An imitation learning-based routing of surface ships

Pujan Pokhrel, Elias Ioup, and Mahdi Abdelguerfi

***Abstract*—This paper introduces a Threshold Relaxation algorithm to iteratively reduce the threshold of machine learning algorithms in reinforcement learning algorithms trained in a supervised manner to solve shortest-path vessel routing problems. Since we utilize Graph Neural Networks to model the data, our method generalizes to different graph sizes. Our experiments are performed on real-world data using the predicted fields from Climate Forecast System Reanalysis (CFSR) hindcasts.**

First, we calculate the edge weights using the resistance estimation from wave heights. The node constraints are considered by only taking the safe nodes while generating the problem domain. We then use various features related to the edge weights, depth, and neighbors to train a machine-learning algorithm using an imitation learning procedure. The machine learning framework used in this study is built using the Diffusion Convolution Layer (DCL) and the Graph Attention Layer (GAL). Next, we introduce a Threshold Relaxation (TR) scheme to relax the classifier while solving the learning-to-rank problem, which is especially applicable in the case of shortest-path problems. The resulting model optimizes the fuel-efficient, safe route within a 8.4% Optimality Gap while only exploring an average of 65.2% of the total nodes.

***Index Terms*—Imitation Learning, Ship Routing, Shortest Path Problem, Threshold Relaxation.**

I. Introduction

In this paper, we study the prediction of a fuel-efficient path for marine surface vessels sailing from one location to another under a dynamic surface ocean gravity-wave field that constrains the vessel's motion. Pokhrel *et al.* have previously explored the improved wave forecasts using Climate Forecast System Reanalysis (CFSR) hindcasts in the context of machine learning data assimilation [1]. The geographical properties (land/islands in oceans) give rise to constraints that stay the same for all planning periods, while the wave effects are dynamic and change depending on the weather.

Several approaches have been previously investigated for path planning for marine voyages. The approaches can be divided into four types: a) optimal control problem formulated on a graph using dynamic programming methods [2], b) heuristic search schemes like A^* algorithm [3] and Rapidly-exploring Random Tree (RRTs) [4–6], c) nonlinear convex optimization [7–10] or evolutionary algorithms [11] d) reinforcement learning approaches that use reward function for task completion and enforce penalties for unsafe state exploration [12, 13].

Pujan Pokhrel is with the Canizaro Livingston Gulf States Center for Environmental Informatics, University of New Orleans, LA 70148, USA (e-mail: ppokhrel@uno.edu).

Mahdi Abdelguerfi is with the Canizaro Livingston Gulf States Center for Environmental Informatics, University of New Orleans, LA 70148, USA (e-mail: mahdi@cs.uno.edu).

Elias Ioup is with the Center for Geospatial Sciences, Naval Research Laboratory, Stennis Space Center, Mississippi, USA (e-mail: elias.ioup@nrlssc.navy.mil).

There is an increasing trend towards applying deep learning approaches to automatically discovering heuristic algorithms that can solve vessel routing problems [12, 14–16]. This trend is motivated by the non-generalizability of heuristic algorithms which require substantial trial-and-error and human experts to improve performance in terms of solution quality. Statistical approaches allow the trained network to generalize on problems that share similar structures but differ only in data and follow an equal distribution. Moreover, the use of reinforcement learning (RL) approaches allows the model to discover underlying patterns from a problem class, which can be used to develop alternative algorithms that are better than the human-designed ones. Reinforcement learning algorithms, however, take a long time to train since the model needs to take action and then observe the rewards to improve performance [17, 18]. Imitation learning approaches have been proposed to reduce the training time of reinforcement learning algorithms. In these approaches, the agent learns to generalize the task using the samples generated from problems that follow similar distribution as the ones in the training set. Graph Neural Networks (GNN) have also been pioneered in the last couple of decades to take advantage of the graph structure of various datasets [19, 20]. The use of convolution, attention, and diffusion layers have enabled graph/node prediction and Spatio-temporal forecasting. These networks have shown adequate generalizability to graphs of different sizes when the problem distribution is the same.

We use encoder-decoder GNN architecture for combinatorial optimization in the setting of vessel routing. In this paper, we utilize an imitation learning framework for weather routing of surface ships. We introduce a Threshold Relaxation (TR) scheme that allows solving the problems close to the minimum while exploring fewer nodes. To validate our results, we use our framework to solve routing problems in realistic scenarios. The rest of the paper is organized as follows. Section II examines the related works in Branch and Bound (B&B) problems and Ship Routing. Sections III, IV, and V show the problem formulation, methodology, and results. Section VI discusses the results, and Section VII concludes the paper.

II. Related Work

A. Imitation learning on branch-and-bound framework

Imitation learning and supervised learning are two common approaches for learning to branch in branch-and-bound algorithms. Balcan [21] has shown that machine learning algorithms can be used to learn high-performing strategies to branch and bound for a given problem set. Khalil *et al.* [17] have previously used a supervised learning framework to learn branching in the context of Mixed Integer Linear Programming (MILP). They have used the data collected employing Strong

Branching (SB) rules to solve the ranking problem, which is competitive with the state-of-the-art solver. Hottung *et al.* [15, 22] have developed a method integrating deep neural networks (GNN) and heuristics tree search to develop a data-driven framework DLTS to solve search problems. Their algorithm can achieve high performance without any problem-specific information. Hottung *et al.* [22] have already shown that the DLTS framework finds smaller gaps in optimality on real-world datasets compared to the state-of-the-art metaheuristics proposed by Karapetyan *et al.* [23]. While their model does not require additional training data, their system performance relies on the quality of the provided solutions.

Various studies have explored methods to learn heuristics using imitation learning algorithms. Addanki *et al.* [14] and Song *et al.* [24] have used learning methods to improve neighborhood search to improve the performance of the solver. Their technique uses a machine learning algorithm to predict whether to select the modified variables or assign a new value to an already selected subset of variables. Their methods, however, are limited in that they require a feasible point at the start to find a solution. Nair *et al.* and [25], Ding *et al.* [26] have used tripartite graph representation to extract correlations, constraints, and objective functions without human intervention. The method by Nair *et al.* [25] proposes a problem posed as predicting variable assignments as a generative modeling problem. Their method can also generate partial assignments to the problem at test time.

Gasse *et al.* [19] have proposed a Graph Convolutional Network (GCN) based framework to learn branching rules using imitation learning on the data generated by the SB method. While competitive with the traditional algorithm, their framework outperforms the other machine learning-based approaches to the MILP problem. Gupta *et al.* [18] have proposed a hybrid branching model that uses GNN at the initial decision point and a faster predictor, like the multi-layer perceptron in the later steps to learn effective policies. Nair *et al.* [25] have used imitation learning to build a MILP branched solver, which uses GNN in a parallel setting to decrease computational time. They have also validated their model on five real-life datasets to show that the machine learning method trained on one problem can generalize to similar problems.

