Computational and Applied Mathematics (2024) 43:250
https://doi.org/10.1007/s40314-024-02756-w

Hybrid search method for Zermelo's navigation problem

Daniel Precioso[1,3] · Robert Milson[2] · Louis Bu[2] · Yvonne Menchions[2] · David Gómez-Ullate[1]

Received: 22 January 2024 / Revised: 22 April 2024 / Accepted: 23 April 2024 /
Published online: 20 May 2024

Abstract
In this paper, we present a novel algorithm called the Hybrid Search algorithm to tackle the Zermelo's navigation problem. This method can be regarded as an extension of the recent Ferraro–Martín de Diego-Sato algorithm to allow for further exploration in search for the global optimum, in situations of complex vector fields where many locally optimal trajectories exist. Our algorithm is designed to work in both Euclidean and spherical spaces and utilizes a heuristic that allows the vessel to move forward while remaining within a predetermined search cone centered around the destination. This approach not only improves efficiency but also includes obstacle avoidance, making it well-suited for real-world applications. We evaluate the performance of the Hybrid Search algorithm on synthetic vector fields and real ocean currents, demonstrating its effectiveness and performance.

Keywords Weather routing · Zermelo navigation problem · Optimization · Time optimal trajectories

✉ Daniel Precioso
dprecioso@faculty.ie.edu

Robert Milson
rmilson@dal.ca

Louis Bu
l.bu@dal.ca

Yvonne Menchions
vg217777@dal.ca

David Gómez-Ullate
dgomezullate@faculty.ie.edu

[1] School of Science and Technology, IE University, Paseo de la Castellana, 259, Madrid 28046, Spain

[2] Department of Mathematics and Statistics, Dalhousie University, Halifax, NS B3H 3J5, Canada

[3] Department of Computer Science, Higher School of Engineering, Universidad de Cádiz, Av. Universidad de Cádiz, 10, Puerto Real, Cádiz 11519, Spain

Mathematics Subject Classification 65K10 Numerical optimization and variational techniques · 65L10 Numerical solution of boundary value problems involving ordinary differential equations · 70-08 Computational methods for problems pertaining to mechanics of particles and systems · 70-10 Mathematical modeling or simulation for problems pertaining to mechanics of particles and systems · 70Q05 Control of mechanical systems · 49Mxx Numerical methods in optimal control · 49M15 Newton-type methods

1 Introduction

Maritime transport handles approximately 80% of the world's trade by volume (Unctad 2021). Despite its importance in world economy, the sector faces significant challenges, including rising fuel costs, stringent environmental regulations, and the need for safe navigation (Roberts et al. 2014; Zis et al. 2020). In this context, weather routing appears as an efficient solution to alleviate these problems.

The first mathematical formulation of weather routing was published by Zermelo (1930, 1931). Named by the community as the Zermelo's navigation problem (ZNP), the goal was to optimize travel time through spaces affected by external forces. While originally framed in the Euclidean space, it has expanded to more realistic models like the spherical Earth representation (Bao et al. 2004; Marchidan and Bakolas 2016). While minimizing travel time is still a popular optimization goal, current ZNP studies implement alternatives such as reducing fuel consumption (Lin et al. 2013; Walther et al. 2016).

Solving the ZNP has experienced a surge in academic popularity during recent years (Zis et al. 2020), partly due to the increasing focus on sustainability in the shipping industry (IMO 2020; Perera and Soares 2017). Most state of the art methods are based on one of three approaches. First we see **variational methods** (Ferraro et al. 2021), which ensure locally optimal paths around the initial guess. These algorithms are best suited for synthetic (smooth) vector fields, where they outperform other approaches. However, variational methods are limited in real applications as weather data is discrete, and thus require the use of interpolation algorithms (Garcelán 2023).

Next is **graph search** (Grifoll et al. 2022; Zyczkowski and Szlapczynski 2023), which tends to be exploitative and in turn fast. Graph search has become the most popular method to solve ZNP in real scenarios, as it performs well in discrete vector fields (Zis et al. 2020). For the same reason, these algorithms under-perform in synthetic (smooth) vector fields, and are hard to implement there (Garcelán 2023).

Finally, **evolutionary algorithms** (Kuhlemann and Tierney 2020; Zhao 2022; Grandcolas 2022) tend to be the most exploratory approach. These methods perform similarly well in both synthetic and real vector fields. Evolutionary algorithms are good at finding feasible routes even in complex scenarios, but the optimality is never guaranteed and are computationally expensive (Garcelán 2023).

Table 1 compares our approach with recent studies. We noticed that there remains a notable gap in creating an unified solution that balances computational efficiency and robust results. This paper introduces the Hybrid Search (HS) algorithm, an innovative approach that combines the strengths of variational methods and evolutionary algorithms.

HS implements a two-step evolutionary process, enabling automatic obstacle avoidance and feasible routes to the destination even in complex vector fields. HS path finding is done using a variational method, resulting is a piece-wise optimal trajectory. As a final smoothing step, HS integrates the Ferraro–Martín de Diego-Sato algorithm (FMS) algorithm (Ferraro

et al. 2021, 2022) to generate a locally optimal route. For this preliminary study, we will focus in minimizing the travel time, however HS can be easily updated to any other function as we will discuss by the end of the article. While Hybrid Search (HS) method performs better in synthetic (smooth) vector fields, it adapts well to real (discrete) scenarios due to its evolutionary algorithm, as we will show in the results.

The paper is structured as follows: Sect. 2 delves into the routing problem and the ZNP equations. Section 3 details our HS method, and Sect. 4 introduces synthetic and real benchmarks. The results of the HS algorithm are presented in Sect. 5, and Sect. 6 discusses these findings, underscoring the impact and future implications of our work in the realm of ZNP solutions.

The code to implement the Hybrid Search algorithm is publicly available in the following GithHub repository: https://github.com/daniprec/hybrid_ivp

2 Routing problem in navigation

The routing problem is about finding efficient routes for ships, considering ocean conditions. This complex challenge impacts fuel use, emissions, costs, and travel time. Our paper introduces the Hybrid Search (HS) algorithm, a mathematical and computational method for time-optimal routing, factoring in ocean currents. Our approach simplifies the problem with certain assumptions. We consider a stationary vector field for ocean currents, and that the ship maintains constant velocity relative to water. While HS can avoid small obstacles, it is not designed for optimal circumnavigation around significant patches of land.

To address the routing problem, we will use the classical formulation from Zermelo's navigation problem (ZNP). This section introduces the equations both on the plane and on the sphere.

2.1 Zermelo's navigation problem on the plane

This problem was proposed in 1931 by Zermelo (1931), is a classic time-optimal control problem, where its aim is to find **time minimum trajectories** under the influence of a drift vector

$$\vec{w}(x_1, x_2) = \langle w_1(x_1, x_2), w_2(x_1, x_2) \rangle$$

Table 1 Summary of recent weather routing studies

References	Objective function	Algorithm
This study	Travel time	HS(variational + evolutionary)
Zyczkowski and Szlapczynski (2023)	Travel time	Dijkstra (graph)
Ferraro et al. (2022)	Travel time	Discrete variational method
Grandcolas (2022)	Fuel consumption	Evolutionary
Grifoll et al. (2022)	Travel time	A* search (graph)
Zhao (2022)	Fuel + Time	Particle swarm (evolutionary)
Kuhlemann and Tierney (2020)	Fuel consumption	Genetic (evolutionary)

where x_1, x_2 are local coordinates, and where w_1, w_2 are the vector components chosen relative to a local frame. This drift vector can be interpreted as wind or water current. In small scale simulations, the coordinates and the vector components can be taken to be Euclidean. Once we pass to larger scale simulations that take into account the curvature of the Earth, the coordinates (x_1, x_2) indicate longitude and latitude (in degrees), while the vector components are taken relative to a local east-north framing (in meters).

The goal is to navigate from a specified initial point along a path that minimizes time, under the influence of $\vec{w}$, assuming the vessel provides constant thrust V (speed over water) and has a heading angle (over water) α w.r.t. the x_1-axis. Thus, the velocity components over ground can be expressed as:

$$\begin{aligned} \frac{dx_1}{dt} &= V\cos\alpha + w_1(x_1, x_2) \\ \frac{dx_2}{dt} &= V\sin\alpha + w_2(x_1, x_2) \end{aligned} \tag{1}$$

Using the Calculus of Variations, one can show that such a path necessarily obeys the following differential equation, first derived by Zermelo (1931)

$$\frac{d\alpha}{dt} = \sin^2(\alpha)\, w_{2,1} + \sin(\alpha)\cos(\alpha)\, \left(w_{1,1} - w_{2,2}\right) - \cos^2(\alpha)\, w_{1,2} \tag{2}$$

where for the sake of brevity, we write $w_{i,j} = \partial w_i/\partial x_j$.

