Minimum Time Sailing Boat Path Algorithm

David Sidoti, Krishna R. Pattipati, *Fellow, IEEE*, and Yaakov Bar-Shalom, *Fellow, IEEE*

***Abstract*—An iterative procedure to solve the nonlinear problem of fastest-path sailing vessel routing in an environment with variable winds and currents is proposed. In the routing of a sailing vessel, the primary control variable is the pointing (heading) of the vessel (assuming that the sails are chosen and trimmed optimally). Sailing vessel routing is highly nonlinear when considering environmental factors, such as winds and currents, and the behavior of the boat, given the weather conditions (i.e., polar diagrams that predict how fast one can sail, given the vessel's pointing relative to the true wind and the wind speed). The key algorithmic contribution of this article is a fastest-path algorithm for graphs with nonconvex edge costs that depend on weather, current, and boat polars. An illustrative scenario, with idealized weather attributes, and a real-world scenario, with parameters generated by numerical weather and current prediction models, are simulated and tested to compare the proposed algorithm against open-source routing software validated by active sailors. Preliminary results from the simulation setups tested are as follows: 1) the proposed sailing boat path algorithm is comparable to the open-source software available; and 2) exploiting the often unused but significant impactor of surface currents and incorporating leeway into sailing boat path planning enables higher fidelity guidance and faster (i.e., shorter time) routes, in comparing against the freely available baseline.**

***Index Terms*—Dynamic programming, meteorology, oceanography, optimal control, path planning, routing, sailing.**

I. Introduction

COMPETITIVE sailing or yacht racing is a sport involving multiple sailboats divided into different classes, racing over a certain course (outlined by buoys) or over the open water in long distance racing (e.g., the Newport to Bermuda Race [1]). With yacht designs diversified, the capabilities of each boat had to be considered, eventually culminating in the races seen today, where a system of time allowances is now established to take into account the strengths and weaknesses of various boat designs and sails.

Manuscript received 17 May 2022; revised 21 September 2022; accepted 1 December 2022. Date of publication 14 February 2023; date of current version 14 April 2023. The work of D. Sidoti was supported in part by the Chief of Naval Research through the U.S. NRL Base Program under Program Element 0602435N, and in part by the U.S. Office of Naval Research under Program Elements 0602235N, 0602435N, and 0603801N. The work of K. R. Pattipati was supported in part by the U.S. Office of Naval Research under Grant N00014-12-1-0238 and Grant N00014-16-1-2036, and in part by the Naval Research Laboratory under Grant N00173-16-1-G905. The work of Y. Bar-Shalom was supported by his hobby. *(Corresponding author: David Sidoti.)*

Associate Editor: N. Cruz.

David Sidoti is with the Marine Meteorology Division, US Naval Research Laboratory, Monterey, CA 93943 USA (e-mail: david.sidoti@nrlmry.navy.mil).

Krishna R. Pattipati and Yaakov Bar-Shalom are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269 USA (e-mail: krishna.pattipati@uconn.edu; yaakov.bar-shalom@uconn.edu).

Digital Object Identifier 10.1109/JOE.2022.3227985

To gain the competitive edge in such races, many skippers and helmsmen use one or more software packages, web applications, or routing services that utilize the observed and forecasted meteorology and oceanography and accordingly route the sailing vessel to complete the course in the shortest possible time. The problem is well suited for dynamic programming [2], [3], [4], [5], [6] and has more recently been viewed in the context of autonomous robot sailing [7], [8], [9], [10], [11], [12]. In most published approaches, the ocean's currents are either not addressed, deemed negligible, or are vaguely incorporated, but the impact is neither explicitly nor directly integrated into the optimization.

The impact and importance of tidal and ocean currents in fastest-path sailing vessel routing is addressed by Kristensen [13], who concluded that currents should and must be taken into consideration if the problem is to be solved in its entirety. By conducting sensitivity analyses with respect to the use of forecasts of currents, he concluded that there are instances when current can even be more impacting than wind on a sailing vessel. Futch and Allen [14] recognize the impact of currents, specifically the need to account for drift and leeway of objects up to the scale of traditional manned sailboats, on search and rescue operations.

