practice of constraint programming* (pp. 628–644). Springer Berlin Heidelberg.

Shi, Y., & Zhang, Y. (2022). The neural network methods for solving traveling salesman problem [The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 2021): Developing Global Digital Economy after COVID-19]. *Procedia Computer Science*, *199*, 681–686. https://doi.org/https://doi.org/10.1016/j.procs.2022.01.084

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., & Sun, Y. (2020). Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

Shou, Y. (2005). A neural network based heuristic for resource-constrained project scheduling. *Lecture Notes in Computer Science*, *3496*, 794–799. https://doi.org/10.1007/11427391_127

Sirimanne, S. N., Hoffman, J., Juan, W., Asariotis, R., Assaf, M., Ayala, G., Benamara, H., Chantrel, D., Hoffmann, J., Premti, A., et al. (2019). Review of maritime transport 2019. *United Nations conference on trade and development, Geneva, Switzerland*.

Smith, K. (1996). An argument for abandoning the travelling salesman problem as a neural-network benchmark. *IEEE Transactions on Neural Networks*, *7*(6), 1542–1544. https://doi.org/10.1109/72.548187

Sola, J., & Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on nuclear science*, *44*(3), 1464–1468.

Sprecher, A., Hartmann, S., & Drexl, A. (1997). An exact algorithm for project scheduling with multiple modes. *Operations-Research-Spektrum*, *19*, 195–203.

Tan, Z., Wang, Y., Meng, Q., & Liu, Z. (2018). Joint ship schedule design and sailing speed optimization for a single inland shipping service with uncertain dam transit time. *Transportation Science*, *52*(6), 1570–1588.

Tao, S., & Dong, Z. S. (2018). Multi-mode resource-constrained project scheduling problem with alternative project structures. *Computers & Industrial Engineering*, *125*, 333–347.

Teichteil-Königsbuch, F., Povéda, G., de Garibay Barba, G. G., Luchterhand, T., & Thiébaux, S. (2023). Fast and robust resource-constrained scheduling with graph neural networks.

Thieme, C. A., Utne, I. B., & Haugen, S. (2018). Assessing ship risk model applicability to marine autonomous surface ships. *Ocean Engineering*, *165*, 140–154.

Tritschler, M., Naber, A., & Kolisch, R. (2017). A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, *262*(1), 262–273. https://doi.org/https://doi.org/10.1016/j.ejor.2017.03.006

Ulusçu, Ö. S., Özbaş, B., Altıok, T., Or, I., & Yılmaz, T. (2009). Transit vessel scheduling in the strait of istanbul. *The Journal of Navigation*, *62*(1), 59–77.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Verstichel, J., De Causmaecker, P., Spieksma, F., & Vanden Berghe, G. (2014a). Exact and heuristic methods for placing ships in locks [Maritime Logistics]. *European Journal of Operational Research*, *235*(2), 387–398. https://doi.org/https://doi.org/10.1016/j.ejor.2013.06.045

Verstichel, J., De Causmaecker, P., Spieksma, F., & Berghe, G. V. (2014b). The generalized lock scheduling problem: An exact approach. *Transportation Research Part E: Logistics and Transportation Review*, *65*, 16–34.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in Neural Information Processing Systems*, 2692–2700.

Wang, C., Zhang, X., Chen, X., Li, R., & Li, G. (2017). Vessel traffic flow forecasting based on bp neural network and residual analysis. *2017 4th International Conference on Information, Cybernetics and Computational Social Systems (ICCSS)*, 350–354.

Wang, S., Liu, M., & Chu, F. (2020). Approximate and exact algorithms for an energy minimization traveling salesman problem. *Journal of Cleaner Production*, *249*, 119433. https://doi.org/https://doi.org/10.1016/j.jclepro.2019.119433

Wang, S., Sun, S., & Zhou, B. (2007). Q-learning based dynamic single machine scheduling. *JOURNAL-SHANGHAI JIAOTONG UNIVERSITY-CHINESE EDITION-*, *41*(8), 1227.

Waterstaat, M. v. I. e. (2021). Inland shipping. https://www.government.nl/topics/freight-transportation/inland-shipping

Wen, Y., Huang, Y., Zhou, C., Yang, J., Xiao, C., & Wu, X. (2015). Modelling of marine traffic flow complexity. *Ocean Engineering*, *104*, 500–510.

Wieder, O., Kohlbacher, S., Kuenemann, M., Garon, A., Ducrot, P., Seidel, T., & Langer, T. (2020). A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, *37*, 1–12.

Xie, F., Li, H., & Xu, Z. (2021). Multi-mode resource-constrained project scheduling with uncertain activity cost. *Expert Systems with Applications*, *168*, 114475.

Xiong, H., Shi, S., Ren, D., & Hu, J. (2022). A survey of job shop scheduling problem: The types and models. *Computers & Operations Research*, *142*, 105731.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? *CoRR*, *abs/1810.00826*. http://arxiv.org/abs/1810.00826

Yang, Z., Shi, H., Chen, K., & Bao, H. (2014). Optimization of container liner network on the yangtze river. *Maritime Policy & Management*, *41*(1), 79–96.

Yuan, Y., Cattaruzza, D., Ogier, M., & Semet, F. (2020). A branch-and-cut algorithm for the generalized traveling salesman problem with time windows. *European Journal of Operational Research*, *286*(3), 849–866. https://doi.org/https://doi.org/10.1016/j.ejor.2020.04.024

Zalzala, A. M., & Fleming, P. J. (1997). *Genetic algorithms in engineering systems* (Vol. 55). Iet.

Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P. S., & Xu, C. (2020). Learning to dispatch for job shop scheduling via deep reinforcement learning.

Zhang, W., & Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. *IJCAI*, *95*, 1114–1120.

Zhang, X., Chen, X., Ji, M., & Yao, S. (2017). Vessel scheduling model of a one-way port channel. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, *143*(5), 04017009.

Zhang, X., Lin, J., Guo, Z., & Liu, T. (2016). Vessel transportation scheduling optimization based on channel–berth coordination. *Ocean Engineering*, *112*, 145–152.

Zhao, X., Song, W., Li, Q., Shi, H., Kang, Z., & Zhang, C. (2022). A deep reinforcement learning approach for resource-constrained project scheduling. *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1226–1234. https://doi.org/10.1109/SSCI51031.2022.10022122

Zhu, G., Bard, J. F., & Yu, G. (2006). A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS J. on Computing*, *18*(3), 377–390. https://doi.org/10.1287/ijoc.1040.0121

A Appendices

Appendix A: Scientific Research Paper (Begins on the Following Page)

GRAPH NEURAL NETWORKS FOR INLAND WATERWAY SHIP SCHEDULING

SCIENTIFIC PAPER

Nahom Tsehaie
Transport Engineering and Logistics
Technical University of Delft
Mekelweg 5, 2628 CD Delft, The Netherlands
nahomtsehaie@gmail.com

Peter Wenzel
Transport Engineering and Logistics
Technical University of Delft
Mekelweg 5, 2628 CD Delft, The Netherlands
p.a.wenzel@tudelft.nl

Frederik Schulte
Transport Engineering and Logistics
Technical University of Delft
Mekelweg 5, 2628 CD Delft, The Netherlands
f.schulte@tudelft.nl

Raka Jovanovic
Qatar Environment and Energy Research Institute
Hamad bin Khalifa University
PO Box 5825, Doha, Qatar
rjovanovic@hbku.edu.qa

December, 2023

ABSTRACT

Waterway Ship Scheduling Problem (WSSP) involves optimising the scheduling and movement of ships through waterway networks, considering various constraints such as waterway capacity, vessel specifications, and timing requirements for ship movements. The WSSP is related to the Resource-Constrained Project Scheduling Problem (RCPSP), as its scheduling and resource management complexities can be addressed using RCPSP methodologies. This study applies Graph Neural Networks (GNN) to the WSSP within the RCPSP framework for inland waterway transport. This approach leverages RCPSP's framework for handling constraints and resources in project scheduling, making it suitable for optimising ship movements and berth allocations in waterway systems. The framework employs GNN to predict solutions for RCPSP, improving the schedule quality and efficiency through a post-processing step using the Schedule Generation Scheme (SGS). The study adapts WSSP variables for single-mode RCPSP and examines various scenarios with different numbers of waterways and ships. The study includes randomly generated data and a case study on the Port of Duisburg.

The findings show that the GNN model closely aligns with optimal solutions, especially in complex scenarios, and demonstrates precision in scheduling with minimal optimality gaps. The model's rapid computation is a significant advantage, though further optimisation is needed for handling larger tasks and more intricate waterway configurations.

Keywords Graph Neural Network · Resource Constraint Project Scheduling Problem · Waterway Ship Scheduling Problem · Inland Waterway Transport · Machine Learning

1 Introduction

Waterway ship scheduling is pivotal for effectively moving cargo on inland waterways, a primary method of large cargo transportation (Lalla-Ruiz et al., 2018). It bolsters the economy of regions like the Rhine Basin and offers an alternative to road and rail systems (Notteboom et al., 2020; Tan et al., 2018). The EU-27 boasts about 30,000 km of rivers and canals aiding the transportation of diverse goods (Hofbauer and Putz, 2020). But challenges like unpredictable weather, fluctuating water levels, and limited shipping data complicate scheduling (Fan et al., 2020). With the increasing use of

ports, understanding and managing maritime traffic characteristics is foundational for enhancing navigational safety and reducing waiting times at ports (Sirimanne et al., 2019).

Inland waterway transportation faces unpredictability due to fluctuating water levels and weather changes. These uncertainties complicate scheduling, potentially leading to delays and inefficiencies. The specific challenges of this mode, such as navigating through narrow channels, make it unique from other transportation methods (Notteboom, 2006). As such, there's a growing need for strategies that enhance its efficiency and reliability (Sirimanne et al., 2019).

In our research, we are delving into the practical aspects of the WSSP, drawing insights from studies by Lalla-Ruiz et al. (2018); Hill et al. (2019). The WSSP addresses critical issues in port connectivity, focusing on efficiently managing ship traffic in capacity-restricted waterways. The challenge lies in scheduling incoming and outgoing ships to minimise waiting times, thereby reducing potential bottlenecks and environmental impacts due to ship emissions. This problem is particularly pertinent in busy maritime corridors, where efficient scheduling can significantly enhance port competitiveness. The WSSP models these complexities effectively, providing a framework for optimising traffic flow and resource utilisation in congested waterway networks.

Transitioning to the modelling framework, the WSSP can be translated into the Resource-Constrained Project Scheduling Problem format. This translation, as explored by Hill et al. (2019), allows for a more structured approach to managing the intricate logistics of waterway traffic. By framing the WSSP within the RCPSP paradigm, we can apply well-established project scheduling techniques to this maritime context. This approach provides a global perspective on resource allocation, time management, and the optimisation of sequential tasks, which are key aspects of WSSP and RCPSP. The specifics of this translation, including the nuances and methodologies, will be elaborated upon in later sections of our work.

In addressing the translation of WSSP to RCPSP, we aim to leverage the capabilities of Graph Neural Networks (GNN) inspired by the research by Teichteil-Königsbuch et al. (2023). GNNs offer a novel and promising approach to handling the complex scheduling and resource allocation challenges inherent in the WSSP when reformulated as an RCPSP. GNNs are particularly suited for this task due to their ability to process and interpret graph-structured data, which is a fundamental aspect of project scheduling problems. The application of GNNs in WSSP and RCPSP shows promising advancements. For WSSP, neural networks, as demonstrated by Wang et al. (2017), are used for predicting vessel traffic flow. In RCPSP, deep learning and GNNs are increasingly recognised for their ability to handle complex scheduling constraints, with studies like those of Shou (2005) and Cai et al. (2022) exploring their applications in optimising project durations and enhancing computational efficiency.

This paper introduces a novel GNN approach to improve prediction and reliability in inland waterway shipping scheduling. Applying GNN methodologies, known for their capability in modelling complex networks, is expected to significantly boost the efficiency and accuracy of scheduling solutions for the dynamic and demanding context of inland waterway systems. The subsequent sections will delve into the specifics of implementing GNNs in this context, illustrating their potential to revolutionise the approach to scheduling in waterway networks. A pivotal part of the research centres on the question: How can the Waterway Ship Scheduling Problem (WSSP) complexities be translated into the Resource-Constrained Project Scheduling Problem (RCPSP) formulation? This involves converting key WSSP variables into the RCPSP context and assessing the performance of the proposed RCPSP-based approach in real-world settings. This GNN application in the inland waterway context, specifically for the RCPSP problem, is a groundbreaking contribution since GNN leverages the strengths in understanding and representing complex data structures, aiming for efficient, accurate, and near-optimal solutions for both RCPSP and WSSP.

After the introduction, we present the related work section, highlighting existing literature and foundational studies upon which this research is built. A deep dive into the formulation of the WSSP and RCPSP follows this. The methodology section discusses the GNN model, its architecture, the data used, and the training process. Our findings are showcased in the results section, and we conclude by encapsulating significant insights and pointing towards future research avenues in waterway shipping scheduling using GNNs.

2 Related Work

In the realm of waterway transportation, the WSSP poses unique challenges due to its distinct operational modes and resource constraints. Foundational work by Lalla-Ruiz et al. (2018) and subsequent adaptations by Hill et al. (2019) have translated WSSP into the RCPSP framework, effectively integrating the multifaceted nature of ship operations in waterways and resolving it with integer programming. Despite these advances, machine learning applications in WSSP remain limited, highlighting the need for more systematic and theoretically informed approaches to overcome the inefficiencies of current manual scheduling methods (Wang et al., 2017; Praetorius et al., 2010).