Equation (2) is known as the **Zermelo differential equation**. Together with (1) it gives the form for time-optimal trajectories as a dynamical system in the 3-dimensional space parameterized by (x_1, x_2, α). We will refer to the initial value problem for this 3-dimensional dynamical system as the ZIVP. This means that given a current vector field $(w_1(x_1, x_2), w_2(x_1, x_2))$, an initial position $(x_1^{(0)}, x_2^{(0)})$ and an initial heading α, the trajectory defined by the initial value problem is guaranteed to be time optimal, i.e. each point in that trajectory cannot be reached in shorter time by a vessel with constant speed over water V starting from $(x_1^{(0)}, x_2^{(0)})$. For the interest of completeness, the derivation of this last equation is fully explained in Appendix A.

2.2 Zermelo's navigation problem on the sphere

We now modify the above equations to the case where the ship is traveling on the surface of the Earth - idealized here as a perfect sphere. We first adopt spherical coordinates $x_1 = \theta$ (longitude) and $x_2 = \phi$ (latitude) measured in units of κ radians. In particular, it may be convenient to take $\kappa = \pi/180$ if we wish to measure in degrees. The velocities of currents will be given relative to a east-north framing, which we represent as the following 2×2 matrix

$$F(\theta, \phi) = \begin{bmatrix} K\cos\theta & 0 \\ 0 & K \end{bmatrix}$$

where K is the conversion scale from the units used to measure θ, ϕ and the units used to measure local velocities. For example, if global position is measured using degrees of arc, and local velocities are measured in kilometres, then letting R be the Earth's radius in kilometres ($R \approx 6367$ km), we have $K = \kappa R = \pi R/180 \approx 111.1$ kilometres per 1 degree of arc.

With these conventions in place, the velocity over ground of a vehicle moving at a speed of V over water is given as

$$\begin{aligned} K\cos(\kappa\phi)\frac{d\theta}{dt} &= V\cos(\kappa\alpha) + w_1(\theta,\phi) \\ K\frac{d\phi}{dt} &= V\sin(\kappa\alpha) + w_2(\theta,\phi) \end{aligned} \tag{3}$$

where α is the vessel's heading measured relative to an East-North framing, $\vec{w}(\theta,\phi) = \langle w_1(\theta,\phi), w_2(\theta,\phi)\rangle$ with w_1 being the component of displacement relative to east, and w_2 the component of displacement relative to north.

Using the Calculus of Variations once again, now on the sphere, one can show that such a path necessarily obeys the following differential equation,

$$\begin{aligned} \kappa K\frac{d\alpha}{dt} = &\begin{bmatrix}\cos(\kappa\alpha) & \sin(\kappa\alpha)\end{bmatrix}\begin{bmatrix}\sec(\kappa\phi)w_{1,1} & w_{1,2}\\ \sec(\kappa\phi)w_{2,1} & w_{2,2}\end{bmatrix}\begin{bmatrix}\sin(\kappa\alpha)\\ -\cos(\kappa\alpha)\end{bmatrix} \\ &-\cos(\kappa\alpha)\tan(\kappa\phi)(V+\cos(\kappa\alpha)w_1+\sin(\kappa\alpha)w_2) \end{aligned} \tag{4}$$

Equation (4) can be considered the analogue of the Zermelo differential equation for motion on a sphere. For the sake of completeness, the derivation of this equation is fully explained in Appendix A.

3 Hybrid search method

Hybrid Search (HS) is the three-step algorithm proposed in this paper for solving the ZNP in either Euclidean or Spherical background. The three steps are (i) exploration, (ii) refinement, and (iii) smoothing. The output of the exploration and refinement phases is a piece-wise optimal trajectory that connects a starting location with a desired destination.

In effect, exploration is a shooting method based on the Zermelo's initial value problem (ZIVP). The exploration algorithm formulates multiple instances of a ZIVP with a given initial position and a cone of directions aimed towards the target. The trajectories are then evolved using RK4 numerical solutions to the Zermelo Differential Equation with dynamic termination conditions. The most obvious termination condition is to select the trajectory that minimizes the distance to the target. In practice, it turns out that a better heuristic is to terminate each trajectory when the difference between the heading angle and the direction to target exceeds a certain pre-set threshold. The algorithm is greedy, in that a single "winner" trajectory is selected from the list of dynamically terminated trajectories. This selection is performed on the basis of minimum distance to target.

The refinement phase is just a second run of the exploration algorithm, but this time the cone of initial directions is more narrow and centered on the winner shot angle of the previous exploration phase. The candidate trajectories are then evolved using the same heuristic and a winner is selected based on proximity to the target. Then a new exploration phase begins, starting at the final waypoint of the winner segment. The precise details of the exploration and refinement sub-algorithms are detailed in Sects. 3.1 and 3.2 respectively.

The third phase consists of smoothing the output of the refinement using the FMS algorithm (Ferraro et al. 2021, 2022). This algorithm is a numerical Boundary Value Problem scheme that works by iteratively shifting a given discretized trajectory towards a time-minimizing route. The approach is based on the discrete Calculus of Variation and can, in principle, be utilized with any given Lagrangian. In our case, we select the time-minimizing Lagrangian

such that the corresponding Euler–Lagrange equations are precisely the Zermelo Differential Equation. We then discretize the time-minimizing Lagrangian using a pre-selected time-step and begin the iteration with the piece-wise optimal solution generated by the exploration and refinement sub-algorithms. Because the initial trajectory is piece-wise optimal, the overall effect is that of smoothing the sharp turns present in the initial trajectory and converting the piece-wise smooth and piece-wise optimal solution to a smooth, near optimal solution of the Zermelo problem. The relevant details of the FMS algorithm are specified in Sect. 3.3.

3.1 Exploration step

Given a start point $\boldsymbol{x_A} = (x_{A,1}, x_{A,2})$, and a goal point $\boldsymbol{x_B} = (x_{B,1}, x_{B,2})$, we can first centre a search cone in the direction of $\Lambda_{A,B}$, following Eq. (5) (assuming an euclidean space). The amplitude for the search cone is γ. If the vector field was null and we started a trajectory with heading $\alpha = \Lambda_{A,B}$, the vessel would eventually arrive to $\boldsymbol{x_B}$. Thus, by taking this search cone, we are assuming that the optimal route will always point close to the destination and that the vector field will have a small effect on the vessel trajectory. However, this assumption can be relaxed by increasing the amplitude of the cone, γ (up to 2π, covering all directions).

$$\Lambda_{i,j} = \Lambda(\boldsymbol{x_i}, \boldsymbol{x_j}) = \arctan\left(\frac{x_{j,2} - x_{i,2}}{x_{j,1} - x_{i,1}}\right) \tag{5}$$

Equation (5) defines the angle $\Lambda_{i,j}$ from point $\boldsymbol{x_i}$ to point $\boldsymbol{x_j}$. This equation is applicable in Euclidean space, and can be generalized to spherical geometry for short distances. However this does not hold for our study as distances between start and end points are significant, so when working in spherical space it is better to replace Eq. (5) by the following:

$$\Lambda_{i,j} = \arctan\left(\frac{-c_j \cdot s_i + c_i \cdot s_j}{-\left(c_i \cdot c_j + s_i \cdot s_j\right) \cdot \sin(x_{i,2}) + \left(c_i^2 + s_i^2\right) \cdot \sin(x_{j,2})}\right) \tag{6}$$

where $c = \cos(x_1) \cdot \cos(x_2)$; and $s = \sin(x_1) \cdot \cos(x_2)$.

Next, we generate N initial shooting angles, namely

$$\alpha_n(0) \in \left[\Lambda_{A,B} - \gamma/2,\ \Lambda_{A,B} + \gamma/2\right].$$

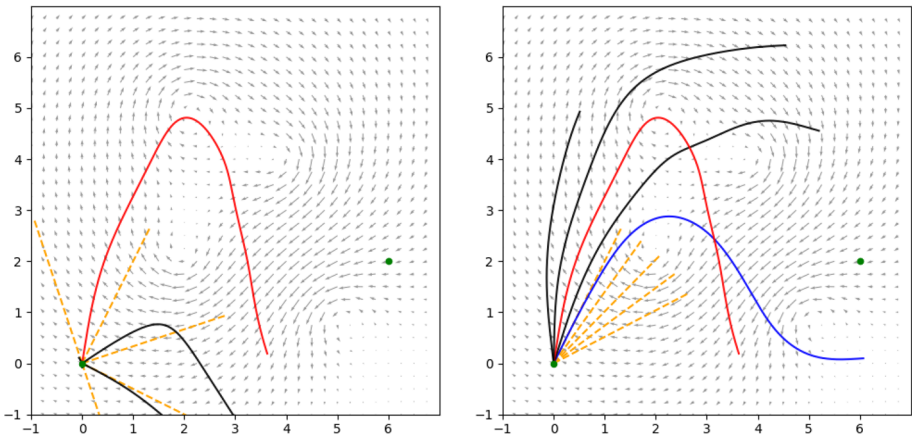
To do so we N-sect the search cone into $\alpha_0, \ldots, \alpha_N$, evenly spread across the whole cone, and use each of these α_n as an initial condition to solve the system of ODE via the Fourth order Runge–Kutta method (RK4). We will use these shooting angles to generate N local paths, or trajectories $q_n(t) = (x_{n,1}(t), x_{n,2}(t), \alpha_n(t)),\ n \in [0, N]$.