Of the literature pertaining to fastest-path sailing vessel routing, Philpott and Mason [2] are often credited with the first foray into such a problem. In [2], they formulate a stochastic dynamic programming approach that minimizes the time between two points under uncertain weather conditions. They mention the impact of ocean currents; however, they do not incorporate it. They instead define environmental impacts as Markov processes solely a function of wind direction and speed. In a similar vein, Tagliaferri et al. [5] formulate a stochastic shortest path problem assuming tidal currents to be negligible; however, they include the human's perspective of sailing in an attempt to incorporate risk propensity into their optimization. Wave resistances are used in lieu of explicit meteorological feature incorporation in [6]. The wave resistances are calculated via a blackbox model derived from strip theory in [15], where hull resistance calculation is feasible given the knowledge of the wave frequency, length, and amplitude, as well as hull characteristics of the ship. The blackbox component outputs dynamic shear forces assuming a ship to be advancing at a constant mean speed for a given heading in regular sinusoidal waves. This approach increases the fidelity of typical path optimization because of its consideration of wave height/period/direction; however, surface currents are not integrated, including corresponding impacts to leeway.

In the autonomous robot sailing literature, current is often neglected entirely with the exception of [10], [11], and [12].

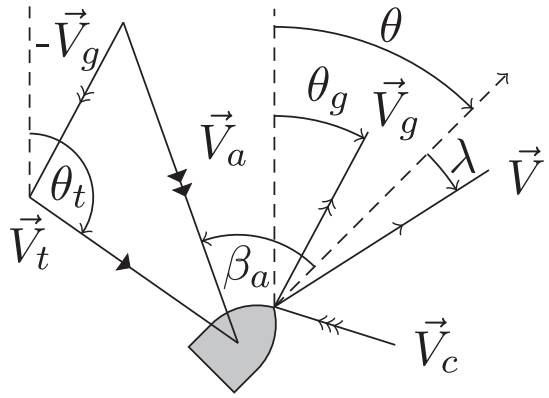

Fig. 1. Free body diagram corresponding to a sailing vessel, while considering wind and current vectors.

Wrede et al. [10] use velocity made good (VMG) and heel angle to optimize objectives pertaining to a four-degree-of-freedom robot sailing model. They utilize a hill-climbing algorithm that takes into account the measurements of drift from currents, but do not use the current forecast information explicitly. The work [11] addresses current in their optimization, but not leeway. In addition, they do not incorporate current on the inner level of their optimization when selecting a heading. Akiyama et al. [12] present one of the most recent approaches to autonomous robot sailing and explicitly recognize surface currents as having *significant* impact on the robot's routes. The robot used in their experiments had a hull and structure similar to that of a traditional human-operated sailboat and observed ocean conditions impacting the trajectory taken by the vessel. However, the path planning optimization did not directly incorporate surface currents, and the deviation from the trajectory was allowed under the condition that the autonomous sailboat remained under control.

In this article, we formulate the fastest-path sailing vessel routing problem as a finite-horizon shortest path problem with nonconvex arc costs that incorporate the weather, current, and boat polars (with the latter's expanded table form that includes leeway).[1] We extend previous approaches by explicitly including the current speed and direction information and propose an iterative procedure (herein, referred to as the sailing boat path (SBP) algorithm) that outputs the pointing of the vessel (heading[2]) at each stage and state in the optimization. We solve the single-source (starting point) single-sink (destination) shortest path algorithm by decomposing the route into a trellis, while adhering to constraints, such as the allowable bearings (headings) at each waypoint (heading too close to the wind is not allowed according to the polars).

The rest of this article is organized as follows. In Section II, we formulate the fastest-path sailing vessel problem and decompose it into two subproblems: that of the dynamic programming equations governing the optimal arc selection, and that of the calculation of the arc costs. In Section III, we detail our solution approach and proposed an iterative procedure step by step. Section IV contains the simulation setup proposed to validate the algorithm in the presence of both current and wind vectors (including a real-world scenario—Newport Bermuda race), and we discuss our findings in Section V. Finally, Section VI concludes this article. Appendix A provides the details of the SBP algorithm and the performance tables it relies upon. Appendix B shows some additional examples for SBP. Appendix C discusses the dynamic programming optimization.

II. Problem Formulation

For completeness, the formulation is divided into two sections to decompose the general fastest-path sailing vessel routing problem into that of a shortest path problem formulation, where we detail the general cost function to be minimized given specified constraints, and a sailing vessel formulation, where we detail the environmental impact on the vessel, and the relations among the angles and vectors of interest used in the cost calculation.