Furthermore, while various methodologies like Lagrangian relaxation and column generation algorithms have been proposed to tackle ship scheduling complexities, their practicality is often hindered by extended computational times, leading to a preference for heuristic solutions (Jia et al., 2019; Li and Jia, 2019; Zhang et al., 2016; Lalla-Ruiz et al., 2018; Hong-xing et al., 2018; Meisel and Fagerholt, 2019; Zhang et al., 2017).

The translation of the WSSP to the RCPSP enhances problem-solving efficiency and aligns with the trend towards more robust optimisation formulations. As explored in Hill et al. (2019), this transition leverages RCPSP's refined project scheduling methods for better resource allocation and time management, crucial in maritime logistics. It enables handling larger, complex scenarios and offers improved decision-making with tighter solution quality bounds. Furthermore, this alignment facilitates the effective use of deep learning, allowing for advanced, data-driven solutions in maritime transport system management and coordination. The study of Hill et al. (2019) utilised a Mixed Integer Programming (MIP) formulation to address the RCPSP within the context of waterway ship scheduling. This MIP model was found to be more efficient than previous models, with significantly fewer binary variables and constraints.

The Resource-Constrained Project Scheduling Problem has become a focal point in project scheduling research, particularly with the advent of deep learning techniques and their applications in combinatorial optimisation (Bengio et al., 2021; Shou, 2005). However, when combined with reinforcement learning, the scalability and generalisation of these methods remain challenging (Zhao et al., 2022; Adamu and Aromolaran, 2018). GNNs, adept at handling graph-structured data, have led to novel methods in scheduling, including the integration of genetic algorithms and neural networks, and the use of GNNs for optimising RCPSP, enhancing both solution quality and computational efficiency (Shou, 2005; Teichteil-Königsbuch et al., 2023).

Notably, Teichteil-Königsbuch et al. (2023) pioneered the integration of Graph Neural Networks into the RCPSP framework. In their approach, activities are characterised as nodes, precedence constraints are delineated as edges, and resource availability is encoded within node features. This innovative representation reformulates scheduling as a combinatorial optimisation task. Their GNN model, trained via the Proximal Policy Optimisation (PPO) algorithm, showcased superior performance in solution quality and computational efficiency over traditional methods.

Building on the innovative approaches of Hill et al. (2019) and Teichteil-Königsbuch et al. (2023), our study explores the use of GNNs in addressing the RCPSP within the context of Inland Waterway Transportation (IWT). This novel approach not only advances the existing methods but also contributes significantly to the literature, marking the first to apply GNNs to solve RCPSP in the IWT setting.

3 Problem Formulation

This section begins with the problem formulation of WSSP, presenting an in-depth examination of its mathematical framework and operational specifics. It then moves on to the RCPSP, including its mathematical formulation. The section concludes with a discussion on the formulation for translating WSSP into the RCPSP context, encompassing the detailed mathematical formulations and definitions required for this intricate translation process.

3.1 WSSP Formulation

The Waterway Ship Scheduling Problem is designed to efficiently coordinate the movements of ships within available waterways over a specified duration Hill et al. (2019). As shown in Hill et al. (2019) and depicted in Figure 1, the process begins when a ship arrives and might face a short wait due to congestion. Ships navigate through the port's inbound and outbound passages of two distinct waterways, where they proceed to port operations. After handling cargo and other activities, vessels may experience a port wait before departing. The flexibility to choose between Waterway 1 and Waterway 2 allows ships to manoeuvre based on real-time traffic conditions and operational feasibility. Environmental factors like tidal patterns and the physical limitations of each waterway further guide such decisions. The diagrammatic representation in the Figure underscores the intricate flow of maritime traffic and the necessity for strategic scheduling to enhance efficiency in port operations and waterway management.

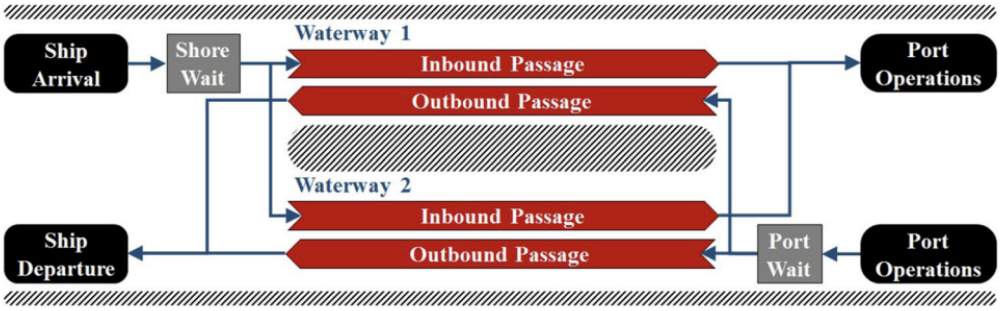

Figure 1: A schematic representation of the maritime traffic flow through the port and waterway system, depicting the potential for shore waits due to traffic before ships enter for inbound processes or depart through outbound channels. This illustrates the WSSP's challenge to minimise ship turnaround times by strategically scheduling arrivals and departures in consideration of traffic, directional use, and tidal conditions (Hill et al., 2019).

The following WSSP formulation is adapted from the work of Hill et al. (2019), where ships and waterways play a pivotal role in this problem:

Ships ($V = \{1..., v\}$) can be categorised as either incoming or outgoing, represented as ($V_1 = \{1, 2..., v_1\}$) and ($V_2 = \{1, 2..., v_2\}$) respectively. The waterways ($W = \{1..., w\}$) are the channels that facilitate these ship movements. Time steps ($H = \{1..., h\}$) denote the specific intervals within two days, serving as the representative time horizon ($T = t_0, t_0 + d, ..., t_0 + h \cdot d$, where each time step $t \in T$ represents the period $[t, t + d]$ and d its time extension).

Waterways (W) come with their own set of attributes. They have inherent time variations and a predetermined width $B(w)$. These channels provide two access modes: an 'in' for entry and an 'out' for exit. Additionally, each waterway has defined capacities, termed as $Q_{in}(w)$ and $Q_{out}(w)$, which denote the maximum number of ships that can enter or exit simultaneously.

Every ship (V), irrespective of its direction, comes with a set of specifications. These include dimensions such as width ($b(v)$), length ($l(v)$), and depth ($d(v)$). For inbound ships ($v_{in} \in V_{in}$), there's an Estimated Time of Arrival (ETA) ($a_{in}(v_{in})$) and a latest permissible entry time ($a'_{in}(v_{in})$). Outbound vessels $v_{out} \in V_{out}$, on the other hand, have both earliest ($a'_{out}(v_{out})$) and last departure times ($a_{out}(v_{out})$). Moreover, ships have travel times that vary based on their direction and the waterway they use ($z_{in}(v_{in}, w)$ and $z_{out}(v_{out}, w)$). Each vessel is also allocated a specific time slot (T_w), marking when they are available for movement.

The crux of the WSSP lies in efficient scheduling. The primary goal is to assign each inbound ship an entry waterway and time and an exit waterway and time for outbound ships. This scheduling ensures:

- Adherence to the maximum allowed ships ($Q_{in}(w)$ and $Q_{out}(w)$ on a waterway at any given time.
- Usage of waterways strictly during operational hours.
- Scheduling of inbound ships must not occur before their ETA, and outbound ships should not be scheduled before their designated departure time.
- Compatibility of waterway depth with ship's draught.
- Accommodation of ships on a waterway without exceeding its width capacity.

The main objective of the WSSP is to optimise ship turnaround times, reducing idle times across all ship types. This formulation encapsulates the intricate balance between the operational constraints and the dynamic nature of waterway traffic. The problem's mathematical formulation, derived from Hill et al. (2019), is presented below:

$$\text{minimise} \quad \sum_{j \in J} \sum_{m \in M_j} \sum_{t \in T} t - \text{EST}_j x_{j,m,t} \tag{1}$$

subject to

$$\sum_{m \in M_j} \sum_{t \in T} x_{j,m,t} = 1 \qquad \forall j \in J, \tag{2}$$

$$\sum_{j \in J} \sum_{m \in M_j} \sum_{t' < t \leq t + d_m} u_{m,r} x_{j,m,t'} \leq q_{r,t} \qquad \forall r \in R, t \in T, \tag{3}$$

$$\text{EST}_j \leq \sum_{m \in M_j} \sum_{t \in T} (t - d_m) x_{j,m,t} \leq \text{LST}_j \qquad \forall j \in J, \tag{4}$$

$$x_{j,m,t} \in \{0, 1\} \qquad \forall j \in J, m \in M_j, t \in T. \tag{5}$$

As outlined in Equation 1, the goal is to minimise the difference between task completion times and release times. The provision in Equation 2 ensures each ship (or job) is allocated a schedule, and precisely one waterway (or mode) is selected. The capacity stipulations in Equation 3 prevent resource over-utilisation in any given time frame. For each resource and timeframe, a knapsack limitation is set. Although all modes are up for consideration, they will only utilise resources if selected. As for the WSSP, Equation 3 maintains consistent adherence to waterway constraints, such as traffic and spatial limitations, throughout the planning span. The changing resource availability across time, especially pertaining to varying waterway depths, is captured by the right-hand side of Equation 3. The constraints in Equation 4 dictate the permissible start times for tasks, corresponding to the timeframes in which ships are allowed to access a waterway. The model introduces a unique set of decision variables, as seen in constraint Equation 5. Specifically, a binary variable $x_{j,m,t}$ is used, which becomes 1 when task j concludes at time t in mode m.

3.2 RCPSP Formulation

The Resource-Constrained Project Scheduling Problem (RCPSP) is pivotal in operations research and project management, focusing on efficiently allocating limited resources to various project activities within multiple constraints (Brucker et al., 1999). Widely applicable across sectors like engineering, software development, and manufacturing, RCPSP extends to other scheduling challenges, such as job-shop, open-shop, and flow-shop scheduling, demonstrating its broad applicability in streamlining project timelines. RCPSP is a combinatorial optimisation challenge that aims to optimally schedule a set of activities within a project, considering resource limits and task precedence. It involves activities, including start and end dummy tasks (source and sink), each with a specific duration, and an Activity-on-Node graph representing task orderings. The problem utilises renewable resources with defined maximum availabilities and particular demands for each activity. A valid schedule sets start times for all activities while adhering to precedence and resource usage constraints. The objective is to minimise the makespan, the start time of the final task, with all activity durations being integers to facilitate optimal scheduling.

Key aspects of RCPSP include:

- Each activity has a predetermined duration and is non-interruptible.
- A precedence system exists, represented by a *Directed Acyclic Graph* (DAG). This ensures one activity is completed before the next begins.
- The source and sink activities, both with zero duration, represent the beginning and the completion of the project timeline, respectively.
- Renewable resources with defined capacities are incorporated, and their allocation must respect these limits.
- The primary goal is optimising the makespan, marking the completion of the sink activity.

The problem is formulated as follows:

$$\text{Objective Function:} \quad \text{minimise} \quad f_n \tag{6}$$

$$\text{Precedence Constraint:} \quad f_i \leq f_j - d_j, \qquad \forall (A_i, A_j) \in \text{Pred} \tag{7}$$

$$\text{Resource Availability Constraint:} \quad \sum_{i \in P_t} u_{irt} \leq U_r, \qquad \forall t = 1, \ldots, f_n, \quad \forall r \in R \tag{8}$$

$$\text{Dummy Activities Constraint:} \quad d_1 = 0, \quad d_n = 0 \tag{9}$$

In the given context, f_n represents the overall project duration, which is the primary objective to be minimised. The terms f_i and f_j refer to the completion times of activities i and j, respectively. The duration of each activity i is denoted

by d_i. The notation Pred represents pairs of activities that directly succeed one another. The variable u_{irt} indicates the amount of a renewable resource r utilised by activity i at a specific time t, while U_r signifies the total available quantity of the renewable resource r.

The activity-on-node graph in Figure 2 illustrates an example of the precedence constraints connecting the activities $A_i \in V$. A schedule with the shortest makespan, $S^*_{n+1} = 12$, is depicted as a two-dimensional Gantt chart in Figure 3. In this chart, the x-axis signifies time, while the y-axis indicates resource utilisation.

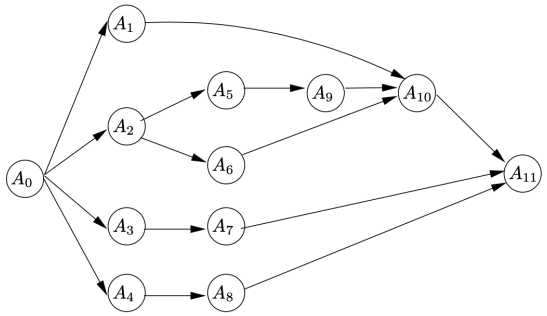

Figure 2: Activity-on-node graph showing the precedence relationships between activities in an RCPSP instance, guiding the project's scheduling constraints.

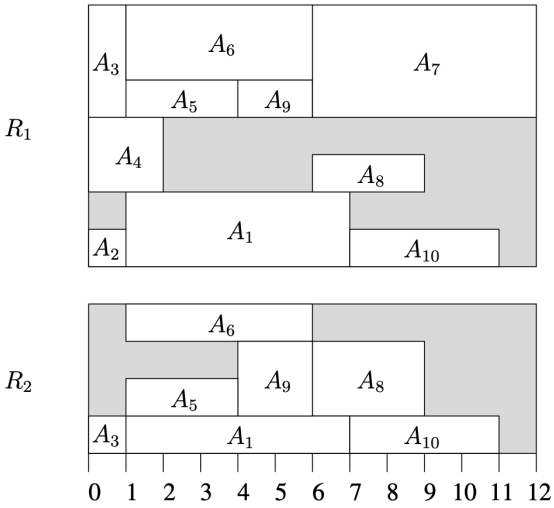

Figure 3: Optimal schedule Gantt chart for an RCPSP instance, demonstrating the shortest makespan and resource allocation over time.