While there is literature on imitation learning on a branch-and-bound framework for MILP solvers, machine learning has not been explored for the ship routing problem.

B. Ship Routing

Dijkstra algorithm and differential equations are common procedures used for weather routing of surface vessels. Using the Dijkstra-based approach, Mannarini *et al.* [27, 28] have devised a vessel routing system that takes weather constraints into account for various ship types. The authors have also extensively explored wind propulsion-based vessels. While they consider the physical properties and the geometries, their algorithm provides sub-optimal paths to make it faster.

In the case of the differential path planning [29], for a ship moving from point A to point B in a strong and dynamic environment, the Hamilton-Jacobi (HJ) equation governs the exact reachability front with certain initial conditions and appropriate boundary conditions for coastlines. The zero level set contour of the solution of the HJ equation used in differential path planning at $t > 0$ is the reachability front for a vehicle starting from the point x_A at $t = 0$, and the first time t at which the zero level set contour reaches target B, which is the endpoint. We can then extract the exact time-optimal path $X^*(t)$ from the time series of the zero-level set contours by solving the particle backtracking ordinary differential equation (ODE).

Mannarini *et al.* [27] have conducted a numerical and theoretical study to show that at a sufficient enough discretization level, the solution of the graph-based approach converges to the differential equation-based approach.

Based on the results of the study from Mannarini *et al.* [27], we use Graph Neural Networks (GNN) as the machine learning method to capture various spatial dependencies. Moreover, to capture temporal dependencies, we use Gated Recurrent Layers [30] and Graph Attention Layers [20]. We formulate the routing problem as a Constrained Markov Decision Process (CMDP), considering geographical and weather-related constraints and removing the nodes that do not follow the constraints from the problem space. While various constraints related to the safe reachability of the vessel to the endpoint are used, they are general and only depend on the bulk wave parameters and a limited set of measurements of the ship.

III. Preliminaries and Problem Formulation

A. Constrained Markov Decision Process (CMDP)

We consider the problem of finding an optimal policy in a finite horizon Constrained Markov Decision Process (CMDP) subject to various constraints. A finite horizon CMDP is a tuple $(S, C, A, T, R, \gamma, s_0, n, \delta)$ where

- S refers to a set of states.
- $C \subseteq S$ is a set of *safe states* that satisfy the mission constraints.
- A is a set of actions.
- $T : S \times A \times S \to [0, 1]$ is a stochastic function indicating state transition.
- $R : S \times A \times S \to R$ is a reward function while moving from one state to another.
- $\gamma \in [0, 1]$ is a discount factor.
- $s_0 \in C$ is the initial state.
- n is the planning horizon.
- $\delta : R \to [0, 1]$ is a risk bounding function that gives the maximum acceptable probability of entering a failure state as a reward function.

If we denote a state at step t as s_t, the state history from time step t to t' can be represented as $h_{t:t'} = (s_t, a_t, s_{t+1}, a_{t+1}, ..., s_{t'})$. We denote the set of all possible state histories as H, and the set of all possible state histories with only safe states as H^C. Similarly, a policy $\pi : S \to P(A)$ can be defined as the mapping from the state space to the space of the probability distribution over the actions, with $\pi(a|s)$ denoting the probability of selecting action a at state s. We use two policies. i.e., branching and pruning policies (π_S and π_P) in this work.

The objective of the problem is to select a policy π_θ that maximizes the discounted cumulative reward $J_R^\pi = E_{r\ \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}]$ while satisfying the constraints $J_{C_i}^\pi[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t, s_{t+1}]$. Since the wave forecasts are supposed to perfectly represent the environment, only the nodes that satisfy the constraints are chosen for the initial search in problem space. Formally, we can represent the problem as

$$max_\theta J_R^{\pi_\theta} \tag{1}$$

subject to environmental constraints defined in the next subsection.

Table I
VARIOUS TERMS USED IN THE CONSTRAINTS

Symbol	Name	units	Value
λ	wavelength	m	-
L	length at waterline	m	16.2
Hs	significant wave height	m	-
T_R	ship natural roll period	s	9.8
α	ship heading direction relative to wave	rad	-
T_W	peak wave period	s	-
Fr	froude number	-	$\frac{0.52v}{\sqrt{g_0 L}}$

Description of various parameters used in this study. In the above table, m refers to meters, s refers to seconds, and rad refers to radians.

B. Environmental Constraints

We use various weather-specific constraints to account for the hazardous conditions encountered by the ship during the voyage, which are mainly caused by the ocean waves. The description of the various variables used in this subsection can be found in Table I. We use the following constraints in this paper:

$$0.8 \leq \lambda/L \leq 2 \tag{2}$$

$$0.8|T_E| \leq T_R \leq 2.1|T_E| \tag{3}$$

$$1/40 \leq H_s/L \leq 1/25 \tag{4}$$

$$|\pi - \alpha| \leq \pi/4 \tag{5}$$

$$1.3T_W \leq vcos(\pi - \alpha) \leq 2.0T_W \tag{6}$$

$$Fr.cos(\pi - \alpha) \geq Fr_{crit} \tag{7}$$

where the critical Froude number is given by

$$Fr_{crit} = 0.2324(H_s/\lambda)^{-1/3} - 0.0764(H_s/\lambda)^{-1/2} \tag{8}$$

We refer the readers to Mannarini *et al.* for the detailed description on the calculation of the variables.

C. Calculation of Fuel Cost

Since the function that calculates the ship fuel cost is non-convex, it is generally expressed in terms of resistance encountered. The parameters used in this subsection are described in Table II. The total resistance encountered is the sum of the resistances generated due to various environmental factors like winds and waves. Since the waves contribute the most to the resistance encountered by the ship, we decompose the resulting force that hinders the ship's motion into calm water resistance R_c and the wave-making resistance R_{aw},

$$R_T = R_c + R_{aw} \tag{9}$$

Table II
SHIP PARAMETERS

Symbol	Name	units	Value
P_{max}	maximum engine break power	hp	4000
c	top speed	kt	16.2
L	length at waterline	m	69
B	beam (width at waterline)	m	14
T	draught	m	3.4
T_R	ship natural roll period	s	9.8
GM	metacentric height	m	2.3
Δ	displacement	t	550

Description of various parameters used in this study. In the above table, m refers to meters, s refers to seconds, t refers to tonnes, hp refers to horsepower, and kt refers to knots.