The N generated trajectories evolve using RK4, in iterations of time $\tau > \Delta t$ (where Δt is the time step of RK4). After every iteration τ, each trajectory n is checked individually to assert whether it meets any one of three stopping conditions. If it does, trajectory n is left out of the RK4 loop and will not evolve further. These three rules are:

1. Trajectory n is stopped at time T if

$$D\left(\boldsymbol{x_n}(T), \boldsymbol{x_B}\right) \leq d,$$

   being $D(\boldsymbol{x_a}, \boldsymbol{x_b})$ the distance metric between two points, defined according to the space we are operating on, and d a certain distance threshold. This implies the vessel has reached its goal.
2. Trajectory n is stopped at time T if its heading $\alpha_n(T)$ deviates too much from the goal. To assert this, we take point $\boldsymbol{x_n}(T)$, and compute its angle to $\boldsymbol{x_B}$, named $\Lambda(\boldsymbol{x_n}(T), \boldsymbol{x_B})$, see



(a) (b)

Fig. 1 First two steps of the Hybrid Search (HS) method: (**a**) exploration and (**b**) refinement. Each trajectory is generated from a different shooting angle (in orange) and evolves using Fourth order Runge–Kutta method (RK4) method iteratively with $\tau = 0.1$, until their heading deviates more than $\gamma_d = \pi/2$ radians from the goal. After all local paths are computed, the one that got closer to the destination is chosen as best (highlighted in the graph). The search cone had an amplitude of $\gamma = \pi$ radians in the exploration step and was centred on the direction of the goal. During refinement, the search cone was centred on the shooting angle of the best route found in the exploration step, and its amplitude is narrower, $\gamma = \pi/5$

Eqs. (5) and (6). Otherwise, the trajectory keeps evolving while the following condition is met:

$$(\Lambda(\boldsymbol{x_n}(T), \boldsymbol{x_B}) - \gamma_d/2) \leq \alpha_n(T) \leq (\Lambda(\boldsymbol{x_n}(T), \boldsymbol{x_B}) + \gamma_d/2)\,,$$

where γ_d is the maximum deviation allowed from the goal, typically equal or lower than the search cone $\gamma_d \leq \gamma$. The higher γ_d, the more exploratory is this method, but it will take more iterations to converge.

3. Trajectory n is stopped at time T if any of its points $\boldsymbol{x_n}(t), t \in [0, T]$ are located in land. In addition to stopping the trajectory, the algorithm discards all the way-points $q_n(t),\ t \geq t_{\text{land}}$, being $\boldsymbol{x_n}(t_{\text{land}})$ the first point located in land. The trajectory $q_n(t),\ t < t_{\text{land}}$ is kept, as it may still be the optimal route and just needs a course correction, that will be done in a later step.

Note that each trajectory may be stopped at a different time T. For this reason, we denote as T_n the last moment of trajectory n, i.e. its waypoints are $q_n(t),\ t \in [0, T_n]$.

One can argue that the second rule is too strict for small γ_d, as the vessel can be heading "wrongly" for a negligible amount of time before turning "correctly" again, and that the resulting route might be optimal. However, when working with real scenarios, the influence of the vector field is small enough to justify that a vessel going in a "wrong" direction will not turn "correctly" on time to compensate this deviation.

Figure 1a shows a visualization of this exploration step, highlighting the one which got closest to the goal. RK4 method ensures that all trajectories are time optimal. After all N trajectories stop, if none of them reached the goal $\boldsymbol{x_B}$ (i.e. none met the first stopping rule), we choose the trajectory

$$m : D\left(\boldsymbol{x_m}(T_m), \boldsymbol{x_B}\right) \leq D\left(\boldsymbol{x_n}(T_n), \boldsymbol{x_B}\right) \forall n \in [0, N] \tag{7}$$

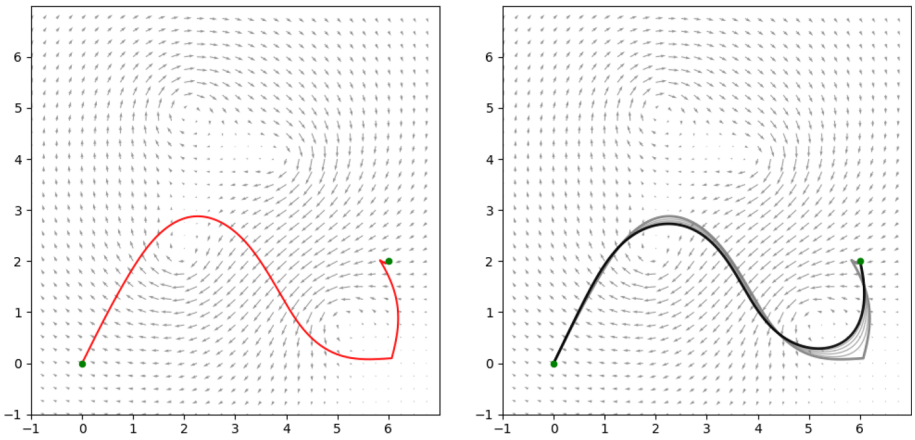


Fig. 2 (**a**) Optimized route obtained by alternating the first two steps of HS method. The segments are locally optimal (thanks to RK4) but are joined by sharp turns. (**b**) The whole route is then smoothed with FMS method for 10 000 iterations

We denote this trajectory m as the "best trajectory", then move to the refinement step (Sect. 3.2).

3.2 Refinement step

In the exploration step, we assumed that the optimal route should be heading closely towards the goal $\boldsymbol{x_B}$, and evolved trajectories defined by the points

$$q_n(t),\ t \in [0, T_n],\ n \in [0, N].$$

with initial shooting angles $\alpha_n(0) \in \left[\Lambda_{A,B} - \gamma/2,\ \Lambda_{A,B} + \gamma/2\right]$.

We now generate a narrower search cone, with amplitude $\gamma_b \ll \gamma$ (for instance, $\gamma_b = \gamma/5$) and we center it on α_m, where m is the "best trajectory" from the exploration step. Thus, the newly generated initial shooting angles are evenly spread across

$$\alpha_n(0) \in \left[\alpha_m - \gamma_b/2,\ \alpha_m + \gamma_b/2\right],\ n \in [0, N]$$

We now re-run the exploration algorithm for this last segment. At some point, all the trajectories will have stopped due to the three rules we set. If no trajectory reached the goal $\boldsymbol{x_B}$ (i.e. no trajectory meets the first stopping criteria), we update the "best trajectory" m following Eq. (7). The algorithm goes back to the exploration step (Sect. 3.1) using $\boldsymbol{x_A} = \boldsymbol{x_m}(T_m)$ as the starting point.

This loop between exploration-refinement continues until the first stopping rule happens, i.e. one trajectory gets close enough to the destination $\boldsymbol{x_B}$. Figure 2a displays one possible result of this process. One issue is apparent: the vessel takes sharp turns in the connections between local paths. This happens because each segment (except the last one) is stopped due to deviating from the goal, so the vessel is forced to correct its course by turning sharply to reach its destination.

3.3 Smoothing step

We see that if we use the paths generated by our approach, it takes sharp turns when two segments connect, which is not realistic for real world situations. Our goal now is to "smooth" the whole route while still optimizing the cost. Following Ferraro et al. (2021), we apply the Newton–Jacobi method to the discretized Euler–Lagrange equation to perform this smoothing. Throughout this paper we will refer to this algorithm as Ferraro–Martín de Diego-Sato algorithm (FMS). Since each segment is already locally optimal, the FMS algorithm can converge to a potential candidate for an optimal solution after a relatively small number of iterations.

