TRANSNAV
http://www.transnav.eu
the International Journal
on Marine Navigation
and Safety of Sea Transportation
Volume 19
Number 1
March 2025
DOI: 10.12716/1001.19.01.12

Towards Improved Ship Weather Routing Through Multi-Objective Optimization with High Performance Computing Support

M. Abdalsalam & J. Szłapczyńska
Gdansk University of Technology, Gdańsk, Poland

ABSTRACT: Maritime transport remains integral to the global economy, facilitating the cost-efficient and scalable movement of cargo and individuals over varying distances. Modern and effective ship routing solutions not only minimize voyage time and operational costs (including fuel consumption), but also improve resource allocation and environmental sustainability. Their planning process relies heavily on optimization algorithms capable of addressing numerous environmental and operational constraints, particularly in the context of dynamic and often unpredictable weather conditions.
A widely adopted approach in the literature is to formulate the ship route optimization problem as a multi-objective optimization (MOO) task, incorporating both static and dynamic constraints. The complexity of this formulation increases significantly when uncertainties related to weather conditions and ship behaviour are introduced, further complicating the optimization process. Meta-heuristic algorithms have gained prominence as effective tools for addressing i.e. complex multi-objective, constrained and nonlinear problems. Despite their demonstrated computational efficiency, the overall process of ship weather route optimization often remains computationally intensive, posing significant challenges for real-time or near-real-time applications in operational maritime contexts.
High Performance Computing (HPC) emerges as a viable approach to overcome this limitation. HPC refers mostly to the use of advanced computational systems composed of parallel processing architectures (such as CUDA, OpenMP, MPI, among others) to solve much faster and more efficiently complex and data-intensive problems. Originating in scientific research domains, HPC technologies have rapidly evolved and are now being applied to support solving a wide range of problem in computer science and engineering. Employing HPC-enabled computations allows for designing a scalable and efficient framework for tackling the growing complexity of ship weather routing.
This paper discusses the possibilities of HPC integration with MOO for ship weather routing, aiming to demonstrate how HPC-enabled methodologies can improve the performance and real-life applicability of the routing systems.

1 INTRODUCTION

Maritime transport is the backbone of global logistics, is heavily relied upon to ship goods between countries and regions[1]. It also enhances national competitiveness by integrating countries into global trade networks. In recent years, trade and logistics exchanges have grown. According to the International Maritime Organization (IMO), around 11 billion tons of goods are transported by sea annually, accounting for over 80% of world trade by volume and more than 70% by value[2]. This makes it the dominant mode of

international trade due to its superior capacity to handle large quantities of goods at low costs[3]. However, it faces various challenges that affect its efficiency and effectiveness. A key issue is the rising cost of fuel due to global price fluctuations, which increases operating expenses. The sector also faces growing environmental pressures, including significant demands to reduce carbon emissions doubts regarding alternative fuel options. Maritime shipping accounts for approximately 3% of global carbon dioxide emissions[1]. As shown in Figure 1, since 1970, shipping emissions have more than doubled, rising from about 350 million to over 700 million metric tons by 2023[4]. According to Statista, the shipping sector accounted for 12% of transport-related CO_2 emissions in 2023, with 4.5% from domestic and 7.5% from international shipping[5].

The situation necessitates a shift toward more sustainable modes and technologies to reduce negative environmental impacts. On the other hand, climate change and extreme weather disrupted shipping operations, affecting global supply chains and increasing accident risks.

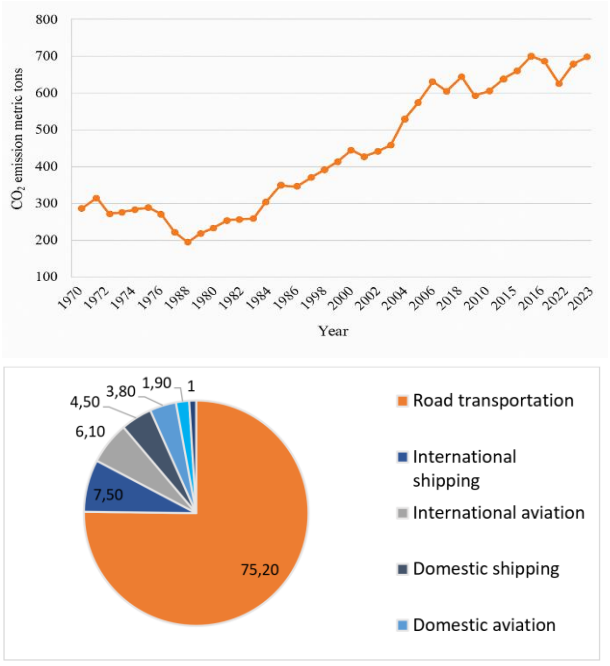

Figure 1. Environmental impact of shipping [4]. a) Carbon dioxide emissions from international shipping worldwide (1970–2023, million metric tons); b) Distribution of carbon dioxide emissions by transportation subsector worldwide (2023)

These factors highlight the need for effective measures to ensure safety and minimize weather-related disruptions. Considering these growing environmental and climate challenges, the need for innovative solutions that address emissions and safety challenges while enhancing efficiency and reducing costs has emerged. Among the most prominent of these solutions is the development of advanced vessel routing systems that take weather conditions into account, simultaneously improving efficiency and reducing fuel consumption. This is where technologies that determine optimal ship routes based on accurate weather data come in. These technologies contribute to reducing fuel consumption, thus reducing emissions and costs, while enhancing maritime safety by avoiding hazardous weather conditions.

Ship weather routing has gained attention in both academic and industrial circles. It primarily aims to improve voyage profitability and ensure the safety of ships and crews. Weather routing optimization identifies the best ship routes, balancing conflicting goals such as reducing fuel use-which may lead to unsafe speeds in rough weather-and ensuring safety, which might increase travel time and emissions. It also facilitates the optimization of ship operational parameters to achieve a balance between speed, fuel efficiency, dynamic loading, and the influence of weather conditions. The goal is to minimize voyage time, fuel usage, and safety risks, rather than to increase speed directly (which is limited by wave impact) [6]. Voyage optimization helps shipping companies to lower costs and improve competitiveness. Efficient operation across cost, energy, and safety is essential. Many vessels now a days use weather routing systems, which rely on meteorological and oceanographic data, ship characteristics, and route information. Depending on operator goals, optimization may focus on energy efficiency, travel time, safety, or a mix of these. The need for quick, effective decisions is growing due to fast-changing weather and navigation conditions. This calls for advanced computational methods to improve efficiency and support fast, reliable planning [7].

The main goals of weather routing are often in conflict, making MOO crucial to find the best trade-offs. This process is resource-intensive and requires analyzing large datasets—ranging from weather forecasts and vessel specifications to environmental limits. Modern routing solutions aim to reduce travel time and costs while improving sustainability and resource use. These solutions depend on optimization algorithms that can handle both environmental and operational constraints, especially under dynamic and unpredictable weather. A common method in the literature is to model routing as an MOO problem [8], incorporating both static (land and shallow waters) and dynamic (e.g. regions of high waves) constraints. The challenge increases with uncertainties in weather and vessel behavior, which make the optimization process far more complex. Metaheuristic algorithms are widely used to solve complex, constrained [9], and nonlinear MOO problems. Although effective, the optimization of ship routes under weather constraints remains computationally intensive, posing challenges for real-time or near-real-time applications.