3.3 WSSP Translation to RCPSP

Translating the WSSP into a single-mode RCPSP is a meticulous process that involves restructuring the distinct characteristics of waterways and ships into a framework suitable for project scheduling. This translation is twofold, addressing the waterways' attributes and the ships' specifics.

For waterway properties, each waterway (W) is characterised by a constant width ($B(w)$) and a depth that varies over time ($D(w,t)$). In the RCPSP model, these waterways are represented as individual resources. Each waterway is bifurcated into two distinct resources, one for incoming traffic and another for outgoing, to reflect their two-directional nature. This delineation allows for more precise traffic flow and resource allocation representation. Additionally, constraints that limit the number of ships entering ($Q_{\text{in}}(w)$) and exiting ($Q_{\text{out}}(w)$) the harbour at any given time are translated into specific resource capacities. This translation ensures that the model can accurately manage the flow of ships through each waterway, adhering to the real-world limitations of harbour traffic.

When it comes to ship properties, each vessel, defined by its width ($b(v)$), length ($l(v)$), and depth ($d(v)$), is modelled as an individual task within the RCPSP framework. The physical dimensions of the ships are thus represented by assigning the requisite resources to each task. The travel durations, contingent upon the direction of the ship's movement, are directly integrated with the durations of these tasks. This integration is crucial for replicating the operational timelines of ships within the scheduling model. However, an important aspect of the WSSP, the time windows (T_w) during which ships are permitted to traverse a waterway, is not incorporated into the RCPSP model. This exclusion is a noteworthy deviation from the original WSSP framework.

Further modifications are required to align the single-mode RCPSP with the constraints of WSSP. Arrival times or Estimated Time of Arrival (ETA), denoted as $a_{\text{in}}(v_{\text{in}})$, are integrated as separate tasks for each ship, incorporating dummy durations that represent time windows with zero resource availability. This inclusion is pivotal for ensuring that ships adhere to their scheduled arrival times, thereby maintaining the integrity of the scheduling process. Similarly, constraints regarding the latest permissible time for a ship to enter an inbound waterway ($a'_{\text{in}}(v_{\text{in}})$), as well as the earliest ($a_{\text{out}}(v_{\text{out}})$) and latest ($a'_{\text{out}}(v_{\text{out}})$) departure times from the port, are integrated as separate tasks. These tasks employ

dummy durations during periods where resource availability is nil, ensuring that the ships do not deviate from their designated time windows.

The integration of these time-related constraints is accomplished through the establishment of precedence relations between tasks and their corresponding dummy counterparts. A "start-to-start" relation between a task and its dummy counterpart dictates the earliest start time. In contrast, a "finish-to-finish" relation ensures adherence to the latest permissible start time. This method is applied consistently across all tasks to establish limits on arrival and departure times.

Constraints and Precedence Relations The resource constraints for maritime navigation scenarios, encompassing either two or four waterways, provide crucial guidelines for efficient and safe operations. In scenarios with two waterways, the constraints are tailored explicitly to Waterways A and B, focusing on a more condensed set of parameters. This includes the total number of incoming and outgoing ships, waterway width limitations, and maximum allowable depths for entry and exit. The approach is designed to optimise maritime traffic and uphold safety standards, ensuring a smooth flow of vessels through these waterways.

The constraints are expanded to address Waterways A through D individually for environments involving four waterways. Each waterway has its specific set of rules regarding traffic capacity, navigational width, and maximum permissible depth for ships. This detailed delineation is essential for managing the complexities associated with more waterways, ensuring precise control over maritime traffic and resource allocation.

Expanding the framework to include six waterways introduces new constraints for Waterways E and F, similar to those for C and D, bringing the total number of constraints to thirty. This expansion enhances the model's ability to handle a broader range of maritime traffic scenarios, demonstrating scalability and adaptability. The constraints, referred to as "Qlimit" henceforth, provide a comprehensive overview of resource distribution and management for various maritime navigation setups.

Regarding task precedence for each ship, the structure ensures a logical and coherent progression of tasks. For instance, Task 1 represents the earliest arrival time and must precede Tasks 2 (latest arrival time) and 3 (duration of incoming ships). Similarly, Task 4 (earliest departure time) follows Task 3 and precedes Task 5 (latest departure time) and 6 (duration of outgoing ships), with Task 6 scheduled before Task 5. This structured precedence is crucial for maintaining an orderly and efficient flow of ships through the waterways. Figure 4 illustrates the diagram with the directed graph showcasing task dependencies.

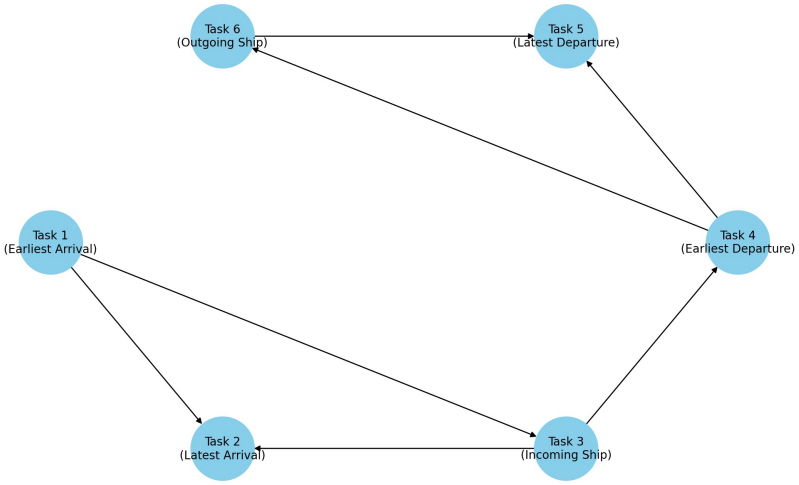

Figure 4: The diagram illustrates the sequence of tasks in a project, with arrows indicating the order of operations: Task 1 leads to Tasks 2 and 3, Task 3 to Task 4, and Task 4 to Tasks 5 and 6, with Task 6 also depending on Task 5.

The time horizon is structured such that each time step represents 15 minutes, translating to 96 steps for a full day and 188 for a two-day horizon. This time structure plays a critical role in the scheduling and management of maritime traffic, providing a clear and manageable framework for operations within the defined time horizons.

4 Methodology

We've developed a method to address the Waterway Ship Scheduling Problem using Graph Neural Networks by transforming it into a single-mode Resource-Constrained Project Scheduling Problem, inspired by the work of Hill et al. (2019). This transformation involves adapting WSSP variables to fit the single-mode RCPSP framework, including introducing vessel time windows, defining the project horizon, and establishing precedence relations between tasks. Our approach leverages the GNN model, as detailed in the RCPSP section and based on the findings of Teichteil-Königsbuch et al. (2023), to solve the problem.

The implementation of GNN in this context faces two primary challenges: customising the GNN to recognise critical relationships within the RCPSP and modifying the GNN-generated schedules to ensure feasibility. To address these challenges, we've reformulated the RCPSP to be GNN-compatible, with details provided in subsequent sections. This includes strategies for aligning GNN predictions with feasible solutions. A significant part of the methodology, heavily influenced by the findings of Teichteil-Königsbuch et al. (2023), involves utilising GNN to solve the RCPSP effectively. This approach leverages GNNs to mimic solutions akin to an exact Constraint Programming (CP) solver. The GNN model is trained using labels derived from CP-sat solver solutions, ensuring supervised training that is efficient and accurate.

Figure 5 outlines the steps and components of the training and inference processes for RCPSP. The training process of the GNN model involves batching GNNs for specific RCPSP problems and refining predictions through back-propagation. Following the training, the GNN predictions are adjusted to address any constraint violations by strategically ordering tasks and applying the Serial Schedule Generation Scheme for feasible schedule creation. This comprehensive approach aims to tackle the WSSP effectively by harnessing the capabilities of GNN within the structured framework of single-mode RCPSP.

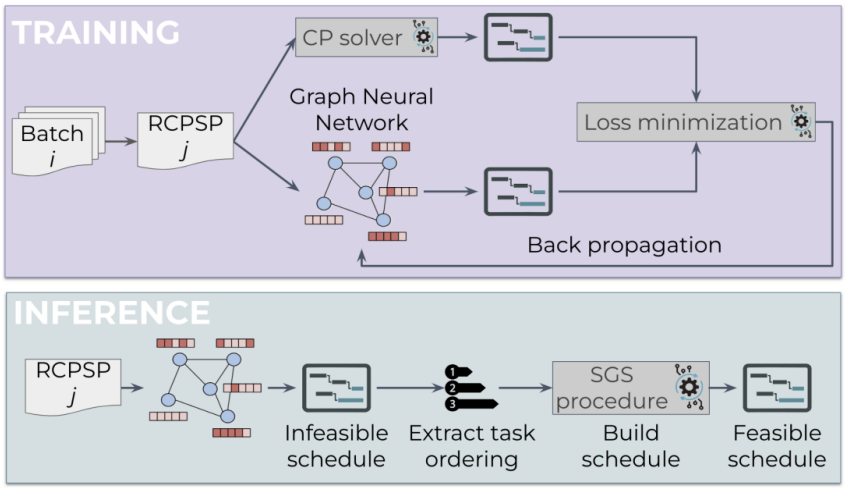

Figure 5: This diagram illustrates the training and inference processes for solving RCPSP problems using a Graph Neural Network. During training, the GNN learns from batches of RCPSP instances with guidance from a CP solver for loss minimisation through back-propagation. In the inference phase, the GNN's output is refined through a Serial SGS procedure to extract task ordering from an initially infeasible schedule, resulting in a feasible schedule that aligns with resource constraints and task dependencies (Teichteil-Königsbuch et al., 2023).

4.1 GNN Graph Representation of RCPSPs

Translating RCPSP into a graph representation is a pivotal step for analysis. This process systematically converts project tasks, their durations, resources, and precedence relations into a structured graph format. In such a graph, each node symbolises a distinct task, encapsulating crucial attributes like duration and resource requirements. Directed edges between these nodes represent the precedence relations, delineating the order in which tasks must be executed. This graphical portrayal simplifies understanding of complex task dependencies and resource allocations and lays the groundwork for applying advanced scheduling algorithms. Consequently, it becomes an indispensable tool for

optimising project timelines and resource utilisation in various industrial and academic applications. Before creating the GNN graph representation of RCPSPs, we visualised in Figure 6 a simple graph representation of RCPSP.

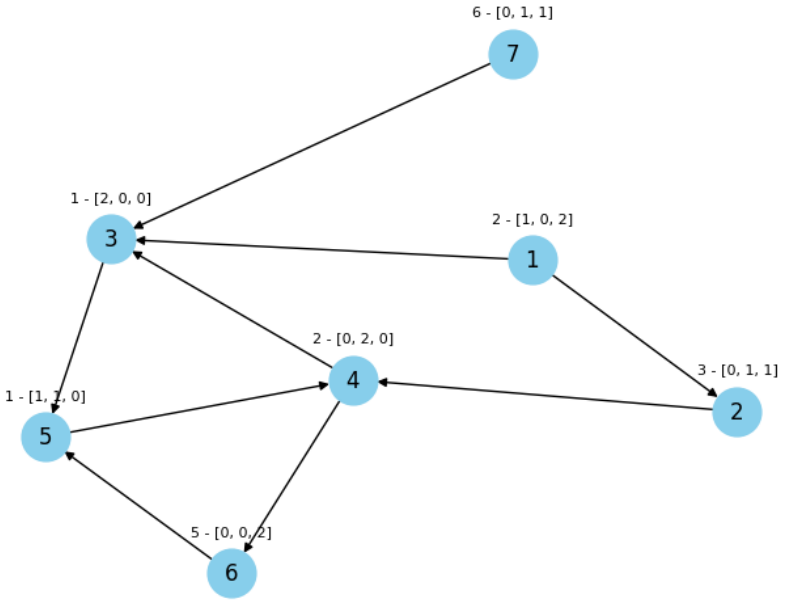

Figure 6: Visual graph representation of a simple RCPSP with seven tasks. The directed edges (arrows) between the nodes represent the precedence relations between tasks. For instance, an arrow pointing from Task 1 to Task 2 indicates that Task 2 can only commence once Task 1 has been completed. Each node in the graph corresponds to a specific task, uniquely numbered from 1 to 7. Above each node, additional information is displayed in square brackets, including the task duration followed by its resource requirements. For example, the label "2 - [1, 0, 2]" for Task 1 signifies that this task has a duration of 2 units of time and requires resources in the following configuration: 1 unit of Resource 1, 0 units of Resource 2, and 2 units of Resource 3.