A. Minimum Time Path Formulation With Dynamic Programming

Let $G = (N, E)$ be a directed acyclic graph, comprising a *traversable* set of nodes N and a set of edges E. Given a start location and a destination, a sailing vessel must adhere to the graph G and traverse along edges $e \in E$, while visiting waypoints (nodes) $n \in N$. It is assumed we have perfect knowledge of the bathymetry (water depth) pertaining to the area spanned by G, and given the sailing vessel draft, the bathymetry is greater than or equal to a specified depth $\forall n \in N$, $\forall e \in E$ such that safe traversal is guaranteed. The graph G is structured as a trellis such that there are one or more groups of vertically aligned nodes across the same horizontal[3] position. The location and spacing of waypoints is dually a function of the granularity of bathymetric and/or weather/current[4] data sources and user specification. Let each vertical set of nodes be a stage $s = 0, 1, \ldots, S-1$, where S is the total number of stages to get from the departure point (denoted as 0) to the specified destination (denoted as $S-1$). In this manner, the graph G (illustrated later in Fig. 2) may be constructed where all nodes in N correspond to waypoints, represented by geographic coordinates, i.e., latitude ϕ and longitude ψ

$$\mathbf{x}_j(s) = [\phi_j(s)\ \psi_j(s)]^T \tag{1}$$

$$\mathbf{x}(s) = \{\mathbf{x}_0(s)\ \mathbf{x}_1(s) \ldots \mathbf{x}_j(s) \ldots \mathbf{x}_{n_s}(s)\} \tag{2}$$

for $s = 0, \ldots, S-1$; $j = 1, \ldots, n_s$, where n_s is the number of nodes (discretized states) in stage s.[5]

[1] The polars are a graphical representation of the boat speed (relative to the water) versus its angle to the true wind for different true wind speeds. The tables corresponding to the polars also indicate the apparent wind and angle.

[2] While heading is the direction the vessel is pointing, course (over ground) is the direction of its motion under the influence of the prevailing current and the boat side slip (leeway).

[3] In a real-world scenario, the nodes (waypoints) can be chosen on circles (or polygons) centered at and progressively closer to the destination. The vertical alignment and horizontal positioning is assumed for the ease of explanation in the construction of a trellis across a region.

[4] The waypoints can be changed based on updated weather information and need not be fixed for the duration of the transit from the start to the destination.

[5] T denotes transpose, so (1) is a column vector; $\{\cdot\}$ denotes a set.

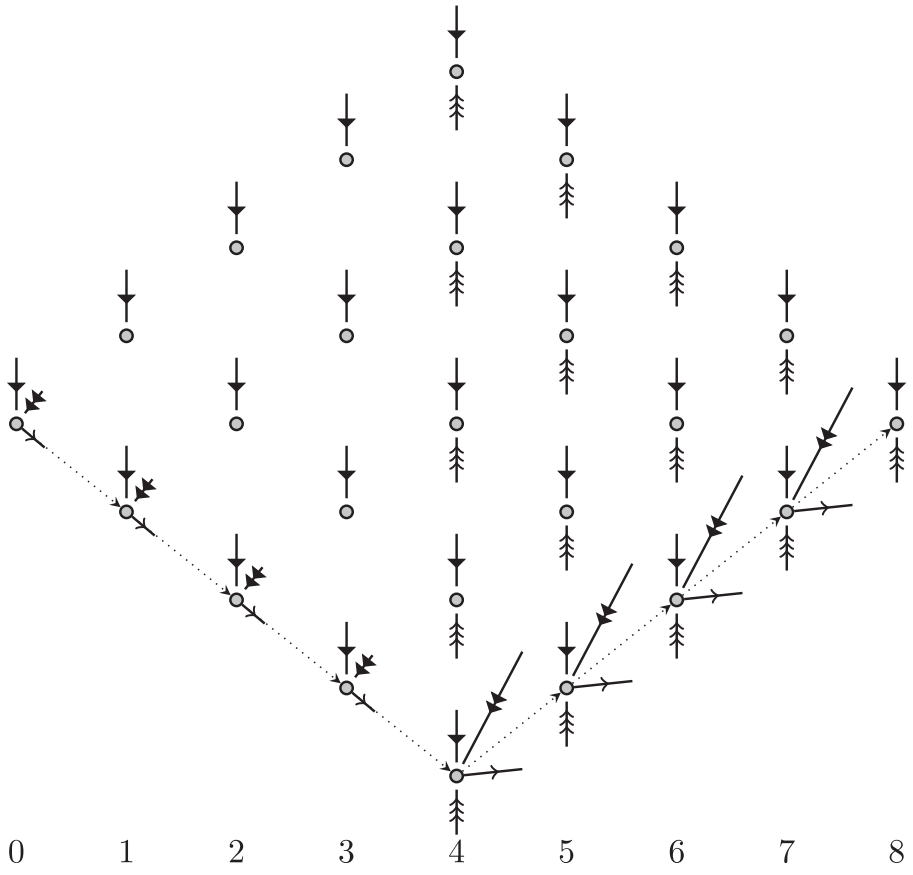

Fig. 2. Illustrative crosswind scenario. The environmental vectors at each possible state within each grid stage generated between the start and destination points. The true wind vectors are illustrated with a single solid arrow, the apparent wind vectors are shown with a double solid arrow, and the current vectors are indicated via a triple thin arrow. The boat velocity vectors in water (including leeway) are illustrated with a single thin arrow. The edges traversed by the sailing vessel between waypoints are dotted with the direction of travel indicated. Note that while the wind is perpendicular to the start–finish line, the point of sail in the first half is (see Table) a broad reach ($\beta_a = -96°$), while, in the second half, it is a close reach ($\beta_a = -56°$). The heading in the second half is around $84°$, so the course is $45°$.