The calm water resistance is generally calculated in terms of dimensionless drag coefficient C_T defined by the equation

$$R_c(v) = C_T \frac{1}{2} \rho S v^2 \tag{10}$$

where ρ refers to the sea water density and S denotes the ship's wet surface area. The calculation of C_T is done as outlined in International Towing Tank Conference (ITTC) 2011 [31, 32], following the method of Holtrop [33].

Following the literature on ship resistance, a non-dimensional resistance σ_{aw} is introduced

$$R_{aw} = \sigma_{aw}(L, B, T, Fr) \frac{\rho g_0 \zeta^2 B^2}{L} \varphi(\frac{L}{\lambda},\ \alpha) \tag{11}$$

where α is the angle between the wave direction and vessel direction of advance, ζ is the wave amplitude calculated as $2\zeta = H_s$.

Mannarini *et al.* derive the following equation for the calculation of normalized non-dimensional reistance

$$\bar{\sigma}_{aw} = 20(B/L)^{-1.20}(T/L)^{0.62} \tag{12}$$

where $\sigma_{aw} = \bar{\sigma}_{aw} Fr/\bar{F}r.$ and $1/\bar{F}r$ is calculated as $\bar{F}r = 206.5867(1/2)^{0.6376} B^{-1.2121}\, T^{0.6247} (1/4)^{1.3611} \bar{F}r$

Afterward, the sustained velocity v is calculated using the formula

$$k_3 v^3 + k_2 v^2 - P = 0 \tag{13}$$

where

$$k_3 = \frac{P_{max}}{c^3} \tag{14}$$

$$k_2 = \bar{\sigma}_{aw} \frac{1}{\eta \bar{F}r} \varphi_0 \rho \zeta^2 B^2 \sqrt{g_0/L^3} \tag{15}$$

where g is the gravitational constant with the value of 9.81. The python library sympy is used to optimize the equation to find the sustained speed v.

The modified R_c and R_{aw} are then calculated according to the modified formula presented by Mannarini *et al.* [28], i.e.,

$$R_c = \eta k_3 v^3 \tag{16}$$

$$R_{aw} = \eta k_2 v \tag{17}$$

We refer the readers to Mannarini *et al.* for the calculation of φ and σ_{aw}.

IV. Methodology

This section discusses the methodology and the description of the algorithm used in this paper. We first explain the branch-and-bound algorithm. Then we illustrate the imitation learning framework, the policy network, training algorithm, threshold relaxation procedure, and the dataset used in this study.

A. Branch-and-bound algorithm

Branch-and-bound (B&B) algorithms refer to a set of algorithms using a divide and conquer strategy to solve optimization problems.

In this framework, when we optimize the function f over a feasible set F, we divide the problem recursively into its subsets $F_1, F_2, ..., F_p$ such that $\cup_{i=1}^{p} F_i = F$. A recursion tree is an enumeration of all feasible solutions in which nodes are sub-problems, and the edges are partition conditions [21]. The B&B algorithm performs convex relaxation of each sub-problem to find a lower bound $l_{lb}(F_i)$ or the upper bound $l_{ub}(F_i)$ for the node and its descendants.

The B&B algorithm maintains a queue L of the active nodes. The first node in the queue is the root. We consider a node to be fathomed (i.e., no further exploration in the subtree) if a) the lower bound $l_{lb}(F_i)$ is larger than the current global upper bound, i.e., no solution in the subtree is better than the current solution, b) $l_{lb}(F_i) = l_{ub}(F_i)$, i.e., B&B has found the best solution in the current subtree, or c) The problem is infeasible.

If the node is not fathomed, it is branched into children of F_i and pushed to the queue L. The algorithm terminates when the gap between the global upper and lower bound achieves a certain tolerance level or the queue L is empty. The details of the algorithm are also provided in the previous literature [17, 18, 21]. The algorithm for B&B is displayed in Figure 1.

B. Learning Policy for Branch and bound algorithm

The branch-and-bound algorithm has two goals: to find the optimal solution and to prove its optimality. While the algorithm can be run till all the nodes are visited to find the global optimum, it is not always feasible to return the optimal solution in a limited time since the algorithm needs to search and prove that the rest of the solutions are worse. The machine learning-driven framework with B&B algorithms seeks to provide a reasonable solution within a limited time without rigorously proving the optimality. This quick-guess procedure makes the search process faster since the algorithm aggressively prunes the unpromising portions of the search.

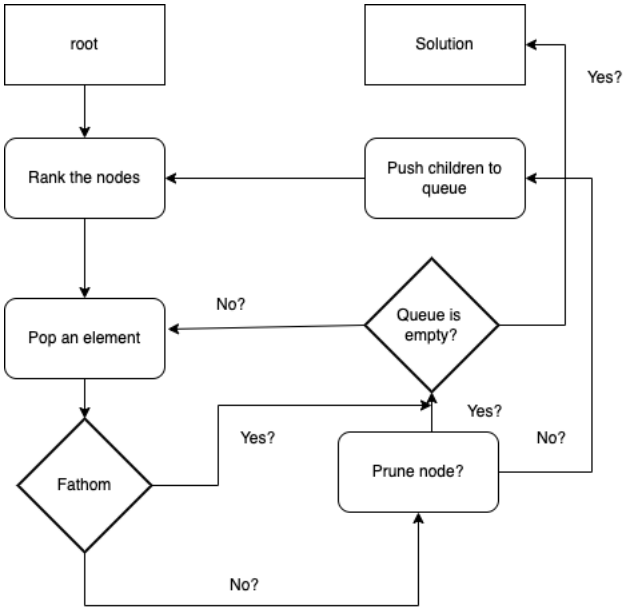

Figure 1. The branch-and-bound algorithm

The above figure displays the branch-and-bound algorithm used in this paper. Similar figure can also be found in [21].

1) Oracle: To generate the Oracle for imitation learning which demonstrates the desired behavior, we run the shortest-path selection algorithm with B&B to create the training dataset. This list of actions by Oracle seeks to demonstrate the ideal behavior of the machine learning algorithm. Since the perfect Oracle requires enormous computational power to obtain the optimal sequence of expanded nodes, we design the Oracle without proof of optimality. This procedure, as suggested by Balcan *et al.* [21] requires two Oracles, i.e., node selection Oracle π_S^* which constantly expands the node which has a feasible set containing the optimal solution and node pruning Oracle π_P^* which prunes the other nodes.

Our implementation of the machine learning algorithm for the B&B algorithm focuses on selecting the nodes to expand (explore the children) and prune (remove the node and its children). Using the behavior cloning algorithm, we can translate the problem of choosing the two classes into a binary classification. We can represent the classes as $prune$ and $expand$, respectively.