Let us quickly review the Newton–Jacobi iterative algorithm for solving nonlinear equation. Consider an equation of the form $0 = f(x)$ where $f(x)$ is a differentiable function of one variable. Newton's method proposes that we pick an approximate solution $x = x^{(0)}$ and then solve the linearized system

$$f(x^{(0)}) + f'(x^{(0)})(x^{(1)} - x^{(0)}) = 0$$

to obtain an $x^{(1)}$. If $x^{(0)}$ is sufficiently close to a root of $f(x) = 0$, one can show that $|f(x^{(1)})| < |f(x^{(0)})|$ and we can iterate to produce a sequence $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ by solving, at each stage the linearized system

$$f(x^{(i)}) + f'(x^{(i)})(x^{(i+1)} - x^{(i)}) = 0.$$

The Newton–Jacobi method generalizes Newton's method to the case of an $n \times n$ system of nonlinear equations $F(q) = 0$ where q is a point in n-dimensional space and F is a transformation of n-dimensional space; i.e., $F(q) = (F_1(q), \ldots, F_n(q))$ is an n-vector of functions. As above, we begin with an initial guess q_0 and then construct a sequence of approximate solutions by solving the linearized equations

$$F(q_i) + DF(q_i)(q_{i+1} - q_i) = 0$$

for q_{i+1}. Under suitable assumptions, one can show that the sequence $q_0, q_1, q_2, \ldots$ converges to a zero of F.

The key idea introduced in Ferraro et al. (2021) is to apply the NJ method iteratively to primitive 3-point trajectories, i.e. trajectory trajectories path consisting of q_{k-1}, q_k, q_{k+1}. For each such trajectory we freeze q_{k-1}, q_{k+1} and seek for the optimal placement of q_k. This amounts to a solution of the discrete Euler–Lagrange equation

$$D_2L_d(q_{k-1}, \bar{q}_k) + D_1L_d(\bar{q}_k, q_{k+1}) = 0$$

for an unknown $\bar{q}_k$. The complete derivation of the system of discrete Euler–Lagrange equations is included in Appendix B.

We now apply the NJ method by taking

$$F(q) = D_2L_d(q_{k-1}, q) + D_1L_d(q, q_{k+1})$$

and apply one iteration of the method to solve the linearized system

$$F(q_k) + DF(q_k)(q_k^* - q_k) = 0$$

for the unknown q_k^*. Fully written, the system for q_k^* is then

$$D_2L_d(q_{k-1}, q_k) + D_1L_d(q_k, q_{k+1}) + (D_{22}(q_{k-1}, q_k) + D_{11}L_d(q_k, q_{k+1}))\,(q_k^* - q_k) = 0$$

We now apply the same one-step iteration to all the primitive trajectories

$$(q_{k-1}, q_k, q_{k+1}),\ k = 1, \ldots, N-1$$

to obtain a new trajectory $q^* = (q_k^*)_{k=0}^N$ with $q_0^* = q_0$ and $q_N^* = q_N$. If the initial trajectory $q^{(0)}$ is well chosen, then the iterated sequence of trajectories $q^{(i)}$, $i = 0, 1, \ldots$ where $q^{(i+1)} = q^{(i)*}$ converges to a solution of the discretized Euler–Lagrange equations. Moreover, this solution is ensured to be time optimal, since the conditions based on the positivity of the Hessian matrix hold generically for the Lagrangian of the Zermelo problem (see Theorems 10 and 14 in Ferraro et al. 2022). By locally time optimal, we mean that any neighboring trajectory will employ a larger time to reach the target.

In the original Zermelo problem seen in Sect. 2.1, we deal with a constrained optimization problem whose Lagrangian function has the form

$$L = \dot{t} + \lambda_1(\dot{x}_1 - (V\cos\alpha + w_1)\dot{t}) + \lambda_2(\dot{x}_2 - (V\sin\alpha + w_2)\dot{t}) \tag{8}$$

A full explanation of the derivation of the above Lagrangian is given in Appendix A. The constraints associated to that Lagrangian are

$$\begin{aligned} \dot{x}_1 &= (V\cos\alpha + w_1)\dot{t} \\ \dot{x}_2 &= (V\sin\alpha + w_2)\dot{t} \end{aligned} \tag{9}$$

We will apply the FMS algorithm to the Euclidean Zermelo problem after suitably transforming (8) into a non-constrained optimization problem. It is possible to extend the FMS methodology to spherical backgrounds and to constrained optimization, but we do not pursue these directions in the present paper.

We begin by combining (9) into the single constraint

$$(\dot{x}_1 - w_1\dot{t})^2 + (\dot{x}_2 - w_2\dot{t})^2 = V^2\dot{t}^2. \tag{10}$$

Setting

$$\begin{aligned} X &= \sqrt{\dot{x}_1^2 + \dot{x}_2^2} \\ W &= \sqrt{w_1^2 + w_2^2} \end{aligned}$$

we rewrite (10) as the following quadratic equation in $\dot{t}$:

$$(V^2 - W^2)\dot{t}^2 + 2XW\cos\beta - X^2 = 0,$$

where β is the angle between $\dot{x}$ and w. The solution gives us the following unconstrained Lagrangian:

$$\hat{L} = \dot{t} = \frac{X}{V^2 - W^2}\left(-W\cos\beta + \sqrt{V^2 - W^2\sin^2\beta}\right)$$

As given, the above $\hat{L}$ is not a regular Lagrangian, and the corresponding $\hat{L}_d$ will not give a convergent FMS algorithm. This difficulty can be remedied by observing that $\hat{L}^2$ **is** regular, and so we take $\hat{L}_d^2$ as the discrete Lagrangian for our implementation of the FMS algorithm.

Figure 2b shows the results of FMS after 10 000 iterations, applied to the route generated at the end of the exploration and refinement loop.

4 Benchmarks

In order to test our optimization approach, we must define a set of benchmarks, containing as many different scenarios as possible. For instance, presence of land between the two ports, strong opposing currents, or highly variable vector fields. Different scenarios can be easily simulated by the use of synthetic benchmarks, which we will comment later. We want, however, to include some real scenarios within our benchmarks. We do not know the best possible route for all benchmarks, but a good point of comparison is the circumnavigation, i.e. the route of minimum distance, also named geodesic when no land is present between the two points. In most realistic scenarios, the optimal routes are found around the geodesic, specially for higher vessel speeds.

4.1 Synthetic benchmarks

It is important to test algorithms first on synthetic benchmarks for several reasons. For starters, synthetic benchmarks provide a controlled and consistent environment for testing, allowing for more accurate and reliable results. This is particularly useful when evaluating the performance of an algorithm under different conditions, as synthetic benchmarks can be easily manipulated to simulate a wide range of scenarios. Secondly, synthetic benchmarks allow for the testing of algorithms without the need for real-world data, which can be expensive and time-consuming to obtain. This allows for faster and more cost-effective testing and evaluation of algorithms. Third, synthetic benchmarks can be used to test the robustness and reliability of algorithms. By introducing challenges and variations to the synthetic benchmark, it is possible to assess how well an algorithm can handle different situations and environments. This can provide valuable insights into the limitations and potential improvements of the algorithm.

4.1.1 Circular vector field

A very simple benchmark we can define is a circular vector field, centred in (a, b). The currents spin clock-wise and have an increasing intensity (defined by s) the further one strays from the centre. This is summarized in the following equation:

$$W(x_1, x_2) = \langle s \cdot (x_2 - b), -s \cdot (x_1 - a) \rangle$$

In this work, we are using $(a, b) = (-3, -1)$ and a scale factor $s = 0.05$, small so that a vessel with unitary velocity can overcome currents near the centre. When testing optimizers we will ask them to develop a path traversing the centre: from $\boldsymbol{x_A} = (3, 2)$ to $\boldsymbol{x_B} = (-7, 2)$. Algorithms are expected to follow the direction of favourable currents.

4.1.2 Four vortices

The next synthetic benchmark we will test is the one appearing in Ferraro et al. (2021), called "four vortices". This vector field is defined by the following equation:

$$W(x_1, x_2) = s \cdot \left(-R_{2,2} - R_{4,4} - R_{2,5} + R_{5,1}\right),$$

where each vortex is expressed as

$$R_{a,b}(x_1, x_2) = \frac{1}{3\left((x_1 - a)^2 + (x_2 - b)^2\right) + 1} \begin{bmatrix} -(x_2 - b) \\ x_1 - a \end{bmatrix}.$$

The authors explained that the scale factor $s = 1.7$ is chosen so that the maximum value of $|W|$ is almost 1. As we are testing these synthetic benchmarks using vessel with unitary velocities, we respect this factor. When testing optimizers we will ask them to develop a path from $\boldsymbol{x_A} = (0, 0)$ to $\boldsymbol{x_B} = (6, 2)$.

4.2 Real benchmarks

Oceanographic data for real case scenarios was downloaded from Copernicus Marine Environment Monitoring Service (Copernicus 2019). Copernicus offers APIs and a Python client to facilitate and automate data downloads, which are typically stored in NetCDF format.