To portray an RCPSP instance suitable for GNN, we utilise a graph defined as $G = (T, R, E, \mathbf{V}, \mathbf{E})$. Tasks ($T$) and resources ($R$) are distinct node types in this graph. The graph has three types of edges: precedence edges (E_P), resource demand edges (E_R), and parallel reverse links (E_{rev}). These edges capture relationships between tasks and resources, such as precedence constraints and resource demands. Edge features, represented by $\mathbf{e_{ij}} \in \mathbf{E}$, help encode resource consumption and differentiate among edge types. For instance, precedence edges have the feature vector $\mathbf{e_{ij}} = [1, 0, 0, 0, 0]$, while resource consumption edges follow the pattern $\mathbf{e_{ki}} = [0, 1, 0, 0, r_{i,k}]$. To represent task durations and available resources, we employ task and resource node features ($\mathbf{v_i} \in \mathbf{V_T}$ and $\mathbf{v_k} \in \mathbf{V_R}$, respectively). A visual representation of the RCPSP is depicted in Figure 7. Unlike edge features, node features evolve with each graph-Transformer layer. The above representation facilitates direct processing using established GNN models and converts an RCPSP into a graph format. It sets up variables for tasks and resources and creates mappings for task dependencies and resource use. It organises nodes for resources and tasks and crafts lists for tracking resource consumption and task lengths. It pinpoints start and end tasks and transforms the scheduling details into tensors. Node and edge features are then crafted to differentiate resources from tasks and to detail connections and resource use. The result is a data object filled with graph details and RCPSP specifics. A CP-sat solver incorporates Existing solutions with start times and project duration.

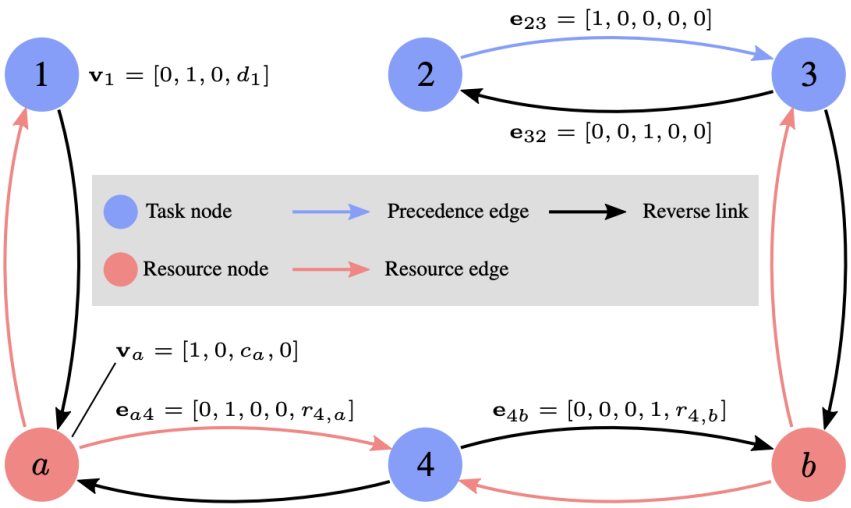

Figure 7: Visual representation of a GNN for Resource-Constrained Project Scheduling Problem, depicting task and resource nodes, along with the various types of edges such as precedence, resource, and reverse links, each annotated with their respective feature vectors to encapsulate the relationships and constraints within the scheduling problem. Note that all edges and nodes are annotated with features, but only some examples are shown here for simplicity (Teichteil-Königsbuch et al., 2023).

4.2 GNN Architecture

The GNN architecture utilised in our study incorporates a multi-layer message-passing mechanism based on the graph Transformer model, which integrates self-attention with graph neural networks for processing graph-structured data (Shi et al., 2020). This model computes node representations by aggregating neighbour features, weighted by attention scores, allowing it to handle variable-sized graphs and capture local and global structures.

Our implementation, the ResTransformer class in PyTorch, features a deep architecture with 15 blocks of Transformer-based convolution layers, integrating residual learning to facilitate gradient flow in deeper networks. Each layer uses a hidden size of 128 units, enhancing information flow across the graph. Node and edge input dimensions align with the previously described RCPSP graph representation.

The processing involves an initial adaptation of node inputs through a Transformer convolution using edge features. Information aggregation at each node is followed by sequential operations in residual blocks: normalisation, ReLU activation, Transformer convolution, and a residual connection. The forward pass scales input logarithmically processes them through the layers and normalises outputs for gradient stability. It concludes with a linear transformation to a binary output, scaled to an integer value.

The attention mechanism determines weights $\alpha_{i,j}$, with transformations relying on weight matrices $\mathbf{W_i}$. Each layer's residual connection assists in training, and the complete operations in a layer involve nonlinear transformations (ReLU), normalisation (ψ), and addition of the transformed output to the original node features, thus maintaining information flow and addressing vanishing gradients in deep networks.

4.3 Training

The training process for optimising scheduling tasks in RCPSP using a GNN involves constructing a batch of GNNs, each representing a different RCPSP. This batch is crucial for back-propagation during the weight update phases, enhancing the GNN's prediction accuracy. Each RCPSP instance has an associated optimal schedule computed using a CP solver. This optimal schedule consists of the start dates of each task and serves as a benchmark for the training process, which is conducted in a supervised manner. The GNN model is trained to replicate these optimal schedules, essentially learning from the ground truth provided by the CP solver's results. This supervised training approach enables the model to understand and predict the scheduling patterns that lead to optimal resource allocation and task sequencing.

This loss computation is essential for iterative learning and model fine-tuning through back-propagation. The model also rigorously checks constraints to validate start times, precedence, and resource constraints and to compute the makespan. Parameters are reinitialised before each fold in a K-Fold cross-validation scheme to ensure training integrity and avoid contamination from previous iterations. Metrics like constraint violations and makespan are monitored throughout each epoch, guiding the optimisation of model weights via the optimiser. The best-performing models are archived to retain the most efficient schedules, which is crucial for long-term effective scheduling outcomes.

The training utilised 1632 diverse instances from the PSPLIB (Sprecher and Kolisch, 1996), accounting for 80% of the total instances, over 20,000 epochs. These instances were labelled using the CP-SAT solver with a 15-minute timeout, although the methodology is independent of the specific RCPSP solver used for labelling.

As training progresses, the percentage of precedence violations may increase as the GNN learns to emulate CP-sat optimised schedules, potentially pushing against precedence constraints. However, this trend does not affect the adherence to resource constraints established from CP-sat solution benchmarks. As detailed in the subsequent section, the inference component is designed to fine-tune GNN predictions towards feasibility, as detailed in the subsequent section.

4.4 Inference

The output of the GNN is a schedule that minimises the makespan of a project given resource constraints. The GNN model is trained to generate start times for each task in a project that adheres to these constraints and precedence relations while minimising the makespan.

Throughout the training process, the percentage of precedence rule violations tends to rise as the GNN hones its ability to generate schedules with minimised makespans that mirror those refined by the CP solver. This upward trend reflects the model's increasing learning ability to optimise the project makespan, which can pressure the precedence relationships between tasks. Conversely, the GNN demonstrates progressive improvement in adhering to resource constraints, learning effectively from the schedules provided by the CP solver's solutions.

Observing minor deviations in the GNN's predictions during the training process indicates a necessity for adjustments in these predictions to achieve a practical scheduling outcome. In the inference phase of scheduling using a GNN, we focus on generating feasible schedules for the RCPSP. The methodology primarily utilises the Serial Schedule Generation Scheme (SSGS), with a detailed contrast in Kolisch and Hartmann (1999). SSGS, crucial for creating feasible RCPSP schedules, prioritises tasks based on heuristic rules or optimisation algorithms, sequentially allocating resources to tasks while adhering to resource capacities and precedence relations.

In practical application, SSGS begins with a provisional schedule (dummy solution) as a baseline, progressing to compute a feasible solution based on the initial task order. This process involves recording the makespan and starts times and integrating these into a dictionary for comprehensive evaluation, ensuring the feasibility of the schedule in terms of resource constraints and task dependencies.

The algorithm (Algorithm 1) extracts task ordering from these predictions and employs SSGS to convert the ordering into a feasible schedule, ensuring resource and constraint adherence. The resultant schedule, a combination of GNN predictions and SGS refinements, represents a feasible and high-quality solution for the RCPSP.

The algorithmic process starts by building a GNN for the specific RCPSP instance using pre-trained weights and then predicting a solution consisting of task start times. Task ordering is derived from these predictions, and SSGS is applied to generate a feasible schedule, with the final output being a feasible solution schedule for the given RCPSP. The dictionary encapsulates these results, providing a comprehensive view of the scheduling system's efficacy, including makespan and start time for each task.

Algorithm 1 Scheduling with Trained GNN Weights (Teichteil-Königsbuch et al., 2023)

Input: Given RCPSP P with unknown solution; GNN trained weights $\mathbf{W}$
Output: Solution schedule x of P

1: Build GNN $G_{\mathbf{W},P}$ from P and using weights $\mathbf{W}$
Predict from $G_{\mathbf{W},P}$ the solution $\hat{\mathbf{x}}$ of P
Extract tasks ordering $O_{\hat{\mathbf{x}}}$ from inferred starting dates $\hat{\mathbf{x}}$
Construct solution schedule x^* by running SGS on $O_{\mathbf{x}^*}$
return x^*

5 Data Description

Our study's methodology includes generating datasets through three distinct approaches: random generation, structured non-random generation, and concluding with an analysis of marine traffic data applied to a case study at the Port of Duisburg employing the GNN model for complex scheduling scenarios. Before discussing the datasets in detail, we will first outline the training dataset of our GNN model, which utilised the PSPLIB set.

Training A standardised data set is crucial for robustly evaluating different heuristic outcomes in RCPSP. The PSPLIB repository, introduced by Kolisch and Hartmann (1999), is a comprehensive benchmark for various RCPSP instances, including multi-mode and single-mode formulations. The single-mode instances restrict tasks to one specific resource.

The PSPLIB comprises diverse instance types and sizes and is a cornerstone for testing exact and heuristic methods in RCPSP research. It offers four distinct sizes in the single-mode RCPSP: *j30, j60, j90, and j120*, representing 30, 60, 90, and 120 activities respectively. These sizes equate to scenarios involving between 5 and 30 ships. The respective instance counts for these sizes are 480, 480, 480, and 600. Each instance has varying parameters, such as network complexity and resource intensity.

For the study's scope, the model is trained on a set from the PSPLIB, retaining other instances for validation and testing. In total, the PSPLIB provides 2040 varied instances. Of these, 1632 instances (80%) spanning different sizes and structures are designated for training, ensuring a comprehensive assessment of the model's capability. The optimal makespan is known and listed in the library for all of these instances.

Randomly Generated Data: The data were created using a randomised approach with specific parameters. The ship's width and draught determined the allocation of each ship to a particular waterway. A ship would be assigned to a waterway only if the waterway's width could accommodate the ship's dimensions. This simulation reflects the real-world constraint that a ship cannot navigate a waterway too narrow for its size. The limit on the number of ships that could enter or leave simultaneously, termed as 'Qlimit', was set to either one or ten. This parameter mirrors the practical limitations of waterway traffic management. Furthermore, the total capacity of each waterway, described in terms of width and depth, was randomly determined for each scenario. The dimensions of the ships (length, width, and draught) were randomly generated within specific ranges. Width varied from 1 to 100 metres, draught from 1 to 10 units, and length from 1 to 20 metres. Additionally, each ship's arrival and departure times were randomly chosen from a range of 1 to 15 time steps.

Table 1 presents data configurations for various scenarios, distinguished by the number of waterways and ships involved. Two primary scenarios are considered: one with two waterways using ten resources and another with four waterways using 20 resources. Ship scenarios range from 5 to 30 ships, with corresponding tasks varying accordingly. For example, a scenario with five ships involves 30 tasks, noted as "w2v5" for two waterways and "w4v5" for four waterways. Each dataset generated five distinct RCPSP problems with a Qlimit of 1, resulting in a total of 60 RCPSP problems. Additional scenarios with up to 175 ships and six waterways expand this dataset, testing more complex environments with Qlimits of 1 and 10, leading to 36 additional RCPSP problems.

Table 1: Combined Data for Scenarios with different numbers of waterways and ships, specifying the total tasks and the representation for scenarios with two, four and six waterways, accompanied by remarks on the distribution of vessels and Qlimits

Ships	**Total Tasks**	**2 Waterways (w2) representation** 10 resources	**4 Waterways (w4) representation** 20 resources	**6 Waterways (w6) representation** 30 resources	**Remarks**
5	30	w2v5	w4v5	-	$\|V\| = 10$, $\|V_{\text{in}}\| = 5$, $\|V_{\text{out}}\| = 5$ Qlimit = 1
10	60	w2v10	w4v10	-	$\|V\| = 20$, $\|V_{\text{in}}\| = 10$, $\|V_{\text{out}}\| = 10$ Qlimit = 1
15	90	w2v15	w4v15	-	$\|V\| = 30$, $\|V_{\text{in}}\| = 15$, $\|V_{\text{out}}\| = 15$ Qlimit = 1
20	120	w2v20	w4v20	-	$\|V\| = 40$, $\|V_{\text{in}}\| = 20$, $\|V_{\text{out}}\| = 20$Qlimit = 1
25	150	w2v25	w4v25	-	$\|V\| = 50$, $\|V_{\text{in}}\| = 25$, $\|V_{\text{out}}\| = 25$ Qlimit = 1
30	180	w2v30	w4v30	-	$\|V\| = 60$, $\|V_{\text{in}}\| = 30$, $\|V_{\text{out}}\| = 30$ Qlimit = 1
50	300	w2v50	w4v50	w6v50	$\|V\| = 100$, $\|V_{\text{in}}\| = 50$, $\|V_{\text{out}}\| = 50$, Qlimit = 1 or 10
75	450	w2v75	w4v75	w6v75	$\|V\| = 150$, $\|V_{\text{in}}\| = 75$, $\|V_{\text{out}}\| = 75$, Qlimit = 1 or 10
100	600	w2v100	w4v100	w6v100	$\|V\| = 200$, $\|V_{\text{in}}\| = 100$, $\|V_{\text{out}}\| = 100$, Qlimit = 1 or 10
125	750	w2v125	w4v125	w6v125	$\|V\| = 250$, $\|V_{\text{in}}\| = 125$, $\|V_{\text{out}}\| = 125$, Qlimit = 1 or 10
150	900	w2v150	w4v150	w6v150	$\|V\| = 300$, $\|V_{\text{in}}\| = 150$, $\|V_{\text{out}}\| = 150$, Qlimit = 1 or 10
175	1050	w2v175	w4v175	w6v175	$\|V\| = 350$, $\|V_{\text{in}}\| = 175$, $\|V_{\text{out}}\| = 175$, Qlimit = 1 or 10

In the extended analysis addressing the RCPSP, our approach involved generating challenging models by creating instances based on complex precedence relations. This was executed in two segments: random data generation and case study. The added precedence relations introduce more complex relationships between tasks. This pattern not only includes inter-task relations for each ship but also integrates cross-ship dependencies. For example, task 4 (earliest departure) might have to precede not only its immediate subsequent tasks but also tasks related to the other two ships (e.g., task 10 (earliest departure) and Task 15 (latest arrival) from different ships). This approach increases the complexity and interconnectivity of the task dependencies, reflecting a more intricate scheduling challenge where tasks from different ships influence one another. This increased complexity aligns with more realistic scenarios in busy ports like Duisburg, where the scheduling involves managing multiple ships with overlapping operations. It represents an advanced level of problem-solving, where the GNN model must account for a broader range of dependencies and interactions among tasks, making the scheduling problem more challenging yet more reflective of real-world conditions.