Recently, High-Performance Computing (HPC) has become important for solving complex problems in science, engineering, and data analysis. It enables faster simulations, supports breakthroughs in Artificial Intelligence (AI), and helps researchers process massive datasets. The success of HPC is counting on multidisciplinary efforts, including supercomputers, parallel programming, network-based HPC environments, and HPC applications. HPC uses advanced systems with parallel processing (e.g., Compute Unified Device Architecture (CUDA), Open Multi-Processing (OpenMP), Message Passing Interface(MPI) to solve complex, data-heavy problems much faster[10]. Originally used in scientific research, HPC is now applied in many computer science and engineering fields.

The ship weather routing models need to process huge amounts of data and a heavy workload with the requirement of ultra-fast response. Therefore, it is of extreme necessity to promote computation efficiency by applying HPC to ship weather routing systems. Using HPC allows the design of scalable and efficient frameworks to manage the growing complexity of weather routing.

This paper tries to outline how HPC could be integrated with multi-objective optimization for more efficient ship routing. It aims to show how HPC can improve the performance and practical use of routing systems. The rest of this paper is structured as follows: the next two sections discuss related work on Ship Weather Routing (SWR), including its Multi-Objective Optimization (MOO) variants, and High Performance Computing (HPC). The following section introduces the few existing s.o.t.a. research on combined SWR and HPC. Then in the next section we identify research gaps and the need for further research for utilizing HPC in this context. The following section outlines a framework for HPC-based SWR solution. Finally, the last section concludes the material presented, including also some recommendations for future research.

2 SHIP WEATHER ROUTING (SWR)

Voyage planning is a fundamental part of safe ship navigation. It helps avoid risks in areas with high maritime traffic, shallow waters, or unfavorable weather conditions[11]. The IMO requires a comprehensive voyage plan approved before departure. This plan must address routing systems, environmental protection, and safety in line with the Safety of Life at Sea (SOLAS) convention[12]. Routing systems—such as Traffic Separation Schemes (TSS), precautionary areas, and zones to avoid—support navigational safety and environmental compliance. However, static hazards alone do not define routing challenges. Dynamic meteorological and oceanographic conditions also play a major role, with waves, winds, and currents directly affecting fuel use, speed, and route safety.

Ship weather routing research typically falls into two groups: global path planning, which seeks an optimal route between origin and destination based on environmental data, and local path planning, which focuses on short-term adjustments. Early log-term (global) weather routing methods relied on the isochrone algorithm introduced in 1957 [13]. It calculates the furthest distance a ship can travel in equal time intervals, assuming fixed environmental conditions. While simple and intuitive, when implemented as a computer algorithm it introduced issues such as isochronal loops. Hagiwara [14] addressed this by using sector-based waypoint expansion and pruning suboptimal subsectors. Later, Lin [15] proposed a three-dimensional variant using a dynamic grid, and Chen et al. [16] improved it further with the Isochrone-A* method, which dynamically adjusts the search space and uses an augmented cost function for multi-objective optimization.

With the rise of digital charts and computational capacity, graph-based algorithms such as Dijkstra's [17] have become widely adopted. These algorithms model sea areas as graphs, with edge weights adjusted for weather-induced speed losses. Sen and Padhy [17] introduced weights based on reduced speeds due to wind and waves. Takashima et al. [18] adapted Dijkstra's algorithm for fuel savings in coastal navigation by adjusting propeller revolutions. However, standard Dijkstra paths are often not smooth and may lack flexibility. Modifications have introduced features such as collision avoidance[19], MOO [20], and time-dependent variables. These enhancements improve applicability but do not resolve limitations in path realism and continuous control.

Dynamic programming (DP) offered an alternative, particularly for grid-based optimization. In early DP approaches to weather routing De Wit[21] and Calvert et al. [22] applied two-dimensional DP (2DDP) to optimize route selection under static ship settings. This approach divides the sailing region into a spatial grid and uses recursive optimization over time steps. To capture more realistic behaviors, later studies extended this to three-dimensional DP (3DDP), including control variables such as speed and engine power. Wei and Zhou[5] reduced the high computational demand of 3DDP by clustering similar control states. Still, DP-based methods remain demanding, especially when re-optimization is needed due to real-time changes in weather or vessel status.

3 MULTI-OBJECTIVE OPTIMIZATION (MOO) IN THE CONTEXT OF SWR

In recent years, Multi-Objective Optimization (MOO) has gained more attention, also in the context of ship weather routing, due to its ability of balancing the complex trade-offs between conflicting goals (in SWR: fuel efficiency, voyage time, emissions, and safety). While traditional methods remain as foundational approaches, they have been adapted to handle multiple goals- modified classical algorithms that generate Pareto-optimal solutions to advanced evolutionary algorithms tailored for simultaneous optimization of time, cost, safety, and emissions. In this context, MOO—particularly in hybrid forms—has demonstrated effectiveness, the gains associated with the speed of classical methods with the flexibility of modern algorithms. These hybrid models usually rely on traditional methods such as Dijkstra[23], isochrones or dynamic programming to generate initial routes or perform local searches while using metaheuristics to explore complex solution spaces.

Numerous MOO-based approaches and hybrid strategies have been introduced to determine the optimal route and support solution diversity. For example, improved multi-objective ant colony optimization (IMACO) [24] combined with the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) has shown potential in balancing navigation risk and fuel consumption. Similarly, hybrid versions of Particle Swarm Optimization (PSO) and Genetic Algorithms (GA), enhanced with heuristics, have been used to optimize ship fuel consumption mode and routes under environmental constraints[25]. Researchers have also customized the

Estimation of Distribution Algorithm (EDA) [26] for planning ship routes in a non-stationary environment (tidal water) with multiple goals by using probabilistic models of the best solutions instead of the usual crossover and mutation methods. In practice, EDA explores the solution space by sampling, selecting strong route candidates, and building probability distributions that reflect their key traits. This method avoids early convergence to weak solutions by keeping a diverse set of options, handles trade-offs between fuel and time more effectively, and adapts well to uncertain weather. Evolutionary algorithms, especially the Non-dominated Sorting Genetic Algorithm II (NSGA-II)[27], have been also applied. These methods provide Pareto-optimal solutions that help operators select routes based on specific preferences. The Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) further improves performance by dividing the problem using weight vectors. An advanced version, w-MOEA/D[28] which applies w-dominance[29] to standard MOEA/D algorithm, uses ensemble forecasts for uncertainty management, and accounts for real-world constraints such as bathymetry and emission zones. Despite these advantages, applying an MOO-based and hybrid approach in ship weather routing is computationally high and time-consuming. Evaluating multiple objectives requires searching a large multidimensional solution space and running various scenarios under varying weather conditions, which can significantly increase computation time. As a result, real-time or near-real-time application of MOO remains difficult without additional support.