As a first example, we consider a journey from Charleston (32.7°N 79.7°W) to the Azores islands (38.5°N 29.5°W) during spring, using data specifically from May 25th, 2022. This represents the simplest real-world scenario as the trajectory is largely over open ocean, and ocean currents are relatively calm in this region of the Atlantic. However, a favourable current flows northeastward near the departure point (see Fig. 3a), which could be exploited by our algorithm to save time.

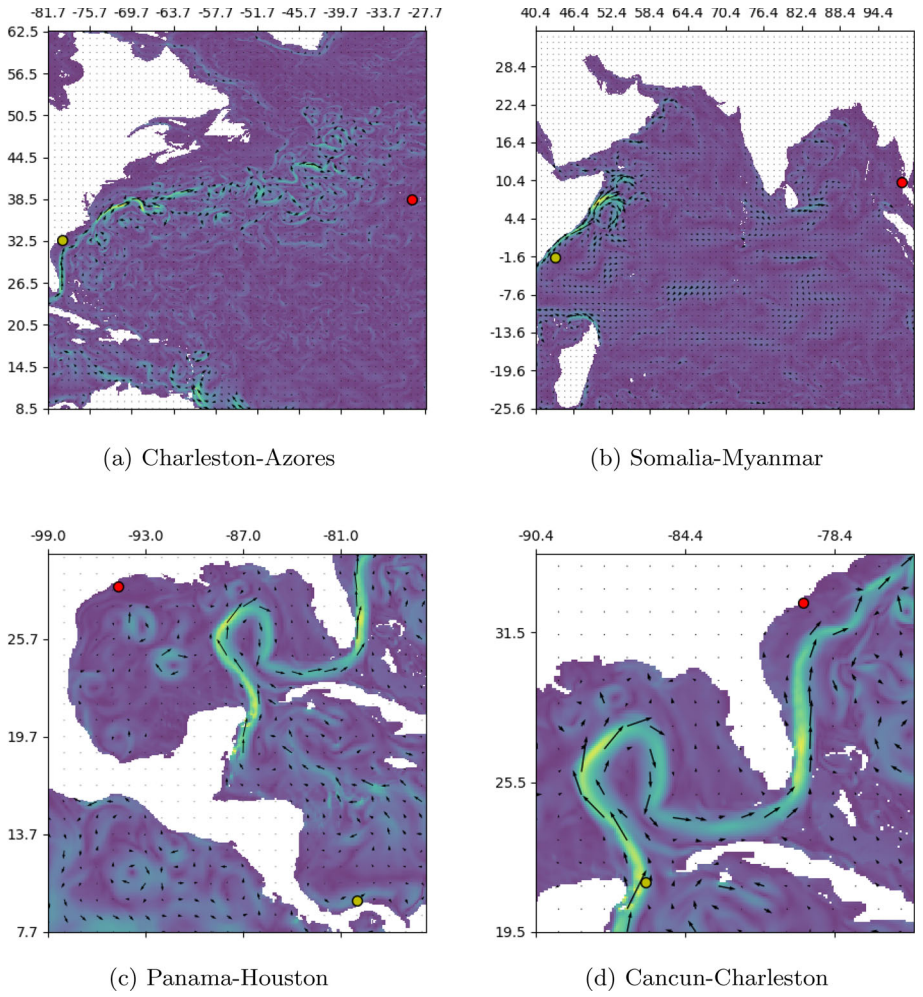
The second example is a journey from Somalia (1.66°S, 42.39°E) to Myanmar (10.21°N, 98.14°E), traversing the Indian Ocean during summer, with data specifically obtained from the 1st of July 2022. This scenario presents a greater challenge than the first example, as there are several islands along the way that the vessel must avoid in order to reach its destination safely, as shown in Fig. 3b. Our goal in using this benchmark is to test the algorithm's ability to avoid land while also making use of the vector field.

The third example involves a journey from Panama (9.7°N, 80.0°W) to Houston (29.0°N, 94.7°W), traversing the Caribbean and the Gulf of Mexico. The data used for this example is the same as in the first example, from the 25th of May 2022. In this case, we are testing the algorithm's ability to avoid large land masses, while also utilizing the Gulf Stream, which is clearly visible in Fig. 3c.

The final example is a journey from Cancun (21.5°N, 86.0°W) to Charleston (32.7°N, 79.7°W), once again traversing the Gulf of Mexico and reaching the Atlantic Ocean. We again use the same data as in the previous example, from the 25th of May 2022. This scenario presents the greatest challenge of all, as the vessel must navigate through a narrow waterway connecting two large land masses (Florida and Cuba), as shown in Fig. 3d. In this case, we will test which parameters work best for the algorithm to avoid large land masses.

5 Results

We run the Hybrid Search (HS) algorithm in all the benchmarks mentioned in Sect. 4, aiming to find the route that takes the least amount of time. The results of the algorithm are shown twice: first after applying HS to solve the Zermelo's initial value problem (ZIVP), and then after smoothing the previous result with the FMS algorithm. To have a point of comparison, we also show the time elapsed by the route of minimum distance. For the synthetic vector fields, operating in Euclidean geometry, the minimum distance is the straight line. For the real vector fields, the minimum distance is the geodesic in the absence of land, otherwise it is called the circumnavigation route.



(a) Charleston-Azores (b) Somalia-Myanmar

(c) Panama-Houston (d) Cancun-Charleston

Fig. 3 Real vector fields. Yellow point marks the starting position and red is the goal. The ocean currents are coloured by intensity, the fastest being represented with brighter (greener) colours

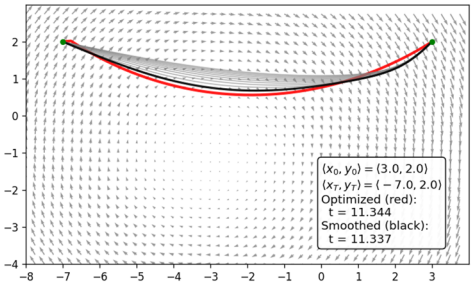
5.1 Synthetic benchmarks

To solve the synthetic vector fields for a vessel of unitary velocity on Euclidean geometry, Hybrid Search (HS) was run using a time step of $\Delta t = 0.01$ and checking the stopping criteria every $\tau = 0.1$. There were twenty one trajectories being tested by the HS, their initial shootings evenly spread across a cone of amplitude $\gamma = \pi$ centered on the direction to the goal. The trajectories would stop if their heading deviated more than $\gamma_d = \pi/2$, or when they got close to the goal (at least $d = 0.1$). Once the HS guessed and optimal route, it would be smoothed by the FMS algorithm during 10 000 iterations.

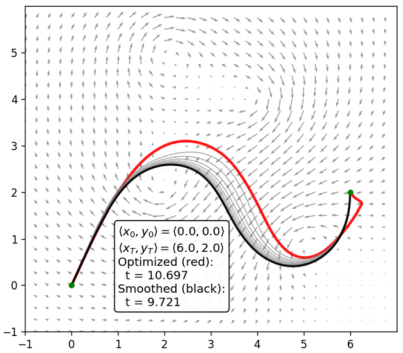
The travel times obtained by our method in the two synthetic benchmarks are shown in Table 2. In addition we include the optimized route for the four vortices vector field in Fig. 4b, which coincidentally is the best route found by the original designers of the benchmark

Table 2 Results on synthetic vector fields with unitary velocity, comparing the route of minimum distance with the output from Hybrid Search (HS) method

Vector field	Method	Time
Circular	Min. dist	11.93
	HS	10.56
Four Vortices	Min. dist	30.44
	HS	9.72