For the randomly generated data, we constructed a total of 12 instances of a large number of ships. These instances were divided into two categories: six instances involved managing 200 ships, and the other six dealt with 400 ships. Each of these instances was further tested under varying conditions: Qlimit was set to 1, 10, and 20 to understand the impact of different congestion levels, and the number of waterways was alternated between 2 and 4 to simulate varying levels of navigational complexity (see Table 2).

Table 2: Extended data used for the randomly generated data with extra precedence relations between ships showcasing different conditions such as Qlimit variations and waterway options to simulate complex operational environments.

Ships	**Tasks**	**2 Waterways**	**4 Waterways**	**Remarks**
200	1200	w2v200Q1	w4v200Q1	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 1
200	1200	w2v200Q10	w4v200Q10	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 10
200	1200	w2v200Q20	w4v200Q20	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 20
400	2400	w2v400Q1	w4v400Q1	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 1
400	2400	w2v400Q10	w4v400Q10	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 10
400	2400	w2v400Q20	w4v400Q20	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 20

Non-Random Generated Data: Non-random generated data is drawn from a detailed table of European inland shipping fleet specifications (de Vries, 2015). Table 3, based mainly on source de Vries (2015), offers vessel specifications spanning across various vessel types and classes. Each vessel is categorised by its CEMT-Class and RWS-Class, alongside other essential dimensions like length overall (LOA), width, and draught. For instance, the Spits (Peniche) vessels, classified as M1, measure 38.5 metres in length, 5.05 metres in width, and have a draught of 2.5 metres. On the larger end of the spectrum, we find the Push convoys with 2x3 bins, classified as VIIa. These massive vessels span 195 metres in length and 34.2 metres in width and have a draught between 3.5 and 4.0 metres, capable of carrying a whopping 14,500 to 27,000 tons. Such detailed insights into the European inland shipping fleet proved invaluable, allowing us to simulate and understand the complexities of maritime operations accurately. This table, in essence, offered a comprehensive framework, ensuring that our datasets are better and more reflective compared to randomly generated data. In addition, a comprehensive overview of the widths and depths of various Dutch inland waterways is listed in the report of Infrastructure and management (2022). From the report, we noticed the waterway with the maximum width is the "Westerschelde," spanning between 2000 to 8000 metres, while the "Wilhelminakanaal" and "Zuid-Willemsvaart" are among the narrower waterways with widths ranging from 12 to 48 metres. In terms of depth, the "Westerschelde" again stands out with a maximum depth of 51.44 metres below sea level, while waterways like the "Noordervaart" have depths of around 2.25 metres below sea level. On average, the waterways tend to be quite broad, reflecting the expansive nature of the Dutch inland water system, with depths varying significantly, catering to ships of different sizes. It is evident that these waterways are designed to accommodate a wide range of vessel sizes, ensuring smooth navigation and transport across the region. This information is crucial to generating more accurate data sets.

The report of Infrastructure and management (2022) details the widths and depths of Dutch inland waterways, complementing the vessel specifications in Table 3 based on de Vries (2015). These sources provided a robust foundation for generating more realistic data as an RCPSP problem.

Table 3: Specifications of the European Inland Shipping Fleet Based on CEMT and RWS Classification, mainly based on (de Vries, 2015).

CEMT-Class	RWS-Class	Vessel type	Length (LOA) [m]	Width [m]	draught [m]
I	M1	Spits (Peniche)	38.5	5.05	2.5
II	M2	Campine vessel (Kempenaar)	50-55	6.6	2.6
III	M3	Hagenaar	55-70	7.2	2.6
III	M4	Dortmund-Ems canal vessel	67-73	8.2	2.7
III	M5	Elongated Dortmund-Ems canal vessel	80-85	8.2	2.7
IV	M6	Rhine-Herne canal vessel	80-105	9.5	2.9
IV	M7	Elongated Rhine-Herne canal vessel	105	9.5	3
Va	M8	Large Rhine vessel	110	11.4	3.5
Vb	M9	Elongated large Rhine vessel	135	11.4	4
Vb		Push convoy with 1x2 longitudinal bins	170-190	11.4	3.5-4.0
VIa	M10	Two lighter push units	110	13.5	4
VIa	M11	Gauge vessel	135	14.2	4
VIa	M12	Rhine max vessel	135	17	4
VIb	-	Push convoy with 2x2 bins	185-195	22.8	3.5-4.0
VIC	-	Push convoy with 3x2 bins	270	22.8	3.5-4.0
VIIa	-	Push convoy with 2x3	195	34.2	3.5-4.0

Case Study - Port of Duisburg: The choice of the Port of Duisburg for the case study in maritime scheduling is highly significant, given its status as the largest inland port in Europe. This port serves as a crucial logistical and transportation hub, with a vast network of waterways connecting it to various regions and industries. Strategically located at the confluence of the Rhine and Ruhr rivers, the Port of Duisburg is a pivotal junction for inland waterway transport, linking major European shipping routes.

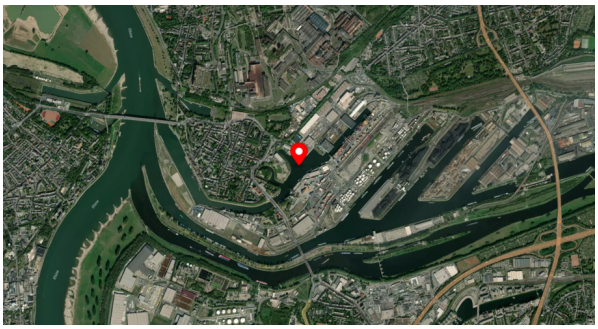

Figure 8: Aerial view of the Port of Duisburg, illustrating its vast infrastructure and strategic location at the heart of Europe's inland waterways. The confluence of the Rhine and Ruhr rivers is evident, underscoring the port's importance as a major hub for maritime commerce and logistics.

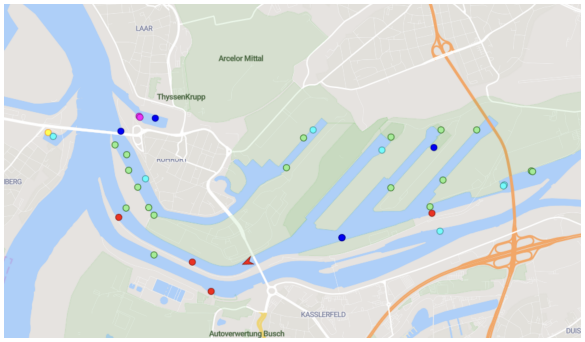

Figure 9: This map showcases the intricate network of ship movements within the Port of Duisburg, captured through AIS data from MarineTraffic. Each coloured dot represents the dynamic positioning of vessels, highlighting the port's role as a bustling nexus of European inland waterway transport.

This study leverages real-world data derived from MarineTraffic, a leader in maritime analytics provider offering real-time ship movement information and location in harbours and ports. The Automatic Identification System (AIS) data from MarineTraffic provides detailed insights into the maritime traffic at the Port of Duisburg. This data includes the arrival and departure times of approximately 200 ships daily, reflecting the high volume and diversity of maritime activities at the port.

For the case study, instances for 200 and 400 ships were meticulously crafted to reflect the high traffic volume at the port. These instances incorporated diverse ship dimensions, including length, width, and draught, extracted directly from MarineTraffic's data. This approach ensures a realistic representation of different vessel types navigating through the port.

In addition to capturing ship dimensions, the study carefully considered the Qlimit - the number of ships that can simultaneously enter or leave the port. Two Qlimit scenarios were examined: a lower limit of one, offering a direct

comparison with other dataset types, and a more realistic higher limit of ten, aligning closely with the actual operational capacity at Duisburg.

The Port of Duisburg's strategic significance as a major European logistics hub is further highlighted by its connection to four major waterways. Several key waterways are connected to the Port of Duisburg, including the Rhine-Herne Canal, the Duisburg-Ruhrorter Ports, and other channels that link to the broader European waterway network. In this study, scenarios involving both two and four waterways were generated. This extensive connectivity makes the port an ideal real-world scenario for studying complex scheduling and logistical challenges in maritime traffic.

The datasets for this case study were constructed over a two-day time horizon, aligning with the typical operational timescales observed at the port. This temporal framing provided a comprehensive view of the port's activities, capturing the intricacies and challenges of scheduling in a bustling maritime environment.

Table 4 provides an overview of the instances generated for the Port of Duisburg, detailing the scenarios with 200 and 400 incoming and outgoing ships under different Qlimit constraints. Two RCPSP problems, "w4v200" and "w2v200," were constructed to simulate operations involving 200 incoming and outgoing ships, each with Qlimits of 1 and 10. Additionally, instances simulating 400 incoming and outgoing ships under different waterway conditions and Qlimits were created, testing the scalability and adaptability of scheduling algorithms under increased operational intensity.

Overall, the Port of Duisburg case study, focusing on realistic and varied ship traffic, offers a robust platform for testing the efficacy of scheduling algorithms. It reflects the complexities of real-world maritime logistics, providing valuable insights into optimising port operations under varying conditions and constraints. The insights gained from this case study have practical implications for improving the efficiency and effectiveness of port operations in one of Europe's busiest inland waterway hubs.

Table 4: Extended data detailing the scenarios for the Port of Duisburg case study, with variations in ship numbers, tasks, and Qlimit constraints for different waterway configurations.

Ships	**Tasks**	**2 Waterways**	**4 Waterways**	**Remarks**
200	1200	w2v200Q1	w4v200Q1	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 1
200	1200	w2v200Q10	w4v200Q10	$\|V\| = 400$, $\|V_{\text{in}}\| = 200$, $\|V_{\text{out}}\| = 200$, Qlimit = 10
400	240	w2v400Q1	w4v400Q1	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 1
400	2400	w2v400Q10	w4v400Q10	$\|V\| = 800$, $\|V_{\text{in}}\| = 400$, $\|V_{\text{out}}\| = 400$, Qlimit = 10

Additionally, our study incorporated an extended analysis of the case study involving integrating additional precedence relations, as described in the randomly generated data subsection. The objective here was to assess the capability of the GNN model in solving complex scheduling problems compared to traditional OR Tools. We applied this to scenarios with 200 vessels, in line with the previously mentioned test setups (see Table 4). The only difference lies in more advanced precedence relations.

6 Results

In the results section, we present the outcomes of our study, which aimed to evaluate the effectiveness of GNNs in the scheduling of inland waterway shipping. The goal was to assess whether GNNs could provide accurate and efficient solutions for the WSSP when translated into the RCPSP framework. The study compared the performance of GNNs against conventional CP-sat approaches across a variety of scenarios involving different numbers of waterways and ships. In our study of the WSSP, we compared the GNN and CP-sat in terms of Computational Efficiency and Solution Quality. We analysed various scenarios with varying numbers of waterways (two, four, and six) and ships (from 5 to 175). Using randomly generated data, the study evaluated the GNN model's accuracy against optimal makespan values. We then shifted to non-randomly generated data, focusing on real realistic ship and waterway dimensions. Afterwards, a case study of the Port of Duisburg will be conducted to test the GNN model against OR Tools. In the case study of the Port of Duisburg, using GNNs facilitates rapid and scalable scheduling solutions to manage its high-traffic volume effectively. This represents a significant improvement over traditional CP solvers, offering a more dynamic adaptation to the port's complex shipping activities, which is particularly advantageous in Europe's largest inland port. Further, we expanded the analysis with more complex scenarios, increasing waterways and tasks and varying the number of ships to assess the GNN model's robustness under diverse operational conditions. The outcomes were compared with OR Tools to gauge the GNN model's overall performance. The code is implemented in Python within a Visual Studio Code environment utilising an 8-core Apple M1 CPU @3.20Ghz, 16GB RAM and an 8-core GPU. We used a suite of libraries, including PyTorch, PyTorch Geometric, optimisation tools (OR tools), and scikit-decide.