Given the current stage s and location (node) j, $\mathbf{x}_j(s)$, we wish to find the next node $\mathbf{x}_k(s+1)$ to traverse to in stage $s+1$,[6] such that the angle of the edge between $\mathbf{x}_j(s)$ and $\mathbf{x}_k(s+1)$, denoted θ_e, is optimized to allow for maximally fast traversal to the destination, while taking into consideration the wind and current vectors forecast for each node $n \in N$. This process is repeated for stages $s = 1, \ldots, S-2$ (from stage $S-1$, the vessel will head directly to the destination).

At each node n, wind and current vectors are considered, along with the relative angles of the arcs connecting the node to possible waypoints in the next stage $s+1$, to determine the cost to traverse from one stage to the next. Let $\mathbf{R}$ be a vector of cost-impacting environmental parameters to consider (namely, wind and current)

$$\mathbf{R} = \left[\vec{V}_t \ \vec{V}_c\right]^T \tag{3}$$

where $\vec{V}_t$ and $\vec{V}_c$ are the true wind and current vectors comprising magnitudes V_t and V_c and angles θ_t and θ_c, respectively.

We denote the cost to traverse along an edge e, with corresponding bearing θ_e, while experiencing environment $\mathbf{R}$ as $c(e, \theta_e, \mathbf{R})$, and assume it to be nonnegative. The cost, which is the transit time, can be explicitly written as $c[\mathbf{x}(s), \mathbf{x}(s+1), \theta(s), \mathbf{R}]$, where $\theta(s)$ is the heading needed to follow the course θ_e corresponding to the edge connecting $\mathbf{x}_j(s)$ to $\mathbf{x}_k(s+1)$, i.e.,

$$\theta_e = \theta_e\left(\mathbf{x}_j(s), \mathbf{x}_k(s+1)\right). \tag{4}$$

The decision/control variables at each stage s, $s = 0, \ldots, S-2$, are as follows:

1) which node in $\mathbf{x}(s+1)$ to traverse to;
2) which pointing of the vessel, $\theta(s)$, to maintain while traversing to a desired waypoint (node) in $\mathbf{x}(s+1)$, i.e., to achieve a course of $\theta_e(\mathbf{x}_j(s), \mathbf{x}_k(s+1))$.

The minimum cost path to the destination node $\mathbf{x}(S-1)$ from a departure point $\mathbf{x}(0)$, where cost is the traversal time, can be recursively solved via dynamic programming. We assume the goal node to have termination cost $J(\mathbf{x}(S-1)) = 0$ and use dynamic programming [16] to proceed backward in time from

[6]For our illustrative scenario, we assume only forward (toward the destination) traversal to be feasible. This can be changed by modifying the grid based on new weather information and regeneration of the graph given the current location and impending hazards, e.g., storms, where real-time regeneration of the graph, as needed, is possible to allow backward or divert actions. The forward motion on the illustrative grid is necessitated by the proposed algorithm's programming approach; it is the analog of time moving forward.

TABLE I
SUMMARY OF NOTATIONS (FOR BOTH SBP AND CPN [20] ALGORITHMS)

$\vec{V}$	$V\angle\theta_v$, boat velocity vector in water (including leeway)
$\vec{V}_a$	$V_a\angle\beta_a$, apparent wind velocity vector (with respect to the centerline of the boat)
$\vec{V}_c$	$V_c\angle\theta_c$, true current velocity vector
$\vec{V}_g$	$V_g\angle\theta_g$, boat velocity vector over ground
$\vec{V}_t$	$V_t\angle\theta_t$, true wind velocity vector over ground
V	Boat speed in water
V_a	Apparent wind speed
V_g	Vessel speed over ground
V_t	True wind speed
β_a	Apparent wind angle (with respect to the centerline of the vessel; positive for starboard tack, $\beta_a \in [0, 180]$, and negative for port tack, $\beta_a \in [-180, 0]$)
β_c	Apparent current angle (with respect to the centerline of the vessel)
β_g	Velocity over ground angle (with respect to the centerline of the vessel)
β_t	True wind angle (with respect to the centerline of the vessel)
θ	Heading; pointing of boat's centerline (magnetic bearing)
θ_a	Apparent wind angle (magnetic)
θ_c	True current angle (magnetic)
θ_e	Course along edge e
θ_g	Velocity over ground angle (magnetic)
θ_t	True wind angle (magnetic)
θ_v	$\theta - \lambda * \text{sgn}(\beta_a)$
λ	Leeway angle