(a) Circular



(b) Four vortices

Fig. 4 Results on the synthetic vector fields, sailing at unit speed

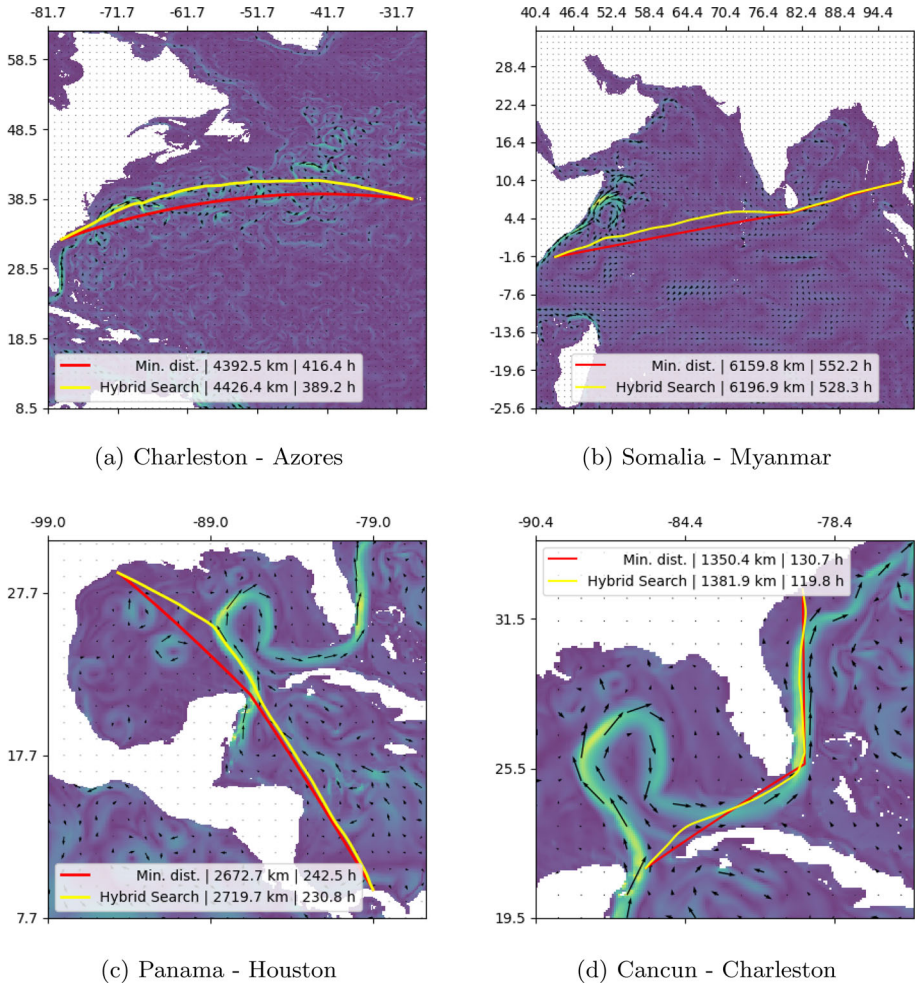
(Ferraro et al. 2021), proving that the HS method is well-suited to give initial guesses that can be smoothed with FMS.

In both benchmarks, the HS method is able to find a route that takes less time than traveling the minimum distance. Improvements are great in the four vortices vector field, due to the currents having a velocity close to the vessel and for that reason having a bigger effect. The circular vector field has currents with lower speeds, and thus the optimized route is not that different from the route of minimum distance.

5.2 Real benchmarks

To solve the real vector fields for a vessel of different velocities on spherical geometry (approximating the Earth's radius as 6367.449 km), Hybrid Search (HS) was run using a time step of $\Delta t = 600$ s (10 min) and checking the stopping criteria every $\tau = 7200$ s (2 h). There were twenty one trajectories being tested by the HS, their initial shootings evenly spread across a cone of amplitude $\gamma = \pi$ (180°) centred on the direction to the goal. The trajectories would stop if their heading deviated more than $\gamma_d = \pi/2$ (90°), or when they got close to the goal (at least $d = 10$ km). Once the HS guessed and optimal route, it would be smoothed by Ferraro–Martín de Diego-Sato algorithm (FMS) during 2000 iterations.

Figure 5 shows how the optimization method meets our expectations: the route deviates to the north and follow a strong current. This makes the vessel cover more distance but in turn reduces the amount of travel time. The effect of this current is relevant enough even for the fastest vessel speeds.



(a) Charleston - Azores

(b) Somalia - Myanmar

(c) Panama - Houston

(d) Cancun - Charleston

Fig. 5 Results on the real vector fields, sailing at 3 m/s

6 Summary and discussion

This paper introduced a new method to solve the Zermelo's initial value problem (ZIVP), named the Hybrid Search (HS) method. The HS method employs a technique of shooting different guesses centred around the general direction towards the goal and simulates the effect of the vector field on these trajectories by applying the fourth-order Runge–Kutta method (RK4). By alternating between an exploratory phase and a refinement step, and following some stop rules to eliminate sub-optimal guesses, the HS method outputs a chain of trajectories that connect the departure and goal points. While each segment is locally optimal since it obeys the Zermelo's system of ODEs, the concatenated full trajectory clearly isn't. For this reason, in the smoothing step we use the concatenated trajectory as seed input for the Ferraro–Martín de Diego-Sato algorithm (FMS), which converges to a locally optimal solution. The advantages of this whole approach are plenty: FMS converges faster to a solution

when given the locally optimal trajectories from RK4, and the sharp turns inherent to the RK4 method are smoothed by FMS.

While the HS method does not guarantee finding the global optimum, its search parameters can be tuned to emphasize its exploratory nature, for instance by opening the search cone or allowing larger deviations from the goal. The more exploration that is allowed, the better local optima can be found, in exchange for increased computation time. Furthermore, the algorithm can be made less greedy by keeping the best N trajectories when moving between exploration and refinement steps, instead of focusing just on the one that gets closest to the goal. This is a future line of work to improve the HS.

The HS algorithm is easily adapted to a spherical space, making it well-suited for its application in real maritime routing problems. Results also show how by fine-tuning the parameters of the algorithm, the HS method finds an optimal route while avoiding land masses. This ability of the HS method can be extrapolated not only to obstacle avoidance but also to circumnavigate dangerous areas or no-sail zones. In addition, by changing the time problem to a realistic consumption model through the use of calculus of variations and modifying the Hamiltonian and initial value problem, our algorithm can be easily adapted to a variety of situations. This makes our approach well-suited for real-world applications in weather routing.

The HS algorithm, while robust, faces challenges such as navigating large land masses and narrow straits. Situations where trajectories must circumnavigate continents or navigate canals may hinder convergence. Enhancing the algorithm's exploratory capabilities and reducing its greediness, as discussed previously, may address these issues. Additionally, introducing intermediate way-points could decompose complex routes into simpler segments, improving convergence. All of these cases will be further explored in future work, where Hybrid Search will be most likely combined with other methods that are better suited to tackle these issues.

Our results demonstrate that the Hybrid Search (HS) method consistently outperforms the route of minimum distance in terms of travel time and distance for navigating vector fields with complex structures. This is particularly evident in the case of the synthetic vector fields, where the HS method achieves a significant reduction in travel time compared to the reference route. In fact, the travel time reduction achieved by the HS method ranges from 11% to 68% depending on the complexity of the vector field, as shown in Table 2. These results indicate that the HS method is a promising approach for optimizing paths in various types of vector fields.

In addition to the synthetic vector fields, we also tested the HS method on four real-world benchmarks, all of which represent common maritime routes. The results, presented in Table 3, demonstrate that the HS method consistently outperforms the route of minimum distance in terms of travel time for these benchmarks. Specifically, the HS method achieves a travel time reduction of up to 6.5%, reducing fuel consumption up to 12.6%, when we consider that the fuel consumed typically scales quadratically with travel time (Harvald 1992; Bialystocki and Konovessis 2016). Although the reduction achieved in these real-world benchmarks is relatively small compared to the synthetic benchmarks, the HS method still shows its potential to improve the efficiency of sailing in real-world scenarios.

In conclusion, the proposed HS method provides a powerful tool for solving the ZIVP and has shown promising results in real-world applications. The ability to fine-tune the parameters of the algorithm enables it to explore a range of possibilities, providing a flexible and adaptive approach to routing problem-solving.

Table 3 Results on the real benchmarks, comparing the route of minimum distance (circumnavigation) with the output from Hybrid Search (HS) method

Benchmark	Speed (m/s)	Method	Travel time (h)	Distance (km)
Charleston Azores	3	Min. dist	416.4	4392.5
		HS	389.2	4426.4
	6	Min. dist	207.9	4392.5
		HS	202.0	4423.3
	10	Min. dist	124.9	4392.5
		HS	123.3	4418.3
Somalia Myanmar	3	Min. dist	552.2	6159.8
		HS	528.3	6196.9
	6	Min. dist	280.0	6159.8
		HS	274.8	6190.4
	10	Min. dist	169.3	6159.8
		HS	167.8	6190.4
Panama Houston	3	Min. dist	242.5	2672.7
		HS	230.8	2719.7
	6	Min. dist	124.0	2672.7
		HS	120.9	2700.8
	10	Min. dist	74.2	2672.7
		HS	74.0	2716.6
Cancun Charleston	3	Min. dist	130.7	1350.4
		HS	119.8	1381.9
	6	Min. dist	70.0	1350.4
		HS	65.6	1354.7
	10	Min. dist	42.9	1350.4
		HS	40.8	1365.8