Recent studies have introduced also Machine Learning (ML) and hybrid methods to improve weather routing performance. ML has been used to enhance heuristic and meta-heuristic functions in classic algorithms, guiding the search toward better candidate routes. For example, Chen and Mao [30] incorporated ML-based cost predictions into the Isochrone method to improve waypoint selection and route smoothness. Moreover, Shin et al.[31] proposed an improved A* algorithm for ship routing that integrates AIS and marine weather data to address limitations in existing methods. They introduced an adaptive grid system to reduce computational complexity and applied a 16-way search strategy to generate smoother and more economical routes. Machine learning models were used to estimate the ETA and evaluate route efficiency. Simulation results showed that their proposed method outperformed traditional A* routing by providing more economical paths. Hybrid methods often combine exact or heuristic algorithms with metaheuristics to balance quality and speed. In this regards, Qian et al. [32] integrated A* with genetic algorithms to handle speed variations over a dynamic grid. Similarly, Ma et al. [33] combined A* with an improved NSGA-III to support MOO under environmental and operational constraints. These approaches leverage the strengths of different algorithms—using graph search for reliable structure and evolutionary methods for flexible optimization. However, their performance still depends on the quality of initial solutions and the tuning of ML models, especially under uncertain and changing sea states.

Despite these improvements, no single method provides an optimal solution covering all cases. Heuristic and graph-based approaches provide speed but often ignore multi-objective trade-offs. Dynamic Programming (DP) provides global optima; however, it sacrifices scalability. Metaheuristics handle multiple objectives and constraints but can suffer from long convergence times and uncertain performance in dynamic settings. To benchmark and improve these approximations, VISIR [34] and its successor[35] provide reference solutions by solving the governing equations of motion under realistic weather conditions. VISIR-2 improves the model by adding factors such as wave angles, wind and current directions, and fuel emissions, which helps in finding routes that produce the least CO_2 and assists researchers in checking how accurate the approximate methods are. Recently, researchers' efforts have shown that ship weather routing has progressed beyond simple time or fuel minimization to advanced multi-objective optimization. Modern methods integrate route planning with control parameters such as speed, power, and trim while accounting for changing weather conditions and vessel behavior. Higher-resolution environmental data and ensemble forecasts have improved accuracy but increased the size of input data[28]. As a result, algorithms must handle larger solution spaces and evaluate more variables, which leads to higher memory usage and longer runtimes. This requirement is especially critical in methods such as 3D dynamic programming or population-based algorithms that evaluate many control states at each step. Machine learning and hybrid approaches arise to guide search and improve performance. However, implementation of it in real-time use remains limited due to computational costs and the need for dynamic updates during long voyages. Clean fuel ships and new propulsion types have more variables to optimize, which makes the problem even harder. Benchmark studies are still rare, which makes comparing methods and validating improvements difficult.

A potential solution able to solve the complexity, high dimensions, and memory requirements of MOO problems in ship weather routing is HPC. It provides an approach to splitting tasks across many processors or GPUs; HPC reduces run times and manages the increasing memory requirements. It lets models evaluate more route options and weather scenarios in near real time.

4 OVERVIEW OF HIGH PERFORMANCE COMPUTING (HPC)

Recently, HPC has become an essential tool across scientific communities. A vast increase in the use of HPC has been observed since its origins in the mid-20th century, and it has become essential in modern systems. Researchers struggle with computational problems when their problem needs a huge computation and memory resources. Since typical desktop computers do not work efficiently compared with HPC, the result is usually low-performance computing. HPC plays a key role in accelerating computational tasks and enabling large-scale data analysis across various fields. In practice, the concept of accelerating computation includes load balancing

across processors, optimized storage systems, and efficient data processing. In this regard, the term speedup in this context (e.g., in parallel computing), refers to the speed at which a task can operate when divided among multiple processors. Amdahl's Law clarifies that the non-parallel part of a program sets a limit on the total gain. For example, if 10% of the process cannot truly run in parallel (the process runs only sequentially), no amount of added processors can reduce the overall time below that 10%. This phenomenon makes sequential computing processes—tasks one after the other—a bottleneck. A bottleneck is something that slows down the overall process—such as a narrow point in a bottle that restricts flow. Moreover, the serial code represents the narrow point. In contrast, Gustafson's Law justifies that scaling the problem size with the number of processors can keep them fully utilized. As a result, a complex problem can be solved in the same time by using more processors, which helps reduce the impact of the serial portion.

These ideas relate to strong and weak scaling in HPC; the terms "weak scaling" and "strong scaling" refer to how well a system can maintain its performance as the number of processors or nodes is increased. Strong scaling looks at performance when the problem size stays fixed but the processor count grows. Weak scaling tests if performance holds steady when both the problem size and processor count grow together. These model help guide parallel algorithm design by highlighting trade-offs between speed, efficiency, and processor usage. That explains why HPC is considered a major leap in computing, offering the power to solve complex problems more efficiently and unlock new capabilities. HPC architecture takes many forms based on users' needs. Researchers can choose different ways to design HPC systems. Designing an HPC system may involve a combination of parallel computing, cluster computing, and grid/distributed computing strategies. HPC architectures are commonly categorized into three main components: compute, network, and storage. These components work together to enable high-speed processing of complex tasks.

Beyond these, systems also differ in how processors and memory interact. Memory-architecture models fall into three categories:

1. Shared-memory systems
   All processor cores access a single, global address space. Threads coordinate through lock or barrier primitives and share data via common variables.
2. Distributed-memory systems
   Each computing node has its own private memory. Nodes exchange data by passing explicit messages over an interconnect (for example, using MPI).
3. Hybrid systems
   Nodes combine multicore CPUs and accelerators (GPUs) and use both shared-memory and message-passing techniques. Within a node, threads share memory; between nodes, processes communicate by messaging.

Each model aligns with different programming frameworks (e.g., OpenMP for shared memory, MPI for distributed memory, CUDA or OpenCL for GPU acceleration). Efficient use of shared, distributed, and hybrid memory architectures depends on choosing the right parallel framework. The following table (Table 1) summarizes common frameworks and their typical use cases.

Table 1- HPC Programming Frameworks

Framework	Model	Target Hardware	Language Support	Use Case
MPI	Distributed	CPU clusters	C, Fortran, Python	Large-scale messaging
OpenMP	Shared	Multi-core CPU	C, C++, Fortran	Loop-level parallelism
CUDA	GPU	NVIDIA GPUs	C/C++	Data-parallel tasks
OpenCL	Heterogeneous	GPUs, CPUs	C	Portable kernels
OpenACC	Directive-based	GPUs	C, Fortran	Simplified GPU use

As mentioned above, the HPC relies on splitting tasks into smaller parts that can run at the same time through applying several technologies that are used to implement and create high-performance computing systems. Parallel processing is a method of running more than one processor to handle separate parts of a single task. Dividing different parts of a task among multiple processors minimizes the time required, achieves lower power consumption, and enables more efficient multitasking of the program.