6.1 Training

This assessment focused on the GNN model's predictive performance on a test set of 408 instances that reflected the training set regarding vessel numbers, tasks, and available resources. The aim was to gauge the model's generalisation capabilities by comparing the time it takes for GNN and CP-SAT to find solutions post-training. The evaluation of the GNN model demonstrated that it often exceeded the computational efficiency of CP-SAT, being faster in 81% of the test cases. Specifically, the results highlighted the GNN's capacity to deliver solutions of similar quality to CP-SAT within a reduced timeframe.

The comparison involved two key aspects. Initially, it accounted for the GNN's solution prediction time after training once the GNN model was evaluated against the time CP-SAT requires to achieve the same quality of makespan. Noteworthy is that in 40% of the cases, the computation time of CP-SAT was at least ten times longer than the GNN model, occasionally reaching a differential of up to 10,000 times. Nonetheless, there were scenarios where CP-SAT was twice as fast as the GNN.

Furthermore, a comparison of solution quality was made within a fixed 5-minute runtime for both the CP solver and the GNN model predictions. Here, the GNN model exhibited impressive performance, matching the optimal solutions in half of the cases and producing makespans on average less than 5% longer than those of CP-SAT. These results are depicted in Figure 11.

This superior performance of the GNN model is illustrated in Figure 10 and Figure 11, where instances above the diagonal line indicate that GNN outperformed CP-sat in terms of speed. The mean relative performance of the GNN, at 0.93, suggests that, on average, its efficiency is comparable to, and occasionally exceeds, CP-sat. This indicates that while the GNN model may slightly drop in performance in some cases, it generally delivers commendable outcomes closely aligning with those of the CP-sat solver.

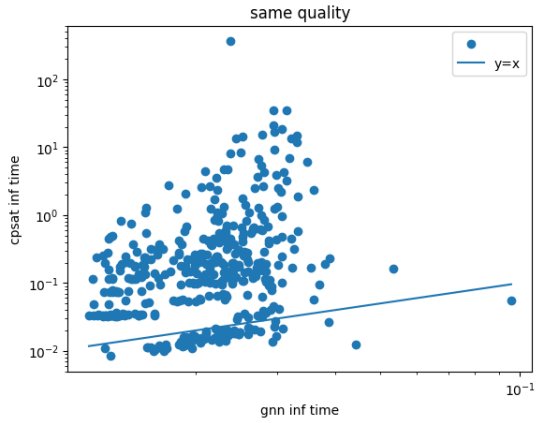

Figure 10: Scatterplot illustrating the comparison between GNN and CP-sat computation time with the same quality. Points on the diagonal indicate identical makespans, while those above the diagonal show the GNN computation time surpassing CP-sat.

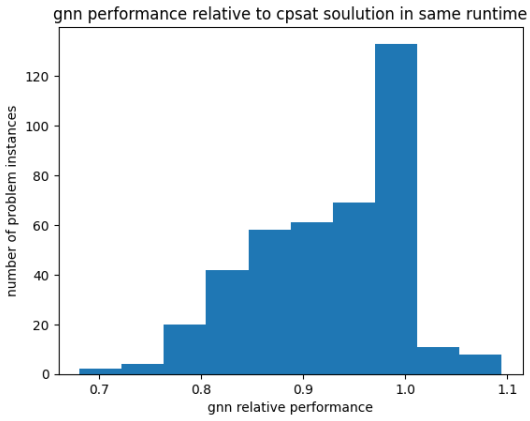

Figure 11: Graphical representation of GNN's relative performance compared to CP-sat solution in the same runtime over 408 problem instances.

6.2 Random Generated Data

In our analysis of the WSSP problem using randomly generated data, we examined scenarios with different numbers of waterways (2 or 4) and ships (ranging from 10 to 60). Further details have been provided in subsection 5. The optimal solutions for these instances were obtained using a CP solver from OR-Tools. The makespan values, representing the average from five unique RCPSP problems per scenario, were compared between optimal solutions and GNN predictions. The results are outlined in Table 5.

Key observations from the data reveal that the GNN model often closely matched or slightly exceeded the optimal makespan values. For example, in a scenario with ten ships and four waterways, the optimal average makespan was 24.6, compared to the GNN's 26.7. Similar trends were observed across various scenarios. In another case with 60 ships and two waterways (labelled "w2v60"), the optimal makespan was 109.2, while GNN reported 116.2.

The optimality gap, indicating the percentage difference between optimal solutions and GNN predictions, was often below 5%. This demonstrates the GNN model's high accuracy, aligning closely with optimal solutions in many instances. As the number of ships increases, the optimality gap tends to decrease. This suggests the model's heightened efficiency in managing larger fleets, particularly in less complex two-waterway scenarios. Conversely, in scenarios involving four waterways, the results show more variability, although most remain within a 7% optimality gap, with an outlier at 15%. This indicates a less consistent performance in more intricate, four-waterway environments.

Additionally, when comparing scenarios based on the number of waterways and ships, the GNN model performs better in four waterway scenarios with fewer ships. At the same time, it excels in two-waterway scenarios with a larger number of ships. For example, in a scenario with 30 ships and four waterways, the GNN model achieved a 0% gap, perfectly matching the optimal makespan. This contrast underscores the model's adaptability to the complexity of the scheduling environment. Moreover, a general pattern emerges where larger makespan values correlate with lower optimality gaps, suggesting that the GNN's predictions are more aligned with optimal solutions in scenarios with longer project durations.

Overall, these findings highlight the GNN model's effectiveness in optimising the WSSP. In several scenarios, the GNN model delivered results comparable to or slightly divergent from the benchmarks set by optimal values, indicating its potential as an efficient tool for complex scheduling challenges. The narrow optimality gap across various scenarios further underscores the GNN model's accuracy and efficiency in addressing the WSSP problem.

Table 5: Performance Evaluation of the GNN model on Randomly Generated Data for the RCPSP. Each row represents the average outcomes from five instances of the RCPSP. The optimality gap percentage indicates the GNN's performance relative to the optimal solutions across different numbers of waterways and ships.

Problem						**Makespan**		
WSSP				**SM-RCPSP**		**Optimal**	**GNN**	
$\|V\|$	$\|V_{in}\|$	$\|V_{out}\|$	$\|W\|$	$\|J\|$	$\|R\|$	**makespan**	**Prediction**	**Optimality Gap**
10	5	5	2	30	10	47	54.4	**16%**
10	5	5	4	30	20	24.6	26,4	**7%**
20	10	10	2	60	10	65.2	73.6	**13%**
20	10	10	4	60	20	40.6	40.6	**0%**
30	15	15	2	90	10	96.8	103.8	**7%**
30	15	15	4	90	20	64.2	64.2	**0%**
40	20	20	2	120	10	126	130.2	**3%**
40	20	20	4	120	20	63	72.2	**15%**
50	25	25	2	150	10	164.6	169.2	**3%**
50	25	25	4	150	20	97.4	104.6	**7%**
60	30	30	2	180	10	207	215.2	**4%**
60	30	30	4	180	20	109.2	116.2	**7%**

In our analysis of more extensive RCPSP instances, we examined various scenarios with different quantities of ships and waterways, incorporating Qlimit constraints of 1 or 10. We focused on comparing the GNN model's performance against OR Tools solutions within a maximum computation time of 15 minutes. The scenarios ranged from 50 to 175 ships navigating through two, four, or six waterways.

The GNN model displayed impressive consistency, particularly in scenarios with two waterways and a Qlimit of 1. It exhibited minimal standard deviation in makespan predictions and a mean relative time average that showcased its computational efficiency. A high relative time average indicates the efficiency of the GNN model in comparison to OR Tools. Specifically, for configurations with two waterways (W2) and a Qlimit of 1, the GNN model predicts, on average, 1.34 times slower than OR Tools, demonstrating a disadvantage in computational speed. Note that the values found by OR tools for "W2Q1" are found as optimal values. Therefore, these instances seem too simplistic for the GNN to match the same computational speed.

In scenarios with a Qlimit of 10, the GNN model's performance varied more, as indicated by higher standard deviations and a wider range of optimality gaps. Despite these challenges, the GNN model maintained competitive performance, with optimality gaps within a 6% or 2% margin, suggesting resilience and robust predictive capabilities.

Notably, in extensive configurations involving six waterways, the GNN model's consistency and accuracy were highlighted by low standard deviations and a 2% average optimality gap compared to OR tools. Moreover, the GNN

model predicted outcomes significantly faster—up to 6.53 times quicker than OR tools, which is particularly valuable in large-scale and time-sensitive operational contexts.

However, across all configurations, the GNN model did not outperform OR Tools when the latter had a full 15-minute computation time. While the GNN model is trained once and then delivers fast predictions, its performance, in terms of optimality, generally remains close to or slightly below that of OR Tools, even in large cases with up to 175 ships.

In conclusion, the GNN model has shown robust performance across various configurations, efficiently handling scenarios with two, four, and six waterways under both Qlimit constraints. Its rapid predictive ability and close-to-optimal results makes it a practical tool for real-time or near-real-time decision-making in complex maritime traffic scheduling scenarios.

Table 6: Summary statistics for optimality gap configurations with varying Qlimits and varying number of ships and waterways. The relative time ratio column reflects the GNN model's computational efficiency relative to OR Tools for equivalent makespan quality predictions.

Waterway	Qlimit	Min	Max	Average	Median	Std Dev	Relative Time
W2	1	1.01	1.02	1.01	1.01	6.54e-03	0.74
W2	10	1.03	1.10	1.06	1.05	2.72e-02	1.33
W4	1	1.01	1.03	1.01	1.01	6.54e-03	0.84
W4	10	1.00	1.05	1.02	1.02	1.90e-02	6.53
W6	1	1.02	1.03	1.02	1.02	5.58e-03	3.67

In our analysis of RCPSP with more complex precedence relations between ships, we examined various scenarios with ship numbers of 200 or 400 and waterways, incorporating Qlimit constraints of 1, 10 or 20. As evidenced in Table 7, the GNN model's performance is notable, especially when it outperforms OR Tools' predictions within the same computational time frame.

In scenarios with a Qlimit of 1, such as "w2v200Q1" and "w4v200Q1", the GNN model exhibits its prowess by providing predictions that closely match or surpass the solutions obtained by OR Tools. Remarkably, the GNN model achieves these results with significantly shorter computation times, highlighting its efficiency.

The improvement is even more pronounced with a Qlimit of 10 and 20. For instance, in w2v200Q10 and w4v200Q10, the GNN model competes with OR Tools' predictions and needs less time for the same solution quality. This indicates the GNN model's potential to efficiently solve more complex instances where traditional methods may struggle to find optimal solutions promptly.

When examining instances with 400 ships, OR Tools consistently found optimal values within a maximum time limit of 5 seconds. This observation suggests that an increase in the number of ships doesn't necessarily lead to improved results for instances predicted by the GNN model. In fact, for all instances evaluated, OR Tools consistently outperformed the GNN predictions.

The GNN model demonstrates its capability to provide accurate and efficient predictions and exhibits a remarkable average relative time performance improvement. For the instances with 200 ships on average, the GNN makespan prediction outperforms OR Tools by 28.4 in terms of relative computation time efficiency. This significant margin underscores the GNN model's potential as a more time-effective solution, particularly in complex scheduling scenarios with increased precedence relations and stringent Qlimits.

Table 7: Computation time and predictions output of the extensive data analysis with extra precedence relations for the randomly generated data. Optimal values are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools. A dash (-) indicates the optimal solution was already found with a smaller computation time.

Instance	**GNN**		**OR Tools prediction after computation time in sec**								
	Prediction	**Computation time**	**0.2**	**0.5**	**2**	**5**	**30**	**120**	**300**	**600**	**900**
w2v200Q1	611	1.95 sec	615	613	610	-	-	-	-	-	-
w4v200Q1	417	1.99 sec	**446**	**441**	**428**	**428**	**417**	411	410	410	410
w2v200Q10	315	1.90 sec	**343**	**333**	**318**	**317**	**317**	312	310	309	308
w4v200Q10	401	1.98 sec	**422**	**417**	**410**	**407**	390	388	387	384	
w2v200Q20	314	1.97 sec	**343**	**335**	**318**	**317**	**315**	312	310	309	309
w4v200Q20	402	1.99 sec	**422**	**417**	**411**	**408**	393	388	385	384	384
w2v400Q1	1236	1.47 sec	1230	1223	1222	-	-	-	-	-	-
w4v400Q1	556	1.46 sec	**567**	**556**	546	-	-	-	-	-	-
w2v400Q10	133	1.49 sec	**139**	132	125	123	-	-	-	-	-
w4v400Q10	83	1.48 sec	76	60	55	-	-	-	-	-	-
w2v400Q20	87	1.46 sec	74	63	62	-	-	-	-	-	-
w4v400Q20	50	1.44 sec	41	-	-	-	-	-	-	-	-

6.3 Non-Random Generated Data

In analysing non-randomly generated data for the WSSP problem, we observed a general trend where the GNN model closely followed the optimal solutions, although with some deviations, especially in more complex scenarios. The evaluation encompassed scenarios with varying numbers of ships (10 to 40) and waterways (2 or 4), as detailed in Table 8.

The GNN model demonstrated commendable performance, with its optimisation values slightly exceeding the optimal values in certain instances. For example, with ten ships and two waterways, the GNN model had an average makespan of 20.6, reflecting an 8% gap from the optimal. In scenarios with more ships, such as 30 ships navigating two waterways, the GNN's average makespan was 42.2, indicating a 20% gap. Meanwhile, for four waterways (w4), the optimality gap consistently remained around 8%, indicating a more stable performance.