2) Imitation Learning: The imitation learning framework for branch-and-bound algorithms formulates the learning problem as a sequential decision-making process. The trajectory consists of the sequence of states $s_1, s_2, ..., s_T$ and actions $a_1, a_2, ..., a_T$. The policy $\pi_\theta \in \Pi$ maps the state S into action $\pi(s_t) = a_t$. The state in the branch-and-bound setting is the tree of nodes visited so far with the lower/upper bounds. The node selection policy π_S has the state space s_t and has the action space a_t which is selected from a queue of active nodes. The nodes selected are then expanded for the next iteration. The node pruning policy π_P predicts the class in $prune$, $expand$ given state s_t and the most recently selected node. The imitation problem can be reduced to supervised

learning where the policy takes the feature-vector description of the state s_t and predicts the next Oracle action a_t^*. The machine learning algorithm learns the expected best action using a simple behavior cloning procedure [34].

C. Policy Network

To learn the policy π_θ in Equation 1, we parameterize it as a neural network π_θ where θ refers to the trainable parameters.

1) Diffusion convolution layer : We use Gated Recurrent diffusion convolution layers in our study [30]. Let the parameter tensor $\Theta \in R^{Q\times P\times K\times 2} = [\theta]_{q,p}$ where $\Theta_{q,p,:,:} \in R^{K\times 2}$ parameterizes the convolutional filter for the pth input and the qth output, P refers to the dimension of features, Q refers to the dimension of outputs, and K refers to the dimension of the convolution filter. We can now define the diffusion convolutional layer as:

$$H_{:,q} = a(\sum_{p=1}^{p} X_{:,p\star G} f_\theta q, p, , :) \; for \; q \; \in \; 1, ..., Q \tag{18}$$

where $X \in R^{N\times P}$ is the input, $H \in R^{N\times Q}$ is the output, $f_{\theta\, q,p,,:}$ are the filters, $\star G$ is the diffusion operator and a is the activation function.

In order to model the temporal dependency, we utilize Gated Recurrent Units (GRU) which model temporal relationships from the data. Similar to Li *et al.* [30], we replace matrix multiplications with the diffusion convolution, i.e.,

$$r^t = \sigma(\Theta_r \star G[X^t, H^{t-1}] + b_r) \tag{19}$$

$$u_t = \sigma(\Theta_u \star G[X^t, H^{t-1}] + b_u) \tag{20}$$

$$C^t = tanh(\Theta_C \star G[X^t, (r^t \dot{H}^{t-1})] + b_c) \tag{21}$$

$$H^t = u^t \dot{H}^{t-1} + (1 - u^t)\dot{C}^t \tag{22}$$

where X^t, H^t, r^t, u^t refer to the inputs, outputs, reset gate, and update gate at time t, respectively. Similarly, $\star G$ refers to the diffusion convolution and Θ_r, Θ_u, and Θ_C refer to the parameters for the corresponding filters.

2) Graph attention layer : Graph attention layer refers to the masked self attention applied on a graph structure [20]. We utilize two types of graph attention layers in this work [35].

1) Type-level attention: For a node v, the hidden representation at the lth layer is h_v^l. The hidden representation of the type t of nodes can be defined as $h_\eta^l = \sum_{v'} \bar{A}_{vv'} h_v^{'l}$ and can be calculated as the sum of the neighbouring node of type t where v and $v^{'}$ refer to the selected node and its neighbouring node, respectively. The type-level attention scores of the same layer can now be defined as

$$a_\eta = \sigma(\mu_\eta^T [h_v^l || h_\eta^l]) \tag{23}$$

where μ_t is the attention vector for type η, $||$ denotes concatenation and $\sigma(.)$ refers to the activation function. The attention weights are then normalized using the softmax function, i.e.,

$$a_\eta = \frac{\exp(a_\eta)}{\sum_{\tau' \in T} \exp(a_{\eta'})} \tag{24}$$

where τ is the set of nodes used for type-level attention calculation.

2) Node-level attention: For a node v with type η and neighbouring node $v^{'} \in N_v$ with a type $\eta^{'}$, the node-level attention score at the lth layer can be defined as

$$b_{vv^t} = \sigma(\phi^T \dot{\alpha}_{\eta'} [h_v^l || h_{v'}^l]) \tag{25}$$

where ϕ is the attention vector and h_v^l is the hidden representation at node v. Similarly, $v^{'} \in N_v$ with type $\tau^{'}$ refer to the neighbouring node of v with type τ The attention scores are then normalized using the softmax function, i.e.,

$$\beta_{vv'1} = \frac{\exp(b_{vv'})}{\sum_{i\in N_v} \exp(b_{vv'})} \tag{26}$$

where N_v is the set of nodes used to calculate node-level attention.

Softmax layer: The softmax layer transforms the node embeddings into outputs within the range (-1, 1) using a $softmax$ function.

The architecture of the proposed network is displayed in Figure 2. We use one encoder and one decoder blocks which contain the Diffusion Layer and the Attention Layer each. The corresponding layers in the encoder supply the outputs to the respective layers in the decoder. The networks used in this study use 500 hidden networks each. Dropouts are set after each network with a value of 0.15. In the case of the GAT network, 16 attention heads are used, and the hidden nodes are set to 500. A greater number of hidden nodes or attention heads are not used since very small improvements were seen, while testing up to 2048 nodes and 24 attention heads while the computational complexity was significantly higher. Afterward, we use MaxPooling to collect the outputs and a softmax function to generate predictions.

D. Training algorithm

To train the policy networks as a supervised learning problem, we use DAgger [36]. DAgger is an iterative imitation learning algorithm that repeatedly trains the policy to make decisions that agree better with Oracle's decisions in the situations encountered while running past versions of the policy. This training procedure makes the trained algorithm more likely to deal with situations arising during test time. While using the trained policy on the test dataset, the optimal node pushed into the queue has a higher ranking than other nodes in the queue, while the non-optimal one has a lower ranking than the rest in the queue. The non-optimal node is then pruned with the node-pruning policy. The DAgger algorithm is displayed in Algorithm 1.