Most systems with more than one central processing unit (CPU) are able to perform parallel processing, in addition to the multi-core processors commonly found on computers today. Another method uses GPUs, which contain thousands of lightweight cores designed to handle many repeated tasks in parallel. Moreover, SIMD (Single Instruction, Multiple Data) is another technique used in CPUs; it allows one instruction to process many data points at once, often speeding up loops. Some advanced models have incredible portability across various computer architectures and operating systems, such as MPI (Message Passing Interface). MPI allows several computers to work simultaneously on a problem by exchanging data across a network. MPI is common in large simulations run on clusters. On the other hand, the OpenMP API supports multi-platform shared-memory parallel programming in C/C++ and Fortran. The OpenMP API defines a portable, highly scalable model.

Furthermore, heterogeneous systems combine CPUs, GPUs, and other accelerators. This setup balances flexibility and speed by allowing each device to handle the parts it performs best on. These models form the core of HPC and make it possible to solve complex problems faster and at larger scales.

OpenMP and PThreads are used for multithreading on shared memory architectures. CUDA and OpenCL enable GPU programming, whereas OpenACC simplifies the use of GPUs through compiler directives. MPI is the de facto standard for message passing, whereas OpenSHMEM provides a shared memory programming model in distributed systems. Libraries such as Apache Hadoop and Spark support large-scale data processing based on a task-based and dataflow abstraction model. HPC programming requires trade-offs between low-level control and high-level abstraction. Lower-level interfaces such as MPI offer fine-grained control at the expense of requiring more intricate development. Higher-level programming models simplify development, possibly at the cost of

performance. Recent developments indicate a shift toward hybrid parallelism, energy-efficient computing, and emerging memory technologies such as NVRAM. These trends aim to increase efficiency and better suit emerging workloads such as AI applications and real-time analytics.

4.1 *High-performance computing (HPC) architectures*

HPC architectures include a broad range of hardware configurations designed to solve complex problems and perform intensive computational tasks. Traditional HPC clusters usually consist of homogeneous compute nodes connected through high-speed, low-latency. HPC systems support multiple layers of parallelism, from instruction-level parallelism within processor cores to node-level parallelism across clusters. One can then organize them as a hierarchical structure of parallelism categories that spans from the lowest level (instruction-level within a single core) through intermediate levels (thread-level, core-level) to the highest levels (node-level, cluster-level). Along with that, most HPC systems also contain complex memory hierarchies, including several cache levels, local memory, and often shared or distributed memory. Shared memory within a node enables rapid data exchange between threads but requires careful coordination to avoid race conditions. This architecture demands close attention to data locality and movement since performance degrades quickly when data must travel farther from the processor. Figure 2 presents a three-tiered view of HPC architectures that includes, at the top, the Parallel Programming Model layer, which encompasses message-passing (MPI) for process-level coordination and shared-memory/threading approaches (OpenMP/PGAS) for intra-node execution; the middle Middleware layer comprises communication libraries (MPI/PGAS/RDMA), job scheduling and resource managers (e.g., Slurm, Torque, Sun Grid Engine), and parallel file systems (Lustre, NFS, GPFS) that connect compute tasks and data movement across nodes; and at the base, the Infrastructure layer includes physical compute engines (CPUs, GPGPUs, FPGAs), network fabrics (Infiniband/Ethernet, iSCSI/FC), operating systems with numerical and accelerator libraries (BLAS/LAPACK, CUDA/OpenCL), and centralized storage arrays (SAN/NAS), illustrating how hardware, system software, and programming frameworks interlock to drive large-scale scientific applications.

Recently, graphical processing units (GPUs) have become a major component of modern HPC systems. The emergence of Graphics Processing Units (GPUs) transformed HPC architecture. Originally built for rendering graphics, GPUs evolved into general-purpose accelerators. With thousands of simple cores, they handle data-parallel workloads efficiently. When aligned with suitable algorithms, GPUs can greatly outperform CPUs. Today, many of HPC computer systems are characterized by their both CPUs and GPUs to match different computational demands. This hybrid model boosts performance in fields such as deep learning, molecular dynamics, and computational fluid dynamics and computational fluid dynamics. However, software must be tuned to the GPU's execution model to exploit its full potential.

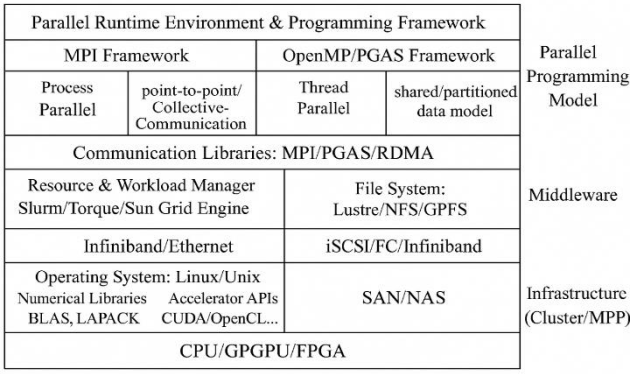

Figure 2. HPC architecture[36]

Widely used frameworks such as MPI, OpenMP, and CUDA provide fine-grained control over resources. Moreover, alternatives techniques such as OpenCL and Heterogeneous-compute Interface for Portability (HIP) enable code portability across different hardware. Other techniques, including Data Parallel C++ (DPC++) and C++ Single-source Heterogeneous Programming for OpenCL (SYCL), offer modern C++-based models for writing portable parallel code. Directive-based approaches such as Open Accelerators (OpenACC) simplify GPU adoption without major code changes. Additional libraries such as Performance Portability Programming EcoSystem (Kokkos) and RAJA Performance Portability Layer (RAJA) support writing portable code across CPUs and GPUs. Such programming models help developers optimize hybrid HPC systems while maintaining flexibility across architectures.

4.2 *Implementation of High-Performance Computing in different domain*

Today, HPC systems are revolutionizing scientific research by providing the power needed to tackle complex problems across many fields. These systems let researchers process large datasets, run detailed simulations, and build predictive models with unmatched speed and accuracy. Engineers use it for mechanical design, device testing, and product development, minimizing costs and increasing efficiency. In the automotive sector, HPC drives digital simulations and safety assessments. Energy companies apply it to reservoir modelling and subsurface geology, evaluating resource potential and estimating costs.

Despite its broad adoption, HPC remains underused in maritime applications—especially ship weather routing. Most routing systems still rely on sequential programming or on low-parallelism methods that have been implemented on some algorithms, such as genetic algorithms or graph-based searches (A*, Dijkstra). On the other hand, related maritime domains, such as coastal flooding, vessel tracking, and ship-ice modelling, have seen some HPC use, but ship routing lags. Ship routing under dynamic weather conditions demands frequent recalculations of forecasts, wave states, wind directions, and ship constraints.