end stage $s = S - 1$ to $s = 0$. The optimal cost from stage s is

$$J^*(\mathbf{x}_j(s)) = \min_{\mathbf{x}_k(s+1),\theta(s)} (c[\mathbf{x}_j(s), \mathbf{x}_k(s+1), \theta(s), \mathbf{R}] + J^*(\mathbf{x}_k(s+1))) \quad (5)$$

subject to

$$\mathbf{x}_j(s), \mathbf{x}_k(s+1) \in N \quad (6)$$

$$\theta_e(\mathbf{x}_j(s), \mathbf{x}_k(s+1)) \in E. \quad (7)$$

Iteratively following (5)–(7) results in a complete path from the current stage s to the destination. However, forecast environment information is imperfect. Following an open-loop feedback policy [16], [17], [18], the algorithm can be repeatedly executed for a given grid, each time using the most recently available forecast information and assuming that no further updates will be received. New information may be available at the next stage $(s + 1)$ or, if the forecast computation runtime is lengthy, after a finite number of stages. This will require the solution of a maximum of $S - 2$ optimal control problems.

If the forecast consists of more than one value for each stage (with a probability distribution), the algorithm can be generalized to stochastic dynamic programming [19].

B. Sailing Formulation

The notation used in the remainder of this article is listed in Table I. All angles are assumed to be positive clockwise from magnetic North.

A free body diagram of the sailing vessel is shown in Fig. 1, where vectors are drawn in accordance with the vector arrow key in Table II.

In sailing, one only has control over the pointing of the sailing vessel, denoted as θ. The boat speed (magnitude of the velocity vector in the water) is a tabulated function of the vessel pointing and the (apparent, i.e., relative to the boat) wind vector. This function, given by the sailing vessel's polar curves (also tabulated), assumes optimal choice of sails and trimming. In sailing, what is of primary interest is the *apparent* wind angle relative to the vessel. The apparent wind in the tabulated function (based on the polar curves) yields the vessel speed in the water.

TABLE II
VECTOR ARROW KEY

Arrow Type	Vector
→	$\vec{V}$
→»	$\vec{V}_g$
→»»	$\vec{V}_c$
→▸	$\vec{V}_t$
→▸▸	$\vec{V}_a$

Let θ_a denote the apparent wind angle with respect to magnetic North, and let β_a denote the apparent wind angle with respect to the boat centerline. The minimum rotation required to point the vessel into the wind is the circular difference between the control variable θ (heading) and the apparent wind angle θ_a. To calculate this difference for magnetic bearings (i.e., for angles $\theta, \theta_a \in [0, 360)$, where 0° corresponds to magnetic North, with positive values clockwise)

$$\beta_a = f_u(\theta_a, \theta) \triangleq \begin{cases} \theta_a - (\theta - 180^\circ) & \theta_a - (\theta - 180^\circ) \le 180^\circ \\ \theta_a - (\theta + 180^\circ) & \theta_a - (\theta - 180^\circ) > 180^\circ \end{cases} \tag{8}$$

where β_a takes on positive values for starboard tack, $\theta \in [0, 180)$, and negative values for port tack, $\theta \in [180, 360)$. Starboard and port tack refer to which side of the sailing vessel the wind is coming from, where starboard refers to the right-hand side of the vessel when facing forward, and port refers to the left-hand side. The sign of β_a serves as the basis for determining the actual pointing of the vessel in the presence of wind. However, the angle of the velocity vector of the sailing vessel in the water does not coincide with the boat's heading due to another factor called *leeway*. Leeway, denoted by λ, is associated with drift motion behavior (side slip) and is the angular difference between the centerline (the heading) and the velocity angle

$$\theta_v = \theta - \lambda \operatorname{sgn}(\beta_a) \tag{9}$$

where "sgn" is a function that takes the value 1 if $\beta_a > 0$ and -1, otherwise.

A current vector $\vec{V}_c$ impacts the sailing vessel's velocity over ground as

$$\vec{V}_g = \vec{V} + \vec{V}_c \tag{10}$$

where the velocity over ground $\vec{V}_g$ is simply $\vec{V}$ when no current is present.