E. Threshold Relaxation

Algorithm 2 shows the threshold relaxation procedure used in this study. First, we train the algorithm using an imitation learning algorithm. While the thresholds can account for the false positives, using the static threshold means that the threshold will have to be set to a small number so that no positive samples are missed. Moreover, the algorithm can miss some nodes essential to find a solution, thus being unable

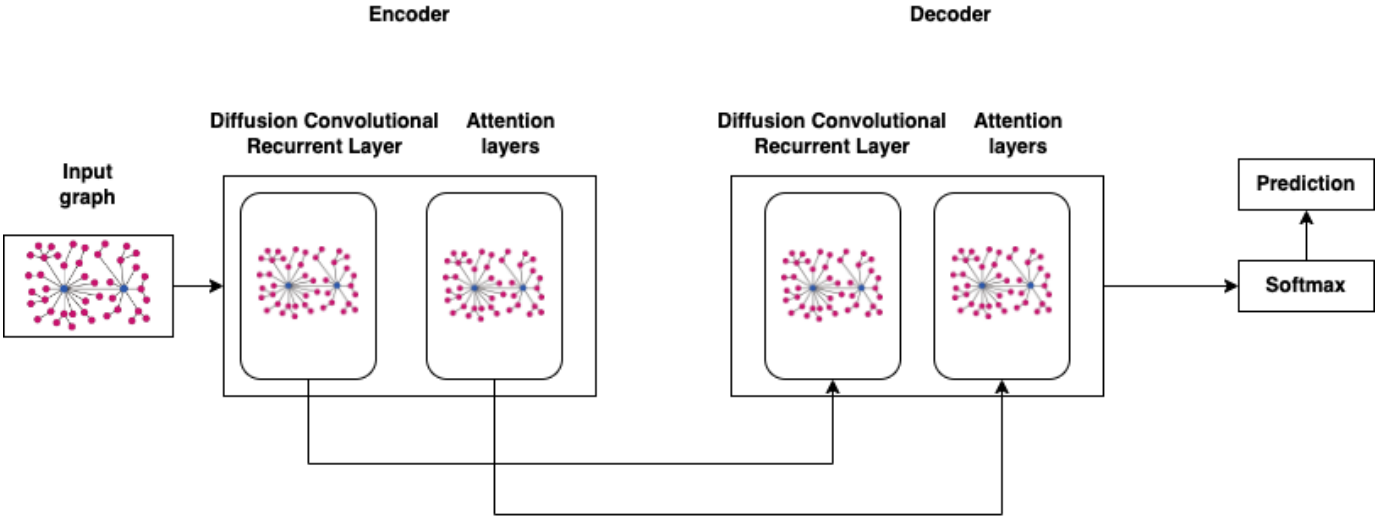

Figure 2. Encoder-Decoder architecture used in this study. The outputs are concatenated at the end and passed to the softmax function to make predictions.

Algorithm 1 Policy Learning (π_S^*, π_P^*)

function DAgger(a, b) ▷ The DAgger algorithm
  for $k \in 1, ..., N$ **do**
    for q in the problem set Q **do**
      $D_S^{(Q)}, D_P^{(Q)} \rightarrow CollectExample(q, \pi_P^{(k)}, \pi_S^{(k)})$
      $D_S \rightarrow D_S \cup D_S^{(1)}$
      $D_P \rightarrow D_P \cup D_P^{q}$
    end for
    $\pi_S^{k+1}, \pi_P^{(k+1} \rightarrow$ train classifiers using D_S and D_P
  end for
  return Best policies π_S^k, π_P^k
end function

Algorithm 2 Threshold Relaxation (TR)

function TR(λ,threshold) ▷ The Threshold Relaxation algorithm
  $\lambda = 0.8$
  $threshold = 0.2$
  while end state not found **do**
    threshold = threshold * $\lambda \rightarrow$ Relax the threshold 0.8 $\times$ original
  end while
  return Best policies π_S^k, π_P^k
end function

to solve the problem. Therefore, we use an iterative scheme in which the termination criteria is that the algorithm must find the endpoint. For an individual problem, we start with a threshold generated based on the number of positive and negative samples (0.2 in this case). We then relax the classifier threshold iteratively till the problem is solved. The Threshold Relaxation (TR) procedure only appends the new nodes to the queue based on the threshold and, thus, does not require generating the problem from scratch.

F. Dataset and Features

The weather dataset used in this study is derived from Ifremer hindcasts. The dataset has the spatial resolution of 720×361 where the former refers to longitudes and the latter refers to latitudes. The study domain of this algorithm is the Atlantic Ocean. First, the land nodes are removed using the land-sea mask in the hindcast files. We then retain the safe nodes using procedures described in Section III(B). We select two random points between the nodes from the geographical coordinates to generate the dataset. The shortest path is computed between the two points using the heap-based branch-and-bound implementation as an Oracle. A total of 1372 samples are developed and divided into train:validation: test sets in the ratio (0.7:0.2:0.1). We select random dates within the date range January 1-30, 2015, to generate the problem domain.

The features used in this study include the significant wave height of the node, current index, indices of the root node and the end node, and current cost and depth. We then normalize the features within the range [-1, 1] using BatchNormalizations. The input graph is generated by taking the snapshot of the resistance for the problem domain. Since the start point and end points are selected at random, the size of the input graph also varies according to the problem. An example dataset used in this paper is displayed in Figure 3.

V. Results

This section describes the results of the methods tested in this study. We first compare the machine learning algorithm to the training data, then compare the individual machine learning layers of the algorithm for ablation study. Afterward, we present the test data results with and without the Threshold Relaxation. Finally, we present the result on an individual problem to study the performance of the proposed algorithm compared to the optimum path.

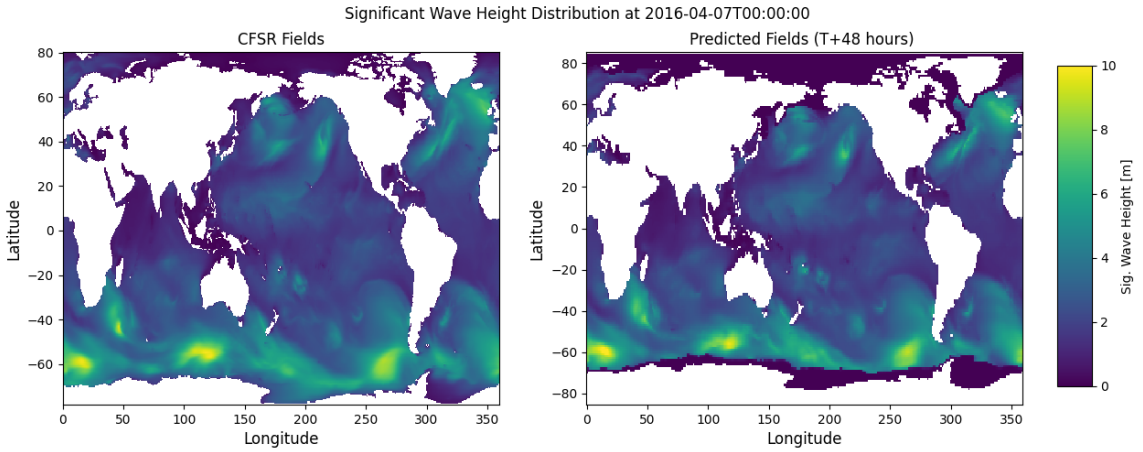

Figure 3. Domain of the dataset. Land masks are represented by white and can't be accessed during ship routing.