Recent studies highlight HPC's potential impact on maritime research. For example, a researcher in cooperation with the National Supercomputing Centre ran full-order fluid–structure interaction simulations

on a 14.4 million-cell [37]. They used the PETSc library — an open-source toolkit for solving large-scale scientific problems— and a GMRES solver to solve big, complex problems at the same time on five computers, (typical multi-core desktop ones, with 24 cores, using Intel® Xeon® E5-2690 v3 CPUs (2.60 GHz). This configuration reduced runtimes from days to mere hours. It let researchers sweep through dozens of design parameters in one campaign—something impossible on smaller clusters. By handling high cell counts and fine time steps, the simulations captured millimeter-scale wave features and non-linear wave–wave interactions. Building on these simulations, the team developed data-driven digital twins of ships and offshore structures. The twins run in near real time, ingesting live environmental data to predict structural responses under changing loads. Such examples emphasize the ability of HPCs to run ensemble forecasts, whereas uncertainty quantification lets operators explore multiple scenarios in minutes. This capability supports the adaptive control of wave paddles in test basins and, ultimately, onboard decision support for vessels at sea. Yet most ship routing systems stick to legacy solvers, even though modern parallel methods could boost performance; they still rarely use these advances. The enhanced ship routing model should implement HPC strategies such as combining distributed computing for broad tasks, shared-memory threads for local work, and hardware accelerators for compute-intensive operations.

The maritime sector now faces rapid changes due to digitization, decarbonization, and safety demands. As ships tap real-time forecasts and IoT data, computing needs will surge. HPC could address these needs by enabling live, multi-objective route planning that balances safety, fuel use, and transit time. Traditional methods remain dominant in current systems; enhanced parallel computing approaches offer significant untapped potential. However, it should be noted that developing HPC solutions is not easy; it demands deep knowledge of parallel programming, memory management, and hardware architecture. To fully utilize HPC, developers must create routing algorithms that are specifically designed to take advantage of the latest parallel hardware and programming models. Many developers avoid HPC methods because they demand deep expertise in parallel programming and system optimization. Writing code for multi-level parallelism requires understanding of memory models, concurrency, and hardware architecture. These challenges raise development costs and extend project timelines as a result, teams stick to simpler sequential or limited-implementation.

5 SHIP WEATHER ROUTING TECHNIQUES BASED ON HPC

Multi-objective ship weather routing balances safety, fuel use, and voyage time by treating each as an optimization objective. The process evolves route candidates over many generations. Each route is evaluated on passage time, fuel consumption, and safety, yielding a set of non-dominated solutions known as the Pareto front. The voyage is divided into stages, and at each stage the algorithm records the best trade-off among objectives for each state. This process builds a map of optimal choices. To compare options, the objectives might be combined into a single score using user-defined weights, but thus the solution may overlook some Pareto-optimal routes.

Ship weather routing is computationally complex because environmental conditions change dynamically (in space and time). Wind, waves, and currents might be represented on a high-resolution grid. Each grid point holds forecasts for certain number of hours. Evaluating a route requires sampling forecasts at each step and multiplying data readings. As a result, datasets can span thousands of latitude–longitude cells over dozens of time steps and require gigabytes of RAM. Tracking constitutes hundreds of route candidates across multiple weather scenarios can push memory needs into tens of gigabytes. These demands make the problem a clear fit for HPC. Indeed, HPC meets these demands by rapidly processing large datasets and executing routing algorithms in parallel. It supports the conflicting goals of travel time, fuel use, and safety under unpredictable sea conditions [38]. Using HPC, one can evaluate numerous route options with high-resolution environmental data in real time [39]. Traditional serial methods cannot meet this requirement. A key advantage—especially with GPU acceleration—is parallelizing route evaluations. For example, implementing Non-local Kernel Density Estimation (NLKDE) on GPUs enabled real-time trajectory visualization of AIS data [40]. Similar methods can be applied to ship routing by evaluating cost functions over thousands of paths using multithreading, GPU or FPGA accelerators, and SIMD architectures. Models such as OpenMP and CUDA support maritime applications, allowing tasks such as segment-based risk analysis or energy prediction to run in parallel.

Integrating HPC into ship routing reduces simulation time, improves accuracy through higher-resolution data, and supports real-time evaluation of diverse options and constraints. It also addresses growing computational needs in the maritime sector. A notable example is the Sailing Assistance Application (SAA) by Życzkowski et al[19]. the first parallel implementation of deterministic weather routing for sailboats. Deployed in a cloud environment with OpenMP, SAA achieved up to 79.1% speedup in routing components while preserving accuracy. It divided tasks into navigation area generation and sailing condition modeling using shared-memory architectures. However, SAA focuses on sail-assisted vessels, not the dominant mode of commercial shipping – and uses Dijkstra's algorithm, which is less optimal for large-scale or multi-objective problems. Their proposed method implements HPC techniques only through OpenMP, which means it does not take advantage of other HPC techniques. Choosing HPC techniques requires analyzing various options, such as GPU acceleration or distributed memory models (MPI), which makes the proposed approach more difficult to scale for large areas and high-resolution grids.

To generalize and improve HPC-based routing, one might adopt a vessel model that supports a generalized engine-driven ships. A multi-level parallelism framework combining MPI, OpenMP, and GPU acceleration would enhance scalability.

Algorithmically, parallelized multi-objective solver can better match real-world needs. Dynamic rerouting and adaptive grid resolution would enable responsive and efficient navigation. These enhancements would yield a versatile, high-performance system across vessel types and scenarios. As presented in another study[32] Segment Parallel A* (SPA*) was presented. SPA* applies HPC principles to optimize maritime path planning under complex environmental conditions. It divides the route into sub-routes and distributes computation across multiple CPU cores using MATLAB.

Each segment uses an enhanced A* algorithm that balances risk, distance, and speedups from (as reported by the Authors) five to over 12,425 times compared to Dijkstra, A*, Bidirectional A*, Ant Colony Optimization, Harris Hawks Optimization, and Sparrow Search Algorithm. SPA* efficiently handles high-resolution, multi-constraint data, including real-time wind, wave, and topographic risk inputs. Yet it relies solely on CPU parallelism and grid-based models, which may incur overhead at high resolutions or from excessive segmentation.

Most prior ship routing methods remain sequential, even though parallel techniques have been applied in related domains such as swarm-based optimization [41][42][43][44], graph computations [45], and robotic motion planning [46][47]. Integrating HPC with deterministic routing may reduce execution time, but it limits flexibility for multi-objective and real-time applications. In contrast, parallelizing metaheuristic MOO algorithms—such as NSGA-II, NSGA-III, or MOEA/D—offers a scalable alternative for full-scale ship weather routing. These methods can better exploit HPC, support dynamic constraints, and enable faster decision-making with broader optimization goals. A deterministic approach, in this context, has limited applicability.