III. Solution Approach: Arc Cost Evaluation and Path Optimization With Dynamic Programming

The objective of the fastest-path sailing vessel routing problem is to minimize the time to destination. To do so, in the manner discussed in [19], we create the grid G consisting of arcs and waypoints, or marks, to steer along and toward to incrementally progress toward the goal node. At each waypoint $\mathbf{x}_j(s), j = 1, \ldots, n_s$ in stage $s, s = 0, \ldots, S-2$, the cost $c[\mathbf{x}_j(s), \mathbf{x}_k(s+1), \theta(s), \mathbf{R}]$ must be calculated using a heading θ to traverse at course $\theta_e(\mathbf{x}_j(s), \mathbf{x}_k(s+1))$ to waypoint $\mathbf{x}_k(s+1)$ making use of the observed or forecast environment information $\mathbf{R}$.

To determine the estimated transit time from one node to the next, first, the Great Circle (geodesic) distance[7] is calculated as

$$d[\mathbf{x}_j(s), \mathbf{x}_k(s+1)] = 2r \arcsin \times \left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_j(s) \cos\phi_k(s+1) \sin^2\left(\frac{\Delta\psi}{2}\right)} \right) \tag{11}$$

where, without loss of generality, r is Earth's overall mean radius or the mean radius between latitudes $\phi_j(s)$ and $\phi_k(s+1)$, and

$$[\Delta\phi \ \Delta\psi]^T = \mathbf{x}_k(s+1) - \mathbf{x}_j(s). \tag{12}$$

are the latitude and longitude increments between waypoints k and j.

Once the distance $d[\mathbf{x}_j(s), \mathbf{x}_k(s+1)]$ is calculated, the speed at which the sailing vessel can traverse between these waypoints must be determined. The apparent wind is needed to obtain the velocity in water and, from it, the velocity over ground. In iteration $i = 0$, the apparent wind vector is set to the (known) true wind vector

$$\vec{V}_a^0 = \vec{V}_t = V_t \angle \theta_t \tag{13}$$

and we calculate the initial circular angular difference with the vessel's centerline (pointing) as

$$\beta_a^0 = f_u(\theta_t, \theta). \tag{14}$$

Then, we follow an iterative procedure with these initializations from iteration $i = 1$ onward. It is assumed that the sailing vessel comes with tabulated information that describes its behavior, namely, speed and leeway versus apparent wind vector[8] (e.g., see Appendix A-B and [21]). Using an initial assumption of apparent wind magnitude V_a^{i-1} and apparent wind angle magnitude relative to the centerline of the sailing vessel $|\beta_a^{i-1}|$

$$V^i = V\left(V_a^{i-1}, |\beta_a^{i-1}|\right) \tag{15}$$

$$\lambda^i = \lambda\left(V_a^{i-1}, |\beta_a^{i-1}|\right) \tag{16}$$

where (15) and (16) may be calculated using an interpolation method, such as a cubic spline or a radial basis function. The interpolated velocity magnitude (15) and the associated leeway (16) yield (9) at iteration i, rewritten as

$$\theta_v^i = \theta - \lambda^i \operatorname{sgn}(\beta_a^{i-1}) \tag{17}$$

which completes the necessary information to calculate $\vec{V}_a^i$. The next iterated boat velocity vector in water is

$$\vec{V}^i = V^i \angle \theta_v^i \tag{18}$$

[7] A great circle route is the shortest distance between two points located on the surface of a sphere.

[8] This is a more detailed version of the polar curves.

TABLE III
ALGORITHM I: SBP ALGORITHM: FOR A GIVEN HEADING θ, TRUE WIND, AND CURRENT, THIS SHOWS HOW TO OBTAIN THE APPARENT WIND AND THEN THE VELOCITY OVER GROUND

Input:	$\theta, V_t, \theta_t, \vec{V}_c$
Initialize:	
$V_a^0 \angle \theta_a^0 = \vec{V}_a^0$	$\leftarrow \vec{V}_t = V_t \angle \theta_t$
β_a^0	$\leftarrow f_u\left(\theta_a^0, \theta\right)$
Iterate for $i = 1, 2, \ldots$	
V^i	$\leftarrow V\left(V_a^{i-1}, \lvert\beta_a^{i-1}\rvert\right)$ (From table [21])
λ^i	$\leftarrow \lambda\left(V_a^{i-1}, \lvert\beta_a^{i-1}\rvert\right)$ (From table [21])
θ_v^i	$\leftarrow \theta - \lambda^i \operatorname{sgn}(\beta_a^{i-1})$
$\vec{V}^i$	$\leftarrow V^i \angle \theta_v^i$
$\vec{V}_g^i$	$\leftarrow \vec{V}^i + \vec{V}_c = V_g^i \angle \theta_g^i$
$\vec{V}_a^i$	$\leftarrow \vec{V}_t - \vec{V}_g^i = V_a^i \angle \theta_a^i$
β_a^i	$= f_u\left(\theta_a^i, \theta\right)$