As the number of ships and waterways increased, the GNN model's optimality gap widened slightly, suggesting challenges in handling more complex configurations. A key observation is that the results were generally better for four waterways than those for two waterways. For instance, in a scenario with 40 ships and four waterways, the GNN model exhibited a 9% gap. This suggests an increased efficiency of the GNN model in more complex scenarios involving a larger number of waterways. Additionally, as the size of the ships increased, there was a slight decrease in the optimality gap, showcasing the model's adaptability to scenarios with larger fleets.

Comparatively, the optimality gap in non-randomly generated data scenarios was lower than in those with randomly generated data. This improvement could be attributed to the more realistic nature of the non-randomly generated data, resulting in lower makespan values. It's important to note that the GNN model predictions can never exceed the optimal values, and the scenarios were consistently set with a Qlimit of one, as previously mentioned in the data section.

In summary, the non-random data analysis revealed the GNN model's robustness in addressing the WSSP problem. While maintaining close proximity to optimal makespan values, the GNN model showed higher variability in scenarios involving a larger number of ships and waterways. These results highlight the GNN model's potential as an effective tool for complex scheduling challenges, although indicating areas for further refinement and optimisation.

Table 8: Performance Evaluation of the GNN model on non-randomly generated data for the RCPSP. Each row represents the average outcomes from five instances of the RCPSP. The scenarios vary by the number of ships and waterways, and the table illustrates both the predicted makespan by the GNN and the optimality gap, which measures the deviation of the GNN predictions from the optimal values.

Problem						**Makespan**		
WSSP				**SM-RCPSP**		**Optimal**	**GNN**	
$\|V\|$	$\|V_{in}\|$	$\|V_{out}\|$	$\|W\|$	$\|J\|$	$\|R\|$	**makespan**	**Prediction**	**Optimality Gap**
10	5	5	2	30	10	19	20.6	**8%**

Continued on next page

Table 8 – continued from previous page

Problem						Makespan		
WSSP				SM-RCPSP		Optimal	GNN	
$\|V\|$	$\|V_{in}\|$	$\|V_{out}\|$	$\|W\|$	$\|J\|$	$\|R\|$	makespan	Prediction	Optimality Gap
10	5	5	4	30	20	16.8	18.6	**10%**
20	10	10	2	60	10	32.2	35.4	**9%**
20	10	10	4	60	20	27.4	29.8	**8%**
30	15	15	2	90	10	38.6	42.2	**9%**
30	15	15	4	90	20	34.8	37.8	**8%**
40	20	20	2	120	10	62.6	75.2	**20%**
40	20	20	4	120	20	43.2	46.8	**9%**

6.4 Case Study - Port of Duisburg

The waterway operations of the Port of Duisburg are characterised by their complexity and scale, with instances involving up to 400 ships, significantly larger than the model's training data. The case study considers a realistic scenario with a Qlimit of 10 and four waterways, mirroring the actual Port of Duisburg and its variations, as explained in the data description. Moreover, the scheduling intricacies involve not only individual ship operations but also coordination among ships. Coordinated scheduling is essential to ensure smooth traffic flow and dock availability, where one ship's departure depends on another's arrival. Resource allocation is another critical aspect, efficiently distributing shared resources like tugs, pilots, and dock space among all vessels. These intricacies are modelled as extra precedence relations and described in the data section. In essence, the WSSP in the Port of Duisburg encompasses optimising overall port efficiency, reducing waiting times, and maintaining high throughput, all while adhering to safety and environmental standards and ensuring that their interdependent operations are harmonised.

In analysing the Port of Duisburg, we compared the predictions made by the GNN model against those computed by OR Tools in different RCPSP instances. These instances varied in waterway configurations (W2 and W4), Qlimit constraints (1 and 10), and the number of tasks (1200 and 2400). The findings are summarised in Table 9.

For scenarios with two waterways (W2) and 1200 tasks, the GNN model closely matched the OR Tools predictions, showing only a 1% optimality gap. This accuracy was maintained even with a higher Qlimit of 10. The GNN model's computation times were notably quick, averaging just above 1.8 seconds, far below the 15-minute threshold set for OR Tools.

In four waterway (W4) scenarios with 1200 tasks, the GNN model retained a 1% gap for Qlimit 1 but experienced a slight increase in the gap to 3% for Qlimit 10. The GNN model's computation times were consistent and slightly improved in the Qlimit 10 scenario.

For larger scenarios with 2400 tasks, the GNN model's predictions were highly accurate in two waterway configurations, showing a 0% gap for Qlimit 1. However, the optimality gap increased to 3% for Qlimit 10. Notably, the computation times were significantly lower in these larger scenarios, demonstrating the GNN model's efficiency.

The most challenging scenarios, with 2400 tasks and four waterways, saw a varied performance from the GNN model. It achieved perfect accuracy with a 0% gap for Qlimit 1, but the gap increased to 6% for Qlimit 10. The computation time for the GNN model was the highest in these cases, particularly for Qlimit 10, reflecting increased complexity.

Overall, the GNN model showcased its capability for fast and accurate predictions in various configurations of the WSSP. While not outperforming OR Tools, the GNN model's rapid prediction times and close results to OR Tools, even in complex cases, highlight its effectiveness and potential utility in operational settings. Note, however, that comparing actual boat paths from AIS data with predictions from a GNN model isn't straightforward. The GNN, in this context, is designed for scheduling optimisation within a port, not for predicting the exact maritime routes of boats. To match GNN predictions with AIS-tracked paths, the GNN would need to be part of a system trained explicitly on AIS data for path prediction, which is a different application from the scheduling focus of this study.

Table 9: Comparative analysis of makespan predictions for the Port of Duisburg using a GNN model and OR Tools across various scenarios, highlighting the model's performance with different waterway configurations and Qlimit constraints, along with the corresponding computation times. Gap is computed w.r.t. OR tools Prediction.

Problem							**Makespan**				
WSSP				**SM-RCPSP**			**OR TOOLS**	**GNN**		**Time**	
$\|V\|$	$\|V_{in}\|$	$\|V_{out}\|$	$\|W\|$	$\|R\|$	$\|J\|$	Qlimit	**Prediction**	**Prediction**	**Optimality Gap**	**OR TOOLS**	**GNN**
400	200	200	2	10	1200	1	602	606	1%	900	1.85
400	200	200	2	10	1200	10	305	315	1%	900	1.83
400	200	200	4	20	1200	1	408	411	1%	900	1.81
400	200	200	4	20	1200	10	379	381	3%	900	2.2
800	400	400	2	10	2400	1	1202	1206	0%	900	7.31
800	400	400	2	10	2400	10	123	127	3%	900	7.47
800	400	400	4	20	2400	1	538	539	0%	900	7.51
800	400	400	4	20	2400	10	72	76	6%	900	14.1

Next, we analyse the results regarding the computation time of the OR Tools more, as presented in Table 10. The comparative analysis of the GNN model and OR Tools reveals some intriguing insights. Optimal values found by OR Tools are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools.

A key observation is that the GNN model, in several cases, does not outperform OR Tools, particularly where optimal makespan values by OR Tools were not found. This is evident in scenarios like w4v200Q1, w4v200Q10, w2v400Q10 and w4v400Q10, where OR Tools delivers better predictions than the GNN model independent of the time limit. The notable exception is the w2v200Q10 instance, where the GNN model's prediction of 315 surpasses OR Tools' prediction of 331 and 327 made by OR Tools. However, the OR tool's time limit is smaller than the computation time for GNN (0.5 vs 1.83 seconds). This unique case suggests that the GNN model has the potential to yield more accurate predictions than OR tools.

However, it's crucial to consider that the instances in the study might be too simplistic for the GNN model, and the heuristics employed by OR Tools may be more effective in these particular scenarios. The fact that OR Tools does not consistently produce optimal results yet often outperforms the GNN model suggests a limitation in the complexity of the tasks being analysed. It indicates that these instances might not fully challenge or leverage the predictive capabilities of the GNN model. Therefore, we extensively analyse extra data instances where additional data instances are generated with more ships, and precedence relations are added, as explained in the data used section.

Table 10: This table details the computation times for the Port of Duisburg case study, comparing the performance of the GNN model against OR Tools' predictions over increasing computation time limits. Optimal values are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools. A dash (-) indicates the optimal solution was already found with a smaller computation time. A 'X' indicated that the computation time limit was too short to find a solution.

Instance	**GNN**		**OR Tools prediction after computation time limit**								
	Prediction	**Computation time**	**0.2**	**0.5**	**2**	**5**	**30**	**120**	**300**	**600**	**900**
w2v200Q1	606	1.85	X	X	602	-	-	-	-	-	-
w2v200Q10	315	1.83	**331**	**327**	305	-	-	-	-	-	-
w4v200Q1	411	1.81	-	-	408	408	408	408	408	408	408
w4v200Q10	381	2.2	380	380	379	379	379	379	379	379	379
w2v400Q1	1206	7.31	X	X	1202	-	-	-	-	-	-
w2v400Q10	127	7.47	126	125	125	123	123	123	123	123	123
w4v400Q1	539	7.51	X	X	538	-	-	-	-	-	-
w4v400Q10	76	14.1	72	72	72	72	72	72	72	72	72

For the Port of Duisburg case study analysis with more precedence relations, table 11 illustrates the outcomes of this comparative analysis.

When examining instances with a Qlimit of 10, which inherently presents a more complex and congested scheduling environment, the GNN model consistently delivers predictions that are on par with or surpass OR Tools' results with the same computation time limit. Notably, the GNN model achieves this with significantly lower computation times. For instance, with w2v200Q10, the GNN model's prediction of 312 closely approaches OR Tools' 305, with the former needing only 1.39 seconds—a stark contrast to the 15-minute cap typically given to OR Tools. To match the same quality makespan takes a minimum of 21.6 times longer for OR tools compared to the GNN model.

This pattern of time efficiency is even more pronounced with the w4v200Q10 instance, where the GNN model's prediction of 387 outperforms OR Tools' later predictions within a constrained computation time of 1.41 seconds. These outcomes highlight the GNN model's capability to swiftly navigate complex scheduling problems, offering a strategic advantage in time-sensitive operational settings.

For the less stringent Qlimit of 1, as seen in the w2v200Q1 and w4v200Q1 instances, the GNN model again demonstrates comparable predictive performance to OR Tools but with the added benefit of reduced computational demand. The GNN's prediction of 606 versus OR Tools' 605 and 412 versus 409 underscores its proficiency in generating timely solutions without compromising on result quality. So, the GNN model generates 192 and 86 times faster for the same quality makespan, respectively.

The findings from this study underscore the GNN model's strength in handling complex instances with difficult constraint limits and intricate precedence relations. When faced with such challenging scenarios, the GNN model competes well with OR Tools and frequently provides better solutions in terms of time efficiency. The GNN model demonstrated an exceptional average relative time performance, outperforming OR Tools by a factor of 96.3. This remarkable efficiency gain emphasises the GNN model's substantial advantages in operational contexts where precision and swift decision-making are essential.

Collectively, these results suggest that the GNN model is not only competitive but also occasionally superior to OR Tools, especially in instances where increased precedence relations and ship numbers heighten the problem complexity. It further implies that in cases where OR Tools cannot find the optimal solution promptly, the GNN model emerges as a more effective tool, offering better predictions in a shorter timeframe. These findings advocate for the GNN model's application in operational settings, where expedited and accurate scheduling is critical.

Table 11: A comparative analysis of the GNN model's predictions against OR Tools for the Port of Duisburg case study, focusing on computation times and the accuracy of predictions in scenarios with complex precedence relations and different Qlimit constraints. Optimal values are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools. A dash (-) indicates the optimal solution was already found with a smaller computation time.

Instance	**GNN**		**OR Tools prediction after computation time in sec**								
	Prediction	**Computation time**	**0.2**	**0.5**	**2**	**5**	**30**	**120**	**300**	**600**	**900**
w2v200Q1	606	1.38 sec	742	717	714	710	687	**634**	605	-	-
w4v200Q1	412	1.39 sec	**454**	**454**	**454**	**448**	**415**	411	410	409	409
w2v200Q10	312	1.39 sec	**327**	**320**	**317**	**316**	310	307	306	305	305
w4v200Q10	387	1.41 sec	**403**	**402**	**399**	**396**	**389**	382	380	379	378

7 Conclusion

This study explores the translation of the Waterway Ship Scheduling Problem (WSSP) into the Resource-Constrained Project Scheduling Problem (RCPSP) framework. Using the Graph Neural Network (GNN) model, we adapted WSSP variables for the single-mode RCPSP, introducing elements like vessel time windows and defining precedence relations, as outlined in the key studies by Lalla-Ruiz et al. (2018) and Hill et al. (2019). This approach integrates existing models to tackle the WSSP challenge effectively.

Our research encompassed a range of scenarios with different numbers of waterways and ships, using both randomly generated and non-randomly generated data, including a case study focused on the Port of Duisburg. The GNN model consistently achieved makespan values close to optimal solutions, demonstrating its proficiency in addressing the WSSP, particularly in complex cases. The optimality gaps were generally small, indicating the model's precision.

In the Duisburg case study, the GNN model's predictions were close to OR Tools benchmarks, especially in scenarios with fewer waterways and tasks. However, its predictions deviated more, though its rapid computation times were a consistent strength. Furthermore, it is recommended to examine if scenarios involving a larger number of vessels are being designed to be simpler, which can impact optimisation scalability and efficiency in Port of Duisburg's real world.