6 THE LITERATURES GAP

The literature review presented in the previous sections revealed that ship weather routing has gained renewed interest recently as regulators and operators press to cut emissions and boost efficiency. Scholars have shifted their focus to ship weather routing, which employs multi-objective optimization to balance competing goals under varying weather conditions. But multi-objective ship weather routing significantly increases computational complexity compared to single-objective optimization. The multi-dimensional optimization space generates exponential growth in computational requirements; it often exceeds the available decision windows in practical navigation scenarios. This "curse of dimensionality" becomes particularly problematic when considering high-resolution environmental data needed for accurate routing. Moreover, ship weather routing requires timely decision-making, especially in dynamic weather conditions, which create significant limitations for providing real-time solutions to complex multi-objective routing problems. State-of-the-art methods can find globally optimal solutions under specific conditions. However, recent reviews highlight the increasing complexity of these methods as it handle multiple objectives, which increases computational demands[48]. While they offer strong guarantees, their efficiency drops with large-scale or dynamic problems. This becomes a key limitation in ship routing, where weather and ocean conditions change frequently and require continuous re-optimization.

These challenges have revealed HPC's potential for overcoming them and paved the way for broader use. Despite these developments, HPC remains underutilized in weather-based ship navigation so far. The literature review identified several significant research gaps at the intersection of HPC and multi-objective ship weather routing. Many ship routing algorithms do not effectively utilize the HPC techniques, such as parallel computing capabilities, multi-core and other HPC techniques and resources. Moreover, there is a lack of standardized frameworks (scalability tests, varying core counts, and problem sizes to show how runtime changes) to evaluate and compare HPC performance in ship routing applications for weather, hindering systematic improvement in the field. Limited discussion exists on how HPC resources should scale to accommodate increasing complexity in multi-objective optimization scenarios. The literature inadequately addresses how existing ship weather routing can overcome time constraints in dynamic weather routing scenarios requiring near real-time decision-making. Additionally, many existing methods for ship weather routing, such as pathfinding algorithms, depend on a step-by-step process that creates links between each step, making it difficult to run them at the same time. Thus, there is a clear need to assess ship weather-routing algorithms for implementation on high-performance computing platforms. This assessment must identify which steps can run in parallel, such as route evaluation, weather-data interpolation, and ship-performance prediction. The existing ship weather routing models face scalability limitations when handling global-scale routing problems with high-resolution environmental data and multiple optimization objectives. Furthermore, challenges in managing HPC resources in shared environments are evident but not thoroughly addressed in the literature. The computational challenges of processing large volumes of dynamic environmental data in HPC environments remain under-explored, and current ship-routing algorithms exhibit limited parallelization.

The inherent sequential nature of many route planning algorithms rely on sequential pathfinding routines that enforce dependencies between steps and hinder concurrent execution. Although some processes could run in parallel, some of the core ship weather routing algorithmic structure demands sequential processing, weakening the benefits of multi-core systems.

Recently, Machine Learning (ML) empowered ship weather routing has shown promise due to the emergence of ML and its role in various domains; however, there is insufficient research on effectively integrating these approaches with HPC for ship routing applications. ML models potentially address some of the inherent limitations of traditional optimization algorithms, but the computational demands of training and using complex models create additional challenges for HPC integration.

To address the existing challenges, we aim to outline a proposal based on implementation of HPC in the existing MOO-SWR algorithm.

7 THE PROPOSED APPROACH TOWARDS SHIP WEATHER ROUTING WITH MOO AND HPC

Despite major improvements, current methods still do not have a single system that can manage both engine-driven vessels and detailed grid models, particularly when the algorithm's execution time is limited. To address the existing challenges, we aim to propose a solution based on the implementation of HPC in the existing MOO-SWR algorithm.

The proposed approach begins with a review of recent literature and practical case studies to identify algorithms with strong potential for HPC implementations. Simultaneously, we'll analyze existing MOO-SWR methods to pinpoint components suitable for HPC techniques—such as parallelization across processors, shared-memory architectures, or specialized hardware—and then design and execute benchmarks to measure CPU/GPU utilization, memory usage, solution quality, and speedup relative to sequential baselines on identical hardware/software stacks.

Next, we'll evaluate these algorithms across various HPC software frameworks and hardware configurations, comparing their performance to sequential implementations. For example, to maximize efficiency, we may adopt a hybrid CPU–GPU architecture: GPUs will handle parallel tasks (e.g., environmental data interpolation and route-cost evaluation), while CPUs manage sequential route-optimization steps, thereby accelerating multi-objective optimization and containing computational growth.

The proposed solution will also explore integrating an ML-based forecasting model—trained on historical and real-time data—to predict short-term sea and weather conditions. These predictions will trigger targeted re-optimizations within the routing algorithm, avoiding full re-computations when changes are localized. Together, these measures enhance speed, accuracy, and resilience in ship weather routing.

Figure 3 presents part of the proposed solution, which ensures the ability to provide reliable route recommendations promptly. According to Figure 3, the framework gathers dynamic weather data, ship parameters, and business information. It maps the ship's speed and course according to time, cost, and CO_2 objectives. The framework then selects an MOO-SWR algorithm based on its capability to operate on HPC systems. Using the selected algorithm, it chooses processes that support HPC features—such as parallel processing—and specifies the hardware and software methods for the chosen solver. These methods may include genetic algorithms or distributed gradient searches, chosen at runtime based on solution quality. Finally, it compares solver performance, route efficiency, and resource use to determine the optimal HPC approach, routes, and metrics.

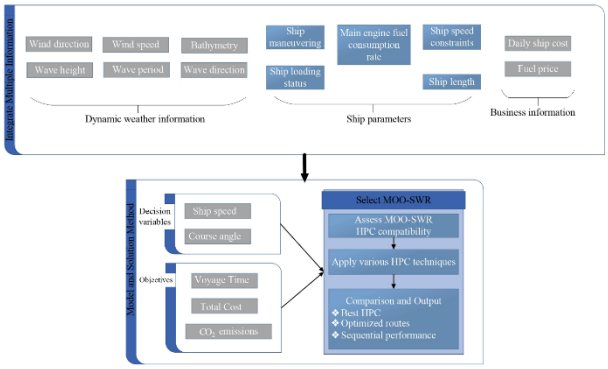

Figure 3. MOO-SWR-based HPC

8 CONCLUSIONS

Ship weather routing has emerged as a critical area of research and application in maritime transportation, with significant potential to improve safety, reduce fuel consumption, and minimize environmental impacts. The multi-objective-based HPC approach builds on the importance of weather-aware voyage planning and provides the computational muscle and methodological sophistication needed to turn routing into practice in real-world scenarios.

Accordingly, using High Performance Computing (HPC) for Multi-Objective Optimization (MOO) in weather-based ship routing is a major advance over simple, deterministic methods. Deterministic methods are fast and easy to use. In contrast, multi-objective methods balance multiple goals in complex, changing sea conditions. The pure MOO methods face challenges such as a multidimensional solution space, trade-off analysis, detailed environmental models, safety integration, high compute needs, data fusion and management, real-time updates, and strict verification and validation. Conversely, the HPC-supported MOO methods promise to solve the complex optimization problems and open opportunities for addressing these challenges with cost savings and more flexibility.