A. Training performance

We train the network for 400 epochs and measure the performance on the validation dataset. The results obtained are displayed in Figure 4.

Figure 4 shows the loss function of the proposed architecture during the training step for 200 epochs. The value of the loss function decreases gradually as the number of epochs increases. This suggests that the function is learnable and the network is able to learn the branching rules.

B. Comparison of various deep learning algorithms

This subsection compares the various deep learning methods used in this study during the validation phase. Specifically, the methods tried are Diffusion Convolution Layers, Graph Attention Layers, and a combination of both. The results obtained are displayed in Table III.

Table III shows the performance of various machine learning methods on the benchmark dataset. We can see that the machine learning algorithm using GDF and GAT layers outperforms the machine learning algorithms using individual layers.

C. Results on the test dataset with and without threshold relaxation

In this subsection, we compare the performance of the best model, i.e., GDF+GAT, on the test problems not included in the training process. We run one instance without the Threshold Relaxation (TR) scheme and another without the scheme. For the algorithm run with TR, the threshold is set to 0.0125 so that the least number of false negatives occurs. The results obtained are displayed in Table IV.

Analogous to the other ML-based B&B algorithms [17, 21, 25], we run the traditional Dijkstra algorithm with the same number of nodes (65.4%) used by the ML algorithm. The Dijkstra algorithm run using the limited node selection can only find the solution to 52 problems with 15.2% optimality gap. This suggests the efficacy of the proposed method compared to the Dijkstra algorithm in finding the solution to the vessel routing problem even when a limited number of nodes are explored (65.4% in our case). We also ran another experiment in which the program terminates when the algorithm finds the endpoint rather than after visiting all the nodes. While the OG is 9.2% in this case, 97.4% of the total nodes need to be visited on an average per problem. For fair comparison, we also run the proposed method till optimality. We find that the proposed method requires 98.3% nodes in average for optimality which is less than 99.7% for Dijkstra algorithm. This Optimality Gap and the node exploration ratio demonstrate that the ML-guided approach is better when the brute-force approach of visiting all the nodes is not desired or when calculating the edge weights is expensive, and a quick solution is expected.

D. Results on a single problem

In this subsection, we compare and contrast the paths obtained using Dijkstra shortest path algorithm, which continues till the whole nodes are explored, and the path obtained using the proposed algorithm. The Dijkstra algorithm is run on the static graph in which the node weights are the resistance encountered by the vessel due to the waves. The results of a single problem are displayed in Figure 5.

Figure 5 shows that the path obtained using the proposed algorithm is 1.8% longer than the optimum route. To verify the solution is optimum in Dijkstra or other algorithms, all the nodes must be visited, which might not be feasible in a large problem space. However, only 14.53% of the total nodes need to be visited by our proposed algorithm to obtain the near-optimum path. The near-optimum path obtained while only visiting 65.4% of the nodes demonstrates the usefulness of the proposed algorithm for vehicle routing problems when the total number of nodes explored is the main criterion for the algorithm selection.

VI. Discussion

This paper explores Graph Neural Network (GNN)-based approach to learning branching rules in the branch-and-bound framework. Assuming that the weather forecasts give complete information, we have explored a machine learning method that performs close to the results generated with the optimum

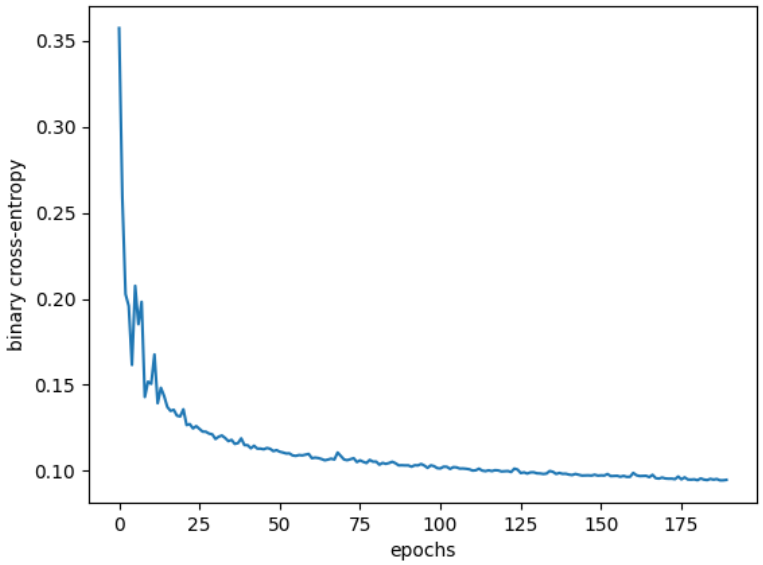

Figure 4. Loss function during the training step. The figure shows that the algorithm gradually learns the function and the loss decreases as the number of epochs increases.

Table III
DEEP LEARNING METHODS TESTED WITHOUT THE TR PROCEDURE

Method	ROC AUC	OG (%)	Not solved (out of 138)	Explored (%)
GDL	0.743	11.52	105	78.4
GAL	0.761	9.43	81	72.5
GDF + GAT	**0.820**	**8.40**	**71**	**67.6**

In the above table, GDL, GAL, and OG refer to the Graph Diffusion Layers, the Graph Attention Layers, and the Optimality Gap, respectively. **Bold** represents the best value.

Table IV
PERFORMANCE ON THE TEST PROBLEMS WITH AND WITHOUT THRESHOLD RELXATION

Method	Not Solved	Explored (%)
Without TR	71	67.6
With TR	**0**	**65.4**

In the above table, With TR, and Without TR refer to the algorithm run with threshold relaxation procedure, and the algorithm run without the procedure, respectively. **Bold** represents the best value. The initial threshold and the relaxation of 0.5 and 0.8 are used for the algorithm run with the threshold relaxation procedure. We use the static threshold of 0.0125 for the algorithm run without the proposed procedure.

solver while exploring significantly fewer nodes. The use of imitation learning allows us to learn the solver with less training compared to the other methods [17, 18, 21].

Table IV shows the performance of the machine learning algorithm with and without the Threshold Relaxation procedure. We find that normally, the algorithm is not able to generalize well to the test problems if the problems in the test set are significantly different than the problems in the training set. However, with the iterative relaxation of the threshold, the algorithm is able to solve the problem. This ability comes at the expense of the total numbers of nodes explored. In any case, since the threshold can be relaxed iteratively to 0, which corresponds to all the nodes visited, our algorithm guarantees that all the problems are solved.