with the velocity vector over ground is

$$\vec{V}_g^i = \vec{V}^i + \vec{V}_c = V_g^i \angle \theta_g^i \tag{19}$$

yielding the new apparent wind vector as

$$\vec{V}_a^i = V_a^i \angle \theta_a^i = \vec{V}_t - \vec{V}_g^i \tag{20}$$

and apparent wind angle

$$\beta_a^i = f_u\left(\theta_a^i, \theta\right) \tag{21}$$

where the procedure, detailed in programmatic form in Table III, repeats until the convergence criterion is met. We assume a specifiable threshold

$$\epsilon > \left|\theta_a^i - \theta_a^{i-1}\right| \tag{22}$$

and a maximum number of iterations, to protect against slow or nonconvergence, that, once either is satisfied, terminates the algorithm. Upon termination, $\vec{V}_g$ is known. However, the output θ_g will not necessarily coincide with a desired θ_e. The feasibility of arc traversal is not guaranteed (e.g., in situations where there is significant cross current or if the desired course is into the wind).

To address the issue of achieving a desired θ_g equal to a certain θ_e, assuming that it is feasible, we iterate over a discretized range of feasible headings θ given the desired course θ_e. We can denote the velocity over ground vector angle as a function of the proposed (candidate) pointing $\theta(s)$ in stage s. The accepted pointing is

$$\theta^*(s) = \min_{\theta(s)\in\Theta(s)} \left|\theta_g\left[\theta(s)\right] - \theta_e\right| \tag{23}$$

where $\Theta(s)$ is the set of allowable discretized pointing angles at stage s. The pointing corresponding to the minimum error in traversing from $\mathbf{x}_s$ to $\mathbf{x}_{s+1}$ at course θ_e in (23) is then used with (11) to calculate the stage transition cost.

We make the assumption that the grid G is sufficiently dense to guarantee the optimum traversal to the destination.

For conciseness, we rewrite

$$V_g = f_g\left(\theta, \mathbf{R}\right) \tag{24}$$

where f_g is the speed obtained according to the iteration shown in Table III.

The transit time, which is the arc cost from waypoint j to k, is

$$c\left[\mathbf{x}_j(s), \mathbf{x}_k(s+1), \theta(s), \mathbf{R}\right] = \frac{d\left[\mathbf{x}_j(s), \mathbf{x}_k(s+1)\right]}{f_g\left(\theta(s), \mathbf{R}\right)} \tag{25}$$

and we select the optimal outgoing arcs from each node that provide passage for the sailing vessel to the next stage based on the minimum of (25) over all headings all the way to the destination. For stages $s = 0, \ldots, S-2$, the optimal arc selection is driven by the dynamic programming optimization (5), which is rewritten with (25) as

$$J^*\left(\mathbf{x}_j(s)\right) = \min_{\mathbf{x}_k(s+1), \theta(s)} \left(\frac{d\left[\mathbf{x}_j(s), \mathbf{x}_k(s+1)\right]}{f_g\left(\theta(s), \mathbf{R}\right)} + J^*\left(\mathbf{x}_k(s+1)\right) \right) \tag{26}$$

for $s = S-1, \ldots, 0$ and subject to

$$\mathbf{x}_j(s), \mathbf{x}_k(s+1) \in N \tag{27}$$

$$\theta_e\left(\mathbf{x}_j(s), \mathbf{x}_k(s+1)\right) \in E. \tag{28}$$

The dynamic programming equation (26) yields, together with the next stage waypoint $\mathbf{x}_k^*(s+1)$, the optimal θ^*.s For

details concerning possible implementation improvements, see Appendix C.

IV. Simulation Setup

To demonstrate and validate the proposed SBP algorithm combined with dynamic programming, we consider one illustrative scenario and one real-world scenario. The utility of the illustrative scenario was to assess the algorithms in an ideal setting and to evaluate if the solutions coincided with our intuition. Real-world forecasts can quickly become difficult to understand due to the stochastic, correlated, and nonconvex nature of many weather parameters. The real-world scenario serves as a means to demonstrate the benefit of the proposed algorithm, when applied using real-world weather forecasts, and to evaluate the solution quality given the prevailing conditions of that specified time for the given hull and sails.