HPC may use different strategies and resources to achieve significantly more computing power than a standard computer. It relies on a combination of specialized hardware and software to enable complex, data-intensive applications to run at high speeds and allow for increased efficiency. HPC involves aggregating resources, such as supercomputers, clusters of computers, and others, to enable the parallel processing of complex calculations across numerous processors. Moreover, it offers a suite of strategies that utilize computational power for various tasks. These strategies can involve expanding computing resources; for instance by using parallel processing with MPI and OpenMP to distribute calculations across many cores. GPU acceleration with CUDA enables dealing with ensemble simulations for diverse weather and routing scenarios. Task-based frameworks and hybrid CPU–GPU architectures handle real-time adaptation and data fusion. These approaches help explore trade-offs, validate solutions, and meet the demands of complex routing. However, there is no guarantee from a theoretical perspective that selecting the HPC strategy from various options—such as MPI-based parallelism,

GPU acceleration, cloud-native clusters, or hybrid models—will yield the best result. Thus, the selection of suitable techniques requires analyzing and comparing these approaches. That analysis demands profound domain knowledge and strong programming skills. Despite these hurdles, multi-objective HPC methods can minimize fuel use, boost safety, and lower the environmental impact. With greater HPC availability and increasing computational demands for algorithms, these methods might gain broader use in weather-based ship routing. The MOO-SWR drive for efficiency, safety, and sustainability will spur further innovation in weather-based routing. Addressing these challenges and exploring diverse HPC strategies, researchers can build efficient models based on HPC to resolve difficult trade-offs in high-demand computational domains, such as maritime domains, in general and particularly for ship weather routing.

Future work will include searching for a validated unified HPC-based multi-objective ship routing framework that better matches real-world scenarios, enabling quick responses under actual conditions. The primary objective is to develop a highly efficient and adaptable ship weather routing method capable of handling large-scale environmental datasets and rapidly updating route plans in response to changing conditions, thereby ensuring reliable performance in time-critical, real-time scenarios.

REFERENCES

[1] S. Deng and Z. Mi, "A review on carbon emissions of global shipping," Mar. Dev., vol. 1, no. 1, pp. 1–10, 2023, doi: 10.1007/s44312-023-00001-2.

[2] ShipUniverse, "Top 30 Reasons Why Maritime Shipping Is the Backbone of Global Trade," 2024.

[3] J. Verschuur, E. E. Koks, and J. W. Hall, "Ports' criticality in international trade and global supply-chains."

[4] Statista, "Breakdown of CO$_2$ emissions in the transportation sector worldwide 2020, by subsector," 2021. [Online]. Available: https://www.statista.com/statistics/1185535/transport-carbon-dioxide-emissions-breakdown/

[5] W. Shao, P. Zhou, and S. K. Thong, "Development of a novel forward dynamic programming method for weather routing," J. Mar. Sci. Technol., vol. 17, no. 2, pp. 239–251, Jun. 2012, doi: 10.1007/s00773-011-0152-z.

[6] L. P. Perera and C. G. Soares, "Weather routing and safe ship handling in the future of shipping," Ocean Eng., vol. 130, pp. 684–695, 2017, doi: 10.1016/j.oceaneng.2016.09.007.

[7] L. Walther, A. Rizvanolli, M. Wendebourg, and C. Jahn, "Modeling and Optimization Algorithms in Ship Weather Routing," Int. J. e-Navigation Marit. Econ., vol. 4, pp. 31–45, Jun. 2016, doi: 10.1016/j.enavi.2016.06.004.

[8] J. Szlapczynska and R. Szlapczynski, "Preference-based evolutionary multi-objective optimization in ship weather routing," Appl. Soft Comput. J., vol. 84, Nov. 2019, doi: 10.1016/j.asoc.2019.105742.

[9] E. G. Talbi, M. Basseur, A. J. Nebro, and E. Alba, "Multi-objective optimization using metaheuristics: Non-standard algorithms," Int. Trans. Oper. Res., vol. 19, no. 1–2, pp. 283–305, Jan. 2012, doi: 10.1111/j.1475-3995.2011.00808.x.

[10] P. Czarnul, J. Proficz, and K. Drypczewski, "Survey of Methodologies, Approaches, and Challenges in Parallel Programming Using High-Performance Computing Systems," 2020, Hindawi Limited. doi: 10.1155/2020/4176794.

[11] Ü. Öztürk, M. Akdağ, and T. Ayabakan, "A review of path planning algorithms in maritime autonomous surface ships: Navigation safety perspective," May 01, 2022, Elsevier Ltd. doi: 10.1016/j.oceaneng.2022.111010.

[12] K. SKÓRA and A. WOLSKI, "VOYAGE PLANNING," Sci. J. Silesian Univ. Technol. Ser. Transp., vol. 92, pp. 123–128, Sep. 2016, doi: 10.20858/sjsutst.2016.92.12.

[13] James and Richard W, Application of wave forecasts to marine navigation. New York University, 1957.

[14] Hagiwara and Hideki, "Weather routing of (sail-assisted) motor vessels.," Technische Universiteit Delft, 1989.

[15] Y. H. Lin, M. C. Fang, and R. W. Yeung, "The optimization of ship weather-routing algorithm based on the composite influence of multi-dynamic elements," Appl. Ocean Res., vol. 43, pp. 184–194, 2013, doi: 10.1016/j.apor.2013.07.010.

[16] Y. Chen, W. Tian, and W. Mao, "Strategies to improve the isochrone algorithm for ship voyage optimisation," Ships Offshore Struct., 2024, doi: 10.1080/17445302.2024.2329011.

[17] D. Sen and C. P. Padhy, "An approach for development of a ship routing algorithm for application in the North Indian Ocean region," Appl. Ocean Res., vol. 50, pp. 173–191, 2015, doi: 10.1016/j.apor.2015.01.019.

[18] K. Takashima, B. Mezaoui, and R. Shoji, "On the fuel saving operation for coastal merchant ships using weather routing," Mar. Navig. Saf. Sea Transp., vol. 3, no. 4, pp. 431–436, 2009, doi: 10.1201/9780203869345.ch75.

[19] M. Zyczkowski and R. Szlapczynski, "Collision risk-informed weather routing for sailboats," Reliab. Eng. Syst. Saf., vol. 232, no. November 2022, p. 109015, 2023, doi: 10.1016/j.ress.2022.109015.

[20] C. U. Bottner, "Weather routing for ships in degraded condition," in Proceedings of the International Symposium on Maritime Safety, Security and Environmental Protection. Athens, Greece, 2007.

[21] S. Calvert, E. Deakins, and R. Motte, "A Dynamic System for Fuel Optimization Trans-Ocean," J. Navig., vol. 44, no. 2, pp. 233–265, 1991, doi: 10.1017/S0373463300009978.

[22] C. de Wit, "Proposal for Low Cost Ocean Weather Routeing," J. Navig., vol. 43, no. 3, pp. 428–439, 1990, doi: 10.1017/S0373463300014053.

[23] H. Wang, W. Mao, and L. Eriksson, "A Three-Dimensional Dijkstra's algorithm for multi-objective ship voyage optimization," Ocean Eng., vol. 186, no. June, p. 106131, 2019, doi: 10.1016/j.oceaneng.2019.106131.