The performance of the GDF + GAT encoder-decoder Graph Neural Network (GNN) compared to the individual layers suggests that the complexity of deep learning algorithms can improve performance. The improved performance might be a good tradeoff to the model complexity in some cases when Graphical Processing Units (GPUs) resources are available. However, the TR procedure used in this paper should enable using classifiers with lower complexity since the thresholds are automatically adjusted depending on the problem.

The efficacy of our procedure can be explained by the fact that other similar works on different problems utilize a significantly higher number of iterations to solve the problems (in the order of 10^5). In our case, while the procedure cannot solve most problems using only the training data, the Threshold Relaxation procedure introduced in this study allows the problem to be solved with far fewer iterations. While the network trained on fewer iterations can fail to solve some problems which are significantly different than the ones in the training set, the Threshold Relaxation procedure

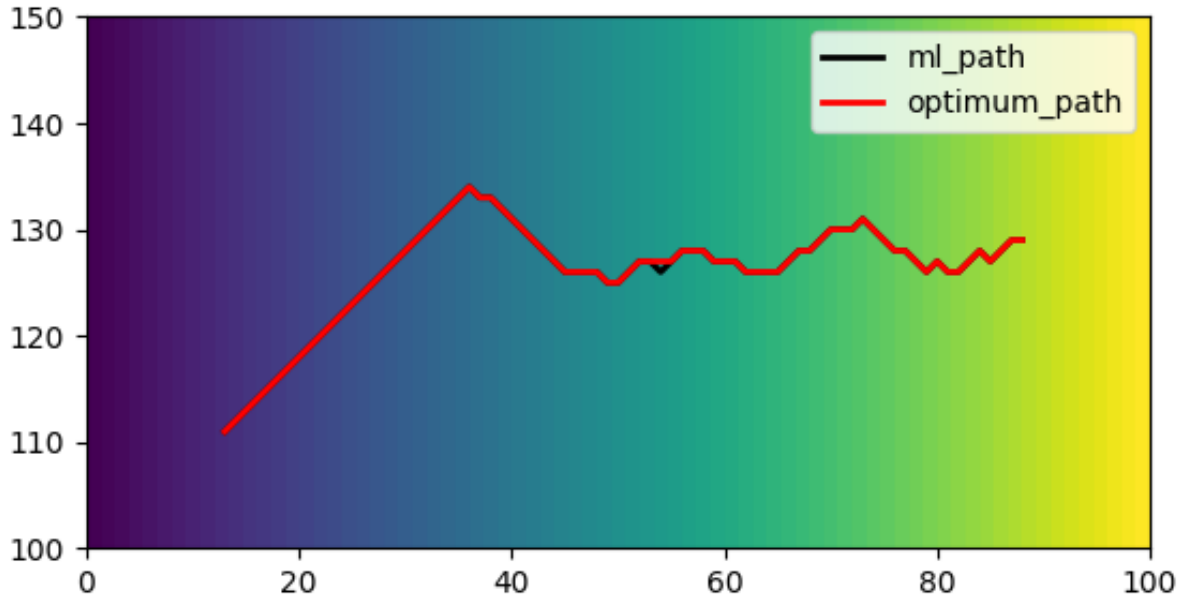

Figure 5. Results obtained on a test problem. The start point is 13 degrees latitude and 111 degrees longitude. The endpoint is 88 degrees latitude and 129 degrees longitude. The algorithm is optimized for minimum fuel cost during the voyage.

guarantees success in solving the problem in any domain, which can even be different than the problems in the training set. This is achievable since we lower the threshold till the endpoint is met.

Note that the imitation learning used in this study tries to build a model miming the Oracle (expert). One caveat of this process is that the model can not outperform the expert. The model, however, can perform a quick search close to the optimum with a speedup compared to the traditional solvers. This speedup is possible because the model does not need to visit all the nodes if it does not need to verify that the solution is optimal. The policy learned during the training phase of imitation learning can later be used in a reinforcement learning setting for further improvements. One limitation of our study is that the weather forecasts are assumed to be deterministic, which might not be, especially when encountering abnormal ocean/wind waves. An obvious extension of our approach would be to design a Partially Observed Markov Decision Process (POMDP), which does not assume that the weather information is complete. Moreover, a dynamic scheme similar to the VISIR can account for weather forecasts updated during the planning window.

VII. Conclusion

This paper introduces a Threshold Relaxation procedure for neural networks for solving shortest-path problems, which guarantees the success of reinforcement learning algorithms trained in a supervised manner in different training and test sets. This success is also guaranteed even when the problems in the test set differ from those in the training set at the expense of node exploration. This guarantee is possible since the threshold can be lowered iteratively and nodes chosen till the endpoint is reached in shortest-path problems. The Threshold Relaxation procedure introduced in this paper also allows the neural networks to be trained for a significantly fewer number of epochs (200 in our case) while still guaranteeing that all the problems in the test set are solved. In addition to introducing a Threshold Relaxation algorithm, we have also compared various graph neural networks and found that combining Graph Attention Networks and Graph Diffusion Networks achieves the best performance in the dataset used in this study. Our study should be a benchmark for future studies on the shortest-path weather routing of surface vessels. Similarly, the Threshold Relaxation procedure introduced in this study should apply to different domains in the usage of neural networks to solve shortest-path problems at the expense of optimality when the test set might be different from the training set in supervised reinforcement learning problems. This procedure is also applicable when the computational capacity only allows the networks to be trained for fewer epochs such that the network cannot fully generalize to a problem domain.

ACKNOWLEDGMENT

This work was partly supported by the U.S. Department of the Navy, Office of Naval Research (ONR), and Naval Research Laboratory under contracts N00173-20-2-C007 and N00173-20-2-C007, respectively.

REFERENCES

[1] P. Pokhrel *et al.*, "A transformer-based regression scheme for forecasting significant wave heights in oceans," *IEEE Journal of Oceanic Technology*, 2022.

[2] R. Zoppoli, "Minimum-time routing as an n-stage decision process," *Journal of Applied Meteorology*, vol. 11, no. 3, pp. 429–435, 1972. DOI: 10 . 1175 / 1520 - 0450(1972)011<0429:mtraas>2.0.co;2.

[3] B. Garau, A. Alvarez, and G. Oliver, "Path planning of autonomous underwater vehicles in current fields with complex spatial variability: An a* approach," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 194–198. DOI: 10.1109/ROBOT.2005.1570118.