A Derivation of Zermelo's equations

A.1 Zermelo's navigation problem on the plane

We are dealing here with a constrained optimization problem whose Lagrangian function has the form

$$L = \dot{t} + \lambda_1(\dot{x}_1 - (V\cos\alpha + w_1)\dot{t}) + \lambda_2(\dot{x}_2 - (V\sin\alpha + w_2)\dot{t}). \tag{11}$$

The goal is to find trajectories $x(s),\ \dot{x}(s) = x'(s),\ t(s),\ \dot{t}(s) = t'(s) > 0,\ \alpha(s)$ with fixed end-points that minimize $t(s_1) - t(s_0) = \int_{s_0}^{s_1} L ds$, and obey constraints

$$\begin{aligned} \dot{x}_1 &= (V\cos\alpha + w_1)\dot{t} \\ \dot{x}_2 &= (V\sin\alpha + w_2)\dot{t} \end{aligned} \tag{12}$$

The quantities λ_1, λ_2 are known as Lagrange multipliers. As we now show, their form is determined by the Euler–Lagrange equations associated with the above Lagrangian, namely

$$\frac{dL_{\dot{t}}}{ds} = 0 \tag{13}$$

$$L_{x_i} - \frac{dL_{\dot{x}_i}}{ds} = 0 \quad i = 1, 2 \tag{14}$$

$$L_\alpha = 0, \tag{15}$$

Equation (13) gives

$$\frac{d}{ds}\left(\lambda_1(V\cos\alpha + w_1) + \lambda_2(V\sin\alpha + w_2)\right) = 0$$

which implies that

$$\lambda_1(V\cos\alpha + w_1) + \lambda_2(V\sin\alpha + w_2) = C \tag{16}$$

where $C \neq 0$ is a constant. Equation (15) gives

$$\lambda_1 \sin\alpha - \lambda_2\cos\alpha = 0. \tag{17}$$

Together, (16) (17) determine the form of the Lagrange multipliers, namely:

$$\lambda_1 = \frac{C\cos\alpha}{V + w_1\cos\alpha + w_2\sin\alpha} \tag{18}$$

$$\lambda_2 = \frac{C\sin\alpha}{V + w_1\cos\alpha + w_2\sin\alpha} \tag{19}$$

Going forward, we re-parameterize all curves with respect to time t so that

$$\frac{d}{dt} = \frac{1}{\dot{t}}\frac{d}{ds}.$$

E–L Eq. (14) give the dynamics of the Lagrange multipliers, namely

$$\frac{d\lambda_1}{dt} = -\lambda_1 w_{1,1} - \lambda_2 w_{2,1} \tag{20}$$

$$\frac{d\lambda_2}{dt} = -\lambda_1 w_{1,2} - \lambda_2 w_{2,2} \tag{21}$$

Rewriting (17) as

$$\tan\alpha = \frac{\lambda_2}{\lambda_1},$$

and taking derivatives, gives

$$\begin{aligned}
\sec^2(\alpha)\frac{d\alpha}{dt} &= \frac{d}{dt}\left(\frac{\lambda_2}{\lambda_1}\right)\\
\left(\frac{\lambda_1^2+\lambda_2^2}{\lambda_1^2}\right)\frac{d\alpha}{dt} &= \frac{1}{\lambda_1^2}\left(-\lambda_2\frac{d\lambda_1}{dt} + \lambda_1\frac{d\lambda_2}{dt}\right)\\
\frac{d\alpha}{dt} &= \frac{\lambda_2^2 w_{2,1} + \lambda_1\lambda_2(w_{1,1} - w_{2,2}) - \lambda_1^2 w_{1,2}}{\lambda_1^2+\lambda_2^2}\\
\frac{d\alpha}{dt} &= \sin^2(\alpha)\, w_{2,1} + \sin(\alpha)\cos(\alpha)\,(w_{1,1} - w_{2,2}) - \cos^2(\alpha)\, w_{1,2}
\end{aligned} \tag{22}$$

A.2 Zermelo's navigation problem on the sphere

The modified Lagrangian takes the form

$$L = \dot{t} + \lambda_1 \left(\dot{\theta} - K^{-1} \sec(\kappa\phi)\,(V\cos(\kappa\alpha) + w_1)\,\dot{t}\right) + \lambda_2 \left(\dot{\phi} - K^{-1}\,(V\sin(\kappa\alpha) + w_2)\,\dot{t}\right)$$

The E–L Eq. (14) now read

$$K\frac{d\lambda_1}{dt} = -\sec(\kappa\phi)\lambda_1 w_{1,1} - \lambda_2 w_{2,1} \tag{23}$$

$$K\frac{d\lambda_2}{dt} = -\lambda_1\kappa\sec(\kappa\phi)\tan(\kappa\phi)(V\cos(\kappa\alpha) + w_1) - \lambda_1\sec(\kappa\phi)w_{1,2} - \lambda_2 w_{2,2} \tag{24}$$

In the current setting (15) gives

$$\tan(\kappa\alpha) = \frac{\lambda_2}{\lambda_1}\cos(\kappa\phi)$$

Taking d/dt yields

$$\begin{aligned}
\kappa\sec^2(\kappa\alpha)\frac{d\alpha}{dt} &= \frac{\cos(\kappa\phi)}{\lambda_1^2}\left(-\lambda_2\frac{d\lambda_1}{dt} + \lambda_1\frac{d\lambda_2}{dt}\right) \\
&\quad - \frac{\kappa}{K}\tan(\kappa\alpha)\tan(\kappa\phi)(V\sin(\kappa\alpha) + w_2) \\
\kappa K\sec^2(\kappa\alpha)\frac{d\alpha}{dt} &= \frac{\lambda_2}{\lambda_1}w_{1,1} + \frac{\lambda_2^2}{\lambda_1^2}\cos(\kappa\phi)w_{2,1} - w_{1,2} - \frac{\lambda_2}{\lambda_1}\cos(\kappa\phi)w_{2,2} \\
&\quad - \kappa\tan(\kappa\phi)(V\cos(\kappa\alpha) + w_1) \\
&\quad - \kappa\tan(\kappa\alpha)\tan(\kappa\phi)(V\sin(\kappa\alpha) + w_2) \\
\kappa K\sec^2(\kappa\alpha)\frac{d\alpha}{dt} &= \sec(\kappa\phi)\tan(\kappa\alpha)w_{1,1} + \sec(\kappa\phi)\tan^2(\kappa\alpha)w_{2,1} - w_{1,2} \\
&\quad - \tan(\kappa\alpha)w_{2,2} - \tan(\kappa\phi)(V\cos(\kappa\alpha) + w_1) \\
&\quad - \kappa\tan(\kappa\alpha)\tan(\kappa\phi)(V\sin(\kappa\alpha) + w_2) \\
\kappa K\frac{d\alpha}{dt} &= \begin{bmatrix}\cos(\kappa\alpha) & \sin(\kappa\alpha)\end{bmatrix}\begin{bmatrix}\sec(\kappa\phi)w_{1,1} & w_{1,2} \\ \sec(\kappa\phi)w_{2,1} & w_{2,2}\end{bmatrix}\begin{bmatrix}\sin(\kappa\alpha) \\ -\cos(\kappa\alpha)\end{bmatrix} \\
&\quad - \cos(\kappa\alpha)\tan(\kappa\phi)(V + \cos(\kappa\alpha)w_1 + \sin(\kappa\alpha)w_2)
\end{aligned} \tag{25}$$

B Euler–Lagrange equations

B.1 Continuous Euler–Lagrange equations

Define an action functional along a curve $q(t)$ in n-dimensional space with fixed end points as follows,

$$J(q(t)) = \int_a^b L(t, q(t), \dot{q}(t))dt, \quad q(a) = \alpha, \quad q(b) = \beta. \tag{26}$$

The function $L(t, q(t), \dot{q}(t))$ is called the Lagrangian of the optimization problem. The classical problem in the Calculus of Variations is to to minimize J by subjecting $q(t)$ to suitable constraints.