[24] J. Yang, L. Wu, and J. Zheng, "Multi-Objective Weather Routing Algorithm for Ships: The Perspective of Shipping Company's Navigation Strategy," J. Mar. Sci. Eng., vol. 10, no. 9, 2022, doi: 10.3390/jmse10091212.

[25] W. Zhao, H. Wang, J. Geng, W. Hu, Z. Zhang, and G. Zhang, "Multi-Objective Weather Routing Algorithm for Ships Based on Hybrid Particle Swarm Optimization," J. Ocean Univ. China, vol. 21, no. 1, pp. 28–38, 2022, doi: 10.1007/s11802-022-4709-8.

[26] R. Dębski and R. Dreżewski, "Multi-Objective Ship Route Optimisation Using Estimation of Distribution Algorithm," Appl. Sci., vol. 14, no. 13, 2024, doi: 10.3390/app14135919.

[27] X. Li, H. Wang, and Q. Wu, "Multi-objective optimization in ship weather routing," 2017 Constr. Nonsmooth Anal. Relat. Top. (Dedicated to Mem. V.F. Demyanov), CNSA 2017 - Proc., pp. 1–4, 2017, doi: 10.1109/CNSA.2017.7973982.

[28] R. Szlapczynski, J. Szlapczynska, and R. Vettor, "Ship weather routing featuring w-MOEA/D and uncertainty handling," Appl. Soft Comput., vol. 138, p. 110142, 2023, doi: 10.1016/j.asoc.2023.110142.

[29] R. Szlapczynski and J. Szlapczynska, "W-dominance: Tradeoff-inspired dominance relation for preference-based evolutionary multi-objective optimization," Swarm Evol. Comput., vol. 63, no. March 2020, p. 100866, 2021, doi: 10.1016/j.swevo.2021.100866.

[30] Y. Chen, "Department of Mechanics and Maritime Sciences Isochrone-Based Voyage Optimization Methods to Increase Shipping Energy Efficiency," 2024.
[31] Y. W. Shin et al., "Near-optimal weather routing by using improved A* algorithm," Appl. Sci., vol. 10, no. 17, 2020, doi: 10.3390/app10176010.
[32] L. Qian, H. Li, M. Hong, Y. Qi, and Z. Guo, "Avoiding sudden maritime risk: A new variable speed route planning model by integrating Spatio-temporal dimension," Ocean Eng., vol. 288, no. P1, p. 115950, 2023, doi: 10.1016/j.oceaneng.2023.115950.
[33] D. Ma, S. Zhou, Y. Han, W. Ma, and H. Huang, "Multi-objective ship weather routing method based on the improved NSGA-III algorithm," J. Ind. Inf. Integr., vol. 38, no. January, p. 100570, 2024, doi: 10.1016/j.jii.2024.100570.
[34] 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 Trans. Intell. Transp. Syst., vol. 21, no. 8, pp. 3581–3593, 2020, doi: 10.1109/TITS.2019.2935614.
[35] G. Mannarini, M. L. Salinas, L. Carelli, N. Petacco, and J. Orović, "VISIR-2: Ship weather routing in Python," Geosci. Model Dev., vol. 17, no. 10, pp. 4355–4382, 2024, doi: 10.5194/gmd-17-4355-2024.
[36] F. Yin and F. Shi, A Comparative Survey of Big Data Computing and HPC: From a Parallel Programming Model to a Cluster Architecture, vol. 50, no. 1. Springer US, 2022. doi: 10.1007/s10766-021-00717-y.
[37] D. K. Panda, Ed., Supercomputing Frontiers, vol. 12082. in Lecture Notes in Computer Science, vol. 12082. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-48842-0.
[38] F. Sattler, B. Carrillo-Perez, S. Barnes, K. Stebner, M. Stephan, and G. Lux, "Embedded 3D reconstruction of dynamic objects in real time for maritime situational awareness pictures," Vis. Comput., vol. 40, no. 2, pp. 571–584, 2024, doi: 10.1007/s00371-023-02802-4.
[39] T. Nakaegawa, "High-Performance Computing in Meteorology under a Context of an Era of Graphical Processing Units," Computers, vol. 11, no. 7, 2022, doi: 10.3390/computers11070114.
[40] M. Liang, K. Liu, R. Gao, and Y. Li, "Integrating GPU-Accelerated for Fast Large-Scale Vessel Trajectories Visualization in Maritime IoT Systems," IEEE Trans. Intell. Transp. Syst., vol. 26, no. 3, pp. 4048–4065, 2025, doi: 10.1109/TITS.2024.3521050.
[41] E. Alhenawi, R. A. Khurma, A. A. Sharieh, O. Al-Adwan, A. Al Shorman, and F. Shannaq, "Parallel Ant Colony Optimization Algorithm for Finding the Shortest Path for Mountain Climbing," IEEE Access, vol. 11, no. January, pp. 6185–6196, 2023, doi: 10.1109/ACCESS.2022.3233786.
[42] S. A. M. Ali and E. H. Al-Hemiary, "A Parallel Implementation Method for Solving the Shortest Path Problem for Vehicular Networks," Proc. 2020 1st Inf. Technol. to Enhanc. E-Learning other Appl. Conf. IT-ELA 2020, pp. 121–126, 2020, doi: 10.1109/IT-ELA50150.2020.9253086.
[43] Z. Cheng, L. Zhao, and Z. Shi, "Decentralized Multi-UAV Path Planning Based on Two-Layer Coordinative Framework for Formation Rendezvous," IEEE Access, vol. 10, pp. 45695–45708, 2022, doi: 10.1109/ACCESS.2022.3170583.
[44] N. Edmonds, A. Breuer, D. Gregor, and A. Lumsdaine, "Single-source shortest paths with the parallel boost graph library," no. May 2014, pp. 219–248, 2009, doi: 10.1090/dimacs/074/09.
[45] J. Kim, "HGED: A Hybrid Search Algorithm for Efficient Parallel Graph Edit Distance Computation," IEEE Access, vol. 8, pp. 175776–175787, 2020, doi: 10.1109/ACCESS.2020.3026648.
[46] D. Josifoski, M. Gusev, V. Zdraveski, and S. Ristov, "Parallelization of Dijkstra's Algorithm for Robot Motion Planning: Is It Worth Increasing Speed Without Losing Accuracy?," 2022 30th Telecommun. Forum, TELFOR 2022 - Proc., pp. 1–4, 2022, doi: 10.1109/TELFOR56187.2022.9983755.
[47] J. H. Seo, H. Lee, and K. D. Kim, "A Parallelization Algorithm for Real-Time Path Shortening of High-DOFs Manipulator," IEEE Access, vol. 9, pp. 123727–123741, 2021, doi: 10.1109/ACCESS.2021.3109744.
[48] Y. Chen et al., "State-of-the-art optimization algorithms in weather routing — ship decision support systems: challenge, taxonomy, and review," Ocean Eng., vol. 331, no. April, p. 121198, 2025, doi: 10.1016/j.oceaneng.2025.121198.