Our expanded analysis with larger data sets and varying Qlimit conditions revealed the GNN model's predictive speed—up to 96 times faster than OR Tools. While the GNN model didn't always excel in optimality compared to OR Tools, it remained competitive and demonstrated robustness across various maritime scheduling challenges.

The study confirms the GNN model's potential as a tool for maritime scheduling, particularly for swift decision-making, despite larger optimality gaps in certain scenarios. This suggests a need for further model optimisation, particularly for handling complexities with larger numbers of tasks and waterways.

Future research opportunities include developing neural architectures for logistical challenges in inland waterway transport (IWT), optimising WSSP translations into multi-mode RCPSP scenarios, and incorporating stochastic elements to account for maritime logistics unpredictability. Exploring varying operational conditions and additional case studies could refine the model's real-world applicability. Integrating real-time data into larger data sets can further enhance GNN capabilities, revolutionising the IWT industry for efficiency, adaptability, and resilience.

References

Adamu, P. I. and Aromolaran, O. (2018). Machine learning priority rule (mlpr) for solving resource-constrained project scheduling problems.

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., and Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112:3–41.

Cai, H., Bian, Y., and Liu, L. (2022). Learning to schedule srcpsp with uncertain resource capacity based on graph neural network and reinforcement learning. *Available at SSRN 4156352*.

de Vries, K. (2015). *Waardevol Transport: De toekomst van het goederenvervoer en de binnenvaart in Europa*. Bureau Voorlichting Binnenvaart.

Fan, S., Blanco-Davis, E., Yang, Z., Zhang, J., and Yan, X. (2020). Incorporation of human factors into maritime accident analysis using a data-driven bayesian network. *Reliability Engineering & System Safety*, 203:107070.

Hill, A., Lalla-Ruiz, E., Voß, S., and Goycoolea, M. (2019). A multi-mode resource-constrained project scheduling reformulation for the waterway ship scheduling problem. *Journal of scheduling*, 22(2):173–182.

Hofbauer, F. and Putz, L.-M. (2020). External costs in inland waterway transport: An analysis of external cost categories and calculation methods. *Sustainability*, 12(14):5874.

Hong-xing, Z., Bao-li, L., Chun-yuan, D., and Pan-pan, F. (2018). Ship scheduling optimization in one-way channel bulk harbor. *Operations Research and Management Science*, 27(12):28.

Jia, S., Li, C.-L., and Xu, Z. (2019). Managing navigation channel traffic and anchorage area utilization of a container port. *Transportation Science*, 53(3):728–745.

Kolisch, R. and Hartmann, S. (1999). *Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis*, pages 147–178. Springer US, Boston, MA.

Lalla-Ruiz, E., Shi, X., and Voß, S. (2018). The waterway ship scheduling problem. *Transportation Research Part D: Transport and Environment*, 60:191–209. Special Issue on Traffic Modeling for Low-Emission Transport.

Li, S. and Jia, S. (2019). The seaport traffic scheduling problem: Formulations and a column-row generation algorithm. *Transportation Research Part B: Methodological*, 128:158–184.

Meisel, F. and Fagerholt, K. (2019). Scheduling two-way ship traffic for the kiel canal: Model, extensions and a matheuristic. *Computers & Operations Research*, 106:119–132.

Notteboom, T., Yang, D., and Xu, H. (2020). Container barge network development in inland rivers: A comparison between the yangtze river and the rhine river. *Transportation Research Part A: Policy and Practice*, 132:587–605.

Notteboom, T. E. (2006). The time factor in liner shipping services. *Maritime Economics & Logistics*, 8:19–39.

of Infrastructure, R. M. and management, W. (2022). Vaarwegenoverzicht - informatie en data.

Praetorius, G., Lützhöft, M., and Bruno, K. (2010). The context matters: Maritime safety in the vessel traffic service (vts) domain.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. (2020). Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

Shou, Y. (2005). A neural network based heuristic for resource-constrained project scheduling. volume 3496, pages 794–799.

Sirimanne, S. N., Hoffman, J., Juan, W., Asariotis, R., Assaf, M., Ayala, G., Benamara, H., Chantrel, D., Hoffmann, J., Premti, A., et al. (2019). Review of maritime transport 2019. In *United Nations conference on trade and development, Geneva, Switzerland*.

Sprecher, A. and Kolisch, R. (1996). Psplib—a project scheduling problem library. *Eur. J. Oper. Res*, 96:205–216.

Tan, Z., Wang, Y., Meng, Q., and Liu, Z. (2018). Joint ship schedule design and sailing speed optimization for a single inland shipping service with uncertain dam transit time. *Transportation Science*, 52(6):1570–1588.

Teichteil-Königsbuch, F., Povéda, G., de Garibay Barba, G. G., Luchterhand, T., and Thiébaux, S. (2023). Fast and robust resource-constrained scheduling with graph neural networks.

Wang, C., Zhang, X., Chen, X., Li, R., and Li, G. (2017). Vessel traffic flow forecasting based on bp neural network and residual analysis. In *2017 4th International Conference on Information, Cybernetics and Computational Social Systems (ICCSS)*, pages 350–354. IEEE.

Zhang, X., Chen, X., Ji, M., and Yao, S. (2017). Vessel scheduling model of a one-way port channel. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 143(5):04017009.

Zhang, X., Lin, J., Guo, Z., and Liu, T. (2016). Vessel transportation scheduling optimization based on channel–berth coordination. *Ocean Engineering*, 112:145–152.

Zhao, X., Song, W., Li, Q., Shi, H., Kang, Z., and Zhang, C. (2022). A deep reinforcement learning approach for resource-constrained project scheduling. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1226–1234.

Table 39: Computation time of data results for W2,W4 and W6 with Qlimit = 1. Optimal values are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools. A dash (-) indicates that the computation time was too short to find a solution or optimal solution was already found.

Instance	**GNN**		**OR Tools prediction after computation time in sec**								
	Prediction	**Computation time**	**0.2**	**0.5**	**2**	**5**	**30**	**120**	**300**	**600**	**900**
w2v50Q1	165	0.10	161	-	-	-	-	-	-	-	-
w4v50Q1	121	0.10	118	118	118	118	118	118	118	118	118
w2v75Q1	230	0.21	226	-	-	-	-	-	-	-	-
w4v75Q1	158	0.21	155	155	155	155	155	155	155	155	155
w2v100Q1	306	0.36	302	-	-	-	-	-	-	-	-
w4v100Q1	208	0.60	205	205	205	205	205	205	205	205	205
w2v125Q1	389	0.55	385	-	-	-	-	-	-	-	-
w4v125Q1	258	0.57	255	255	255	255	255	255	255	255	255
w2v150Q1	465	0.80	461	-	-	-	-	-	-	-	-
w4v150Q1	324	0.80	321	321	321	321	321	321	321	321	321
w2v175Q1	530	0.97	526	-	-	-	-	-	-	-	-
w4v175Q1	361	1.07	358	358	358	358	358	358	358	358	358
w6v50Q1	97	0.01	94	94	94	94	94	94	94	94	94
w6v75Q1	123	0.21	120	120	120	120	120	120	120	120	120
w6v100Q1	169	0.35	166	166	166	166	166	166	166	166	166
w6v125Q1	189	5.6	186	186	186	186	186	186	186	186	186
w6v150Q1	258	0.81	252	252	252	252	252	252	252	252	252
w6v175Q1	271	1.09	265	265	265	265	265	265	265	265	265

Table 40: Computation time of data results for LARGE W2 and W4, Qlimit = 10. Optimal values are marked with an underline, and predictions in bold denote instances where the GNN model outperforms OR Tools. A dash (-) indicates that the computation time was too short to find a solution or optimal solution was already found.

Instance	**GNN**		**OR Tools prediction after computation time in sec**								
	Prediction	**Computation time**	**0.2**	**0.5**	**2**	**5**	**30**	**120**	**300**	**600**	**900**
w2v50Q10	89	0.1	82	82	82	81	80	80	81	81	81
w4v50Q10	116	0.1	112	112	111	111	111	111	110	110	110
w2v75Q10	122	0.21	117	115	115	115	115	114	114	114	114
w4v75Q10	151	0.22	145	145	145	144	144	144	144	144	144
w2v100Q10	163	0.37	156	155	154	154	151	151	151	151	151
w4v100Q10	193	5.52	191	191	191	191	191	189	189	189	189
w2v125Q10	199	0.57	199	195	195	193	193	191	191	191	191
w4v125Q10	242	0.57	236	236	236	236	236	236	236	236	236
w2v150Q10	240	0.84	247	241	**237**	237	233	231	231	231	231
w4v150Q10	301	0.84	302	302	302	302	301	**299**	299	299	299
w2v175Q10	277	1.13	284	279	**274**	271	268	265	265	265	265
w4v175Q10	334	1.13	332	332	332	332	332	332	332	332	332

Table 37: Comparison of Optimum and GNN Predicted Values for RCPSP Instances

RCPSP Instance	Optimum	GNN Predicted	Optimality Gap
j301_w2v5.sm	32	37	16%
j302_w2v5.sm	45	49	9%
j303_w2v5.sm	36	43	19%
j304_w2v5.sm	60	71	18%
j305_w2v5.sm	62	72	16%
j301_w4v5.sm	22	22	0%
j302_w4v5.sm	31	34	10%
j303_w4v5.sm	27	30	11%
j304_w4v5.sm	24	27	13%
j305_w4v5.sm	19	19	0%
j601_w2v10.sm	54	65	20%
j602_w2v10.sm	53	63	19%
j603_w2v10.sm	76	85	12%
j604_w2v10.sm	66	68	3%
j605_w2v10.sm	77	87	13%
j601_w4v10.sm	35	35	0%
j602_w4v10.sm	51	51	0%
j603_w4v10.sm	41	41	0%
j604_w4v10.sm	28	28	0%
j605_w4v10.sm	48	48	0%
j901_w2v15.sm	85	90	6%
j902_w2v15.sm	99	107	8%
j903_w2v15.sm	95	102	7%
j904_w2v15.sm	108	116	7%
j905_w2v15.sm	97	104	7%
j901_w4v15.sm	77	77	0%
j902_w4v15.sm	70	70	0%
j903_w4v15.sm	56	56	0%
j904_w4v15.sm	66	66	0%
j905_w4v15.sm	52	52	0%
j1201_w2v20.sm	117	121	3%
j1202_w2v20.sm	136	137	1%
j1203_w2v20.sm	128	132	3%
j1204_w2v20.sm	118	125	6%
j1205_w2v20.sm	131	136	4%
j1201_w4v20.sm	63	70	11%
j1202_w4v20.sm	70	80	14%
j1203_w4v20.sm	63	70	11%
j1204_w4v20.sm	55	62	13%
j1205_w4v20.sm	64	79	23%
j1501_w2v25.sm	164	167	2%
j1502_w2v25.sm	165	169	2%
j1503_w2v25.sm	175	178	2%
j1504_w2v25.sm	158	165	4%
j1505_w2v25.sm	161	167	4%
j1501_w4v25.sm	111	116	5%
j1502_w4v25.sm	82	87	6%
j1503_w4v25.sm	101	108	7%
j1504_w4v25.sm	76	82	8%
j1505_w4v25.sm	117	130	11%
j1801_w2v30.sm	202	212	5%
j1802_w2v30.sm	218	225	3%
j1803_w2v30.sm	217	224	3%
j1804_w2v30.sm	221	228	3%
j1805_w2v30.sm	177	187	6%
j1801_w4v30.sm	113	117	4%
j1802_w4v30.sm	90	101	12%
j1803_w4v30.sm	116	119	3%
j1804_w4v30.sm	95	106	12%
j1805_w4v30.sm	132	138	5%

Table 38: Optimality Gap Analysis for RCPSP Instances

RCPSP Instance	Optimum	GNN Predicted	Optimality Gap
j301_w2v5.sm	15	15	0%
j302_w2v5.sm	22	25	14%
j303_w2v5.sm	18	19	6%
j304_w2v5.sm	21	23	10%
j305_w2v5.sm	19	21	11%
j301_w4v5.sm	14	14	0%
j302_w4v5.sm	21	24	14%
j303_w4v5.sm	18	21	17%
j304_w4v5.sm	16	18	13%
j305_w4v5.sm	15	16	7%
j601_w2v10.sm	29	30	3%
j602_w2v10.sm	36	43	19%
j603_w2v10.sm	35	38	9%
j604_w2v10.sm	33	37	12%
j605_w2v10.sm	28	29	4%
j601_w4v10.sm	32	35	9%
j602_w4v10.sm	29	34	17%
j603_w4v10.sm	23	23	0%
j604_w4v10.sm	30	34	13%
j605_w4v10.sm	23	23	0%
j901_w2v15.sm	51	56	10%
j902_w2v15.sm	37	41	11%
j903_w2v15.sm	35	37	6%
j904_w2v15.sm	35	37	6%
j905_w2v15.sm	35	40	14%
j901_w4v15.sm	34	37	9%
j902_w4v15.sm	37	42	14%
j903_w4v15.sm	32	33	3%
j904_w4v15.sm	41	45	10%
j905_w4v15.sm	30	32	7%
j1201_w2v20.sm	69	82	19%
j1202_w2v20.sm	60	71	18%
j1203_w2v20.sm	52	63	21%
j1204_w2v20.sm	66	79	20%
j1205_w2v20.sm	66	81	23%
j1201_w4v20.sm	50	53	6%
j1202_w4v20.sm	44	48	9%
j1203_w4v20.sm	43	44	2%
j1204_w4v20.sm	43	46	7%
j1205_w4v20.sm	36	43	19%