A necessary condition for minimization is that the variation δJ vanishes for all possible variations of the trajectory $\delta q = \epsilon\phi$, where $\phi(t)$ vanishes at the endpoints, and ϵ is the variational parameter. From the functional (26), define

$$h(\epsilon) = J(q + \epsilon\phi) = \int_a^b L(t, q(t) + \epsilon\phi(t), \dot{q}(t) + \epsilon\dot{\phi}(t))dt.$$

Now differentiate and use the smoothness of L to interchange the derivative and the integral to get

$$\begin{aligned} h'(\epsilon) = \frac{d}{d\epsilon} J(q + \epsilon\phi) &= \int_a^b \frac{d}{d\epsilon} L\left(t, q(t) + \epsilon\phi(t), \dot{q}(t) + \epsilon\dot{\phi}(t)\right) dt \\ &= \int_a^b \phi(t)\Bigg[\frac{\partial L}{\partial q}\left(t, q(t) + \epsilon\phi(t), \dot{q}(t) + \epsilon\dot{\phi}(t)\right) \\ &\qquad + \dot{\phi}(t)\frac{\partial L}{\partial \dot{q}}\left(t, q(t) + \epsilon\phi(t), \dot{q}(t) + \epsilon\dot{\phi}(t)\right)\Bigg] dt. \end{aligned}$$

Now setting $\epsilon = 0$ and using our definition of the variational derivative yields

$$\delta J(q)(\phi) = \int_a^b \left[\phi(t)\frac{\partial L}{\partial q}(t, q, \dot{q}) + \dot{\phi}(t)\frac{\partial L}{\partial \dot{q}}(t, q, \dot{q})\right] dt. \tag{27}$$

This functional is known as the *first variation* of J. In order to obtain an explicit formula for δJ, we need the integral on the right side of the above equation to be linear in $\phi(t)$. We can accomplish this via integration by parts.

$$\int_a^b \dot{\phi}(t)\frac{\partial L}{\partial \dot{q}}(t, q, \dot{q})dt = \left[\phi(t)\frac{\partial L}{\partial \dot{q}}(t, q(t), \dot{q}(t))\right]_{t=a}^{t=b} - \int_a^b \phi(t)\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}(t, q, \dot{q})\right) dt$$

Since $\phi(b) = \phi(a) = 0$, by assumption, we obtain the following formula for the first variation:

$$\delta J(q)(\phi) = \int_a^b \left[\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}}\right]\phi(t)dt.$$

Therefore, in order for $\delta J(\phi)$ to vanish for all ϕ, the critical trajectory $q(t)$ must satisfy the *Euler–Lagrange equations*

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = 0. \tag{28}$$

B.2 Discrete Euler–Lagrange equations

Now consider two positions: q_0 and q_1, and a time step $h > 0$. We discretize a continuous Lagrangian $L(q, \dot{q})$ by assuming that q_1, q_0 are close together so that $\dot{q}$ can be approximated by $(q_1 - q_0)/h$. This allows us to define the following discrete Lagrangian

$$L_d(q_0, q_1; h) := \frac{h}{2}\left(L\left(q_0, \frac{q_1 - q_0}{h}\right) + L\left(q_1, \frac{q_1 - q_0}{h}\right)\right),$$

which approximates the action integral along a straight trajectory from q_0 to q_1. In the discrete Calculus of Variations, we replace a continuous curve $q(t)$ with a piece-wise linear curve determined by a sequence of points $\{q_k\}_{k=0}^N$ with h units of time required to go from q_k to

q_{k+1}. We will now calculate the discrete action over this sequence by summing the discrete Lagrangian.

$$J_d = \sum_{k=0}^{N-1} L_d(q_k, q_{k+1}; h).$$

We now vary the trajectory by $dq = \{dq_k\}_{k=0}^{N}$ with $dq_0 = dq_N = 0$ in order to fix the boundary points q_0, q_N. Note that we use dq rather than $\epsilon\phi$ to describe the variation because the discretized system has finite degrees of freedom. The variation of the discrete action can now be given as

$$\begin{aligned} dJ_d &= \sum_{j=1}^{N-1} \frac{\partial}{\partial \boldsymbol{x_j}} \left(\sum_{k=0}^{N-1} L_d(q_k, q_{k+1}; h) \right) dq_j \\ &= \sum_{k=0}^{N-1} \left[D_1 L_d(q_k, q_{k+1}; h) dq_k + D_2 L_d(q_k, q_{k+1}; h) dq_{k+1} \right] \end{aligned}$$

Recall that each $\boldsymbol{x_j} = (q_{j1}, \ldots, q_{jn})$ is a point in n-dimensional space, so that $\partial/\partial \boldsymbol{x_j}$, D_1, D_2 are actually n-vectors of partial derivative operators. Rearranging the above sum (this corresponds to the integration by parts step in the continuous case) we obtain

$$dJ_d = \sum_{k=1}^{N-1} \left[D_2 L_d(q_{k-1}, q_k; h) + D_1 L_d(q_k, q_{k+1}; h) \right] dq_k.$$

If we require that the variation of the action is 0 for all dq_k, then we obtain the discrete Euler–Lagrange equations

$$D_2 L_d(q_{k-1}, q_k; h) + D_1 L_d(q_k, q_{k+1}; h) = 0, \quad k = 1, \ldots, N-1. \tag{29}$$

Acknowledgements The research of RM, DP and DGU is supported by the BBVA Foundation via the project "Mathematical optimization for a more efficient, safer and decarbonized maritime transport". This research is part of the project TED2021-129455B-I00 funded by MICIU/AEI/10.13039/501100011033 and by EU in the program "NextGenerationEU/PRTR". We acknowledge also support from the project PID2021-122154NB-I00, funded by MICIU/AEI/10.13039/501100011033 and "ERDF A Way of making Europe". The authors would like to thank the MITACS Accelerate program that enabled the 3 months visit of DP to Dalhousie University in the summer of 2022, where this work was initiated, and the visit of LB to Madrid in the summer of 2023, where the work was completed.

Data availability This paper only uses external data from ocean currents which are publicly available from Copernicus Marine Environment Monitoring Service (Copernicus 2019). Copernicus offers APIs and a Python client to facilitate and automate data downloads, which are typically stored in NetCDF format. In addition, the following GitHub repository contains our source code for all computations and methods reported in this paper: https://github.com/daniprec/hybrid_ivp

Declarations

Conflict of interest The authors have no Conflict of interest to declare that are relevant to the content of this article.

References

Bao D, Robles C, Shen Z (2004) Zermelo navigation on riemannian manifolds. J Differ Geometry 66:377–435

Bialystocki N, Konovessis D (2016) On the estimation of ship's fuel consumption and speed curve: a statistical approach. J Ocean Eng Sci 1:157–166
Copernicus (2019) Global ocean 1/12 physics analysis and forecast updated daily product. eu copernicus marine service information [data set]
Ferraro SJ, Martín de Diego D, Sato Martín de Almagro RT (2022) A parallel iterative method for variational integration. arXiv preprint arXiv:2206.08968
Ferraro SJ, Martín de Diego D, Martín Sato, de Almagro RT (2021) Parallel iterative methods for variational integration applied to navigation problems. IFAC-Pap 54:321–326
Garcelán DP (2023) Applications of machine learning and data science to the blue economy sustainable fishing and weather routing
Grandcolas S (2022) A metaheuristic algorithm for ship weather routing. Oper Res 3:35
Grifoll M, Borén C, Castells-Sanabra M (2022) A comprehensive ship weather routing system using cmems products and a* algorithm. Ocean Eng 255:111427
Harvald SA (1992) Resistance and propulsion of ships (Krieger Pub)
IMO (2020) Fourth imo ghg study 2020—doc. mepc59/inf.10. International Maritime Organization (IMO),London, UK
Kuhlemann S, Tierney K (2020) A genetic algorithm for finding realistic sea routes considering the weather. J Heurist 26:801–825
Lin YH, Fang M-CC, Yeung RW (2013) The optimization of ship weather-routing algorithm based on the composite influence of multi-dynamic elements. Appl Ocean Res 43:184–194
Marchidan A, Bakolas E (2016) Numerical techniques for minimum-time routing on sphere with realistic winds. J Guid Control Dyn 39:188–193
Perera LP, Soares CG (2017) Weather routing and safe ship handling in the future of shipping. Ocean Eng 130:684–695
Roberts SE, Nielsen D, Kotłowski A, Jaremin B (2014) Fatal accidents and injuries among merchant seafarers worldwide. Occup Med 64:259–266
Unctad. Review of maritime transport 2021 (UN, 2021)
Walther L, Rizvanolli A, Wendebourg M, Jahn C (2016) Modeling and optimization algorithms in ship weather routing. Int J e-Navig Maritime Econ 4:31–45
Zermelo E (1930) Über die navigation in der luft als problem der variationsrechnung. Jahresbericht der deutschen Mathematiker-Vereinigung, Angelegenheiten 39:44–48
Zermelo E (1931) Über das navigationsproblem bei ruhender oder veränderlicher windverteilung. ZAMM J Appl Math Mech 11:114–124
Zhao W et al (2022) Multi-objective weather routing algorithm for ships based on hybrid particle swarm optimization. J Ocean Univ China 21:28–38
Zis TP, Psaraftis HN, Ding L (2020) Ship weather routing: A taxonomy and survey. Ocean Eng 213:107697
Zyczkowski M, Szlapczynski R (2023) Collision risk-informed weather routing for sailboats. Reliab Eng Syst Saf 232:109015