[4] D. Rao and S. B. Williams, "Large-scale path planning for underwater gliders in ocean currents," *Australasian Conference on Robotics and Automation (ACRA),*, Dec. 2009.

[5] J. Wang *et al.*, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020. DOI: 10.1109/TASE.2020.2976560.

[6] J. Wang, W. Chi, C. Li, and M. Q.-H. Meng, "Efficient robot motion planning using bidirectional-unidirectional rrt extend function," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1859–1868, 2022. DOI: 10.1109/TASE.2021.3130372.

[7] T. Inanc, S. Shadden, and J. Marsden, "Optimal trajectory generation in ocean flows," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, 674–679 vol. 1. DOI: 10.1109/ACC.2005.1470035.

[8] D. Kruger, R. Stolkin, A. Blum, and J. Briganti, "Optimal auv path planning for extended missions in complex, fast-flowing estuarine environments," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 4265–4270. DOI: 10.1109/ROBOT.2007.364135.

[9] D. Jones and G. A. Hollinger, "Planning energy-efficient trajectories in strong disturbances," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2080–2087, 2017. DOI: 10 . 1109 / LRA . 2017 . 2719760.

[10] W. Zhang, T. Inanc, S. Ober-Blobaum, and J. E. Marsden, "Optimal trajectory generation for a glider in time-varying 2d ocean flows with a b-spline model," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1083–1088. DOI: 10.1109/ROBOT.2008.4543348.

[11] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 418–429, 2004. DOI: 10.1109/JOE.2004.827837.

[12] S. Guo *et al.*, *An autonomous path planning model for unmanned ships based on deep reinforcement learning*, Jan. 2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7013856/.

[13] T. Phiboonbanakit, T. Horanont, V.-N. Huynh, and T. Supnithi, "A hybrid reinforcement learning-based model for the vehicle routing problem in transportation logistics," *IEEE Access*, vol. 9, pp. 163 325–163 347, Nov. 2021. DOI: 10.1109/access.2021.3131799.

[14] R. Addanki, V. Nair, and M. Alizadeh, *Neural large neighborhood search*, Oct. 2020. [Online]. Available: https://openreview.net/forum?id=xEQhKANoVW.

[15] A. Hottung and K. Tierney, *Neural large neighborhood search for the capacitated vehicle routing problem*, Nov. 2020. [Online]. Available: https://arxiv.org/abs/1911.09539.

[16] K. Zhang, A. Koppel, H. Zhu, and T. Başar, "Global convergence of policy gradient methods to (almost) locally optimal policies," *SIAM Journal on Control and Optimization*, vol. 58, no. 6, pp. 3586–3612, Jan. 2020. DOI: 10.1137/19m1288012.

[17] E. Khalil *et al.*, *Learning to branch in mixed integer programming*, Feb. 2016. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10080.

[18] P. Gupta *et al.*, *Hybrid models for learning to branch*, Oct. 2020. [Online]. Available: https://arxiv.org/abs/2006.15212.

[19] M. Gasse *et al.*, *Exact combinatorial optimization with graph convolutional neural networks*, Oct. 2019. [Online]. Available: https://arxiv.org/abs/1906.01629.

[20] P. Veličković *et al.*, *Graph attention networks*, Feb. 2018. [Online]. Available: https://arxiv.org/abs/1710.10903.

[21] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik, *Learning to branch*, May 2018. [Online]. Available: https://arxiv.org/abs/1803.10150.

[22] A. Hottung, S. Tanaka, and K. Tierney, *Deep learning assisted heuristic tree search for the container pre-marshalling problem*, Sep. 2019. [Online]. Available: https://arxiv.org/abs/1709.09972.

[23] D. Karapetyan, A. P. Punnen, and A. J. Parkes, *Markov chain methods for the bipartite boolean quadratic programming problem*, Jan. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221717300061.

[24] J. Song, R. Lanka, Y. Yue, and B. Dilkina, *A general large neighborhood search framework for solving integer linear programs*, Dec. 2020. [Online]. Available: https://arxiv.org/abs/2004.00422.

[25] V. Nair *et al.*, *Solving mixed integer programs using neural networks*, Jul. 2021. [Online]. Available: https://arxiv.org/abs/2012.13349.

[26] D. Ding *et al.*, *Provably efficient safe exploration via primal-dual policy optimization*, Oct. 2020. [Online]. Available: https://arxiv.org/abs/2003.00534.

[27] G. Mannarini, D. N. Subramani, P. F. J. Lermusiaux, and N. Pinardi, "Graph-search and differential equations for time-optimal vessel route planning in dynamic ocean waves," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3581–3593, 2020. DOI: 10.1109/TITS.2019.2935614.

[28] G. Mannarini *et al.*, *Visir-i: Small vessels – least-time nautical routes using wave forecasts*, May 2016. [Online]. Available: https://gmd.copernicus.org/articles/9/1597/2016/.

[29] T. Lolla, P. F. Lermusiaux, M. P. Ueckermann, and P. J. Haley, "Time-optimal path planning in dynamic flows using level set equations: Theory and schemes," *Ocean Dynamics*, vol. 64, no. 10, pp. 1373–1397, 2014. DOI: 10.1007/s10236-014-0757-y.

[30] Y. Li, R. Yu, C. Shahabi, and Y. Liu, *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting*, Feb. 2018. [Online]. Available: https://arxiv.org/abs/1707.01926.

[31] *Recommended 7.5-02 -03-01.1 procedures and guidelines - ittc*, 2008. [Online]. Available: https://ittc.info/media/1587/75-02-03-011.pdf.

[32] *Https://www.ittc.info/media/8103/75-02-07-022.pdf*, 2011. [Online]. Available: https://www.ittc.info/media/8103/75-02-07-022.pdf.

[33] J. Holtrop, *A statistical re-analysis of resistance and propulsion data: Semantic scholar*, Jan. 1984. [Online]. Available: https://www.semanticscholar.org/paper/A-statistical-re-analysis-of-resistance-and-data-Holtrop/98b237b4f69618262121f27a33b3483ebb8f3dbf.

[34] V. G. Goecks *et al.*, *Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments*, Apr. 2020. [Online]. Available: https://arxiv.org/abs/1910.04281.

[35] H. Linmei *et al.*, "Heterogeneous graph attention networks for semi-supervised short text classification," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4821–4830. DOI: 10.18653/v1/D19-1488. [Online]. Available: https://aclanthology.org/D19-1488.

[36] S. Ross, G. J. Gordon, and J. A. Bagnell, *A reduction of imitation learning and structured prediction to no-regret online learning*, Mar. 2011. [Online]. Available: https://arxiv.org/abs/1011.0686?context=cs.