In each scenario, we assumed a 36-ft fast cruising boat, specifically a Beneteau First 36.7-Racing Keel,[9] with behavior and speeds, as detailed in [21], where, given the wind speed and the magnitude of the true wind angle relative to the centerline of the vessel, a table lookup scheme was available and input to the software discussed in [19] to obtain the vessel's speed in the water versus the apparent wind and the corresponding leeway as part of the iterative procedure in Table III. Details regarding the tabulated functions for the vessel's speed and leeway for a true wind speed at various true wind angles are summarized in Appendix A-B.

A. Illustrative Scenario

We consider, for the sake of illustration, a fictional "cross-wind" scenario with a graph, G, comprising $S = 9$ stages, where every node has successor nodes located 4 nmi[10] away at a corresponding course of either 45° or 135° (with respect to magnetic North). We assume each successor node to be located in the subsequent stage $s + 1$. Each stage leading up to the middle stage $s = 4$ has incrementally one *more* waypoint node than the previous and, for each stage thereafter, has decrementally one *less* waypoint node than the previous until the final stage. In this manner, we construct a uniform grid G, illustrated in Fig. 2, for evaluation, where stages $s = 0$ and $s = S - 1$ in (5) contain solely the start and destination nodes, respectively.

To validate and compare the route generated by our proposed SBP algorithm against that which was output by an open-source sailing software [20], we set up a simple weather scenario on the grid G. We assumed a true wind speed of $V_t = 4$ kn with an angle of $\theta_t = 180°$ (northerly, i.e., the wind came from angle 0°, with its vector pointing south, i.e., 180°) at each node $n \in N$. We assumed the current speed (in knots) is

$$V_c = \begin{cases} 0, & s < 4 \\ 5, & \text{otherwise} \end{cases} \tag{29}$$

where, when $V_c > 0$, the corresponding angle with respect to magnetic North is $\theta_c = 0°$. In assuming such a weather scenario,[11] we were able to visually validate whether the path and headings suggested by the proposed SBP algorithm were intuitive. In the first half of the graph, the algorithm needed to solely consider the wind, while in the second half, it needed to appropriately consider both the wind and current.

We set a threshold $\epsilon = 10^{-3}$ for our convergence criterion and a maximum number of iteration of 10 loops to protect against slow or nonconvergence.

Two additional illustrative scenarios (downwind and upwind) are discussed in Appendix B.

B. Real-World Scenario

For our real-world scenario, we use the date, start time, and weather available before the beginning of the 2018 Newport Bermuda Race, the year coinciding with the version of the open-source software used as a baseline for comparison in our numerical experiments. The scenario used a graph, G, comprising $S = 16$ stages, where every stage is inserted approximately 36 nmi after the previous, and successor nodes are located at varying distances located between 36 and 600 nmi away at a corresponding minimum and maximum courses of approximately 60° and 240° (with respect to magnetic North), respectively. We assume each successor node to be located in the subsequent stage $s + 1$. Each stage leading up to the middlemost stage has incrementally one or more waypoint nodes, wherein each stage after has decrementally one or less waypoint nodes until the final stage. In this manner, we construct a variable grid G, illustrated in Fig. 3, for evaluation, where stages $s = 0$ and $s = S - 1$ contain solely the start and destination nodes, respectively.

In this article, accurate short- and medium-range weather predictions are used for the real-world scenario. Primary impacting weather data used in our real-world numerical experiment was limited to surface winds and currents, since the SBP algorithm directly relates such weather parameters to a recommended sailing boat pointing. In the numerical weather prediction of the wind and ocean currents, u (vector component towards East) and v (vector component towards North) components are the sufficient means to calculate the speed and direction of these weather parameters in the SBP and OpenCPN algorithms. The u and v surface wind components were ingested from the Navy Global Environmental Model (NAVGEM) [22]. The u and v surface current components are according to the Global Hybrid Coordinate Ocean Model (HYCOM) [23]. A reference forecast[12] date and time of June 15, 2018 and 1200Z (UTC), coinciding with the most recent information that would have been available before the start of the race of that year, was used by the algorithms for routing across the region spanned by the

[9]The specifications of the boat are as follows: overall length, 35'11", draft 7'2", sail area 655 ft^2, and displacement/ballast 12 800/3750 lb.

[10]For a practical problem, a higher density graph would be needed.

[11]While wind speeds on the open ocean (when calm) are generally on the order of 10–15 kn, a much smaller magnitude was chosen (4 kn) to emphasize the effect of currents in the fictional scenario.

[12]A reference time is the date-time when a model creates a new forecast in its moving horizon. The process consists of assimilation and initialization based on observations up until the computation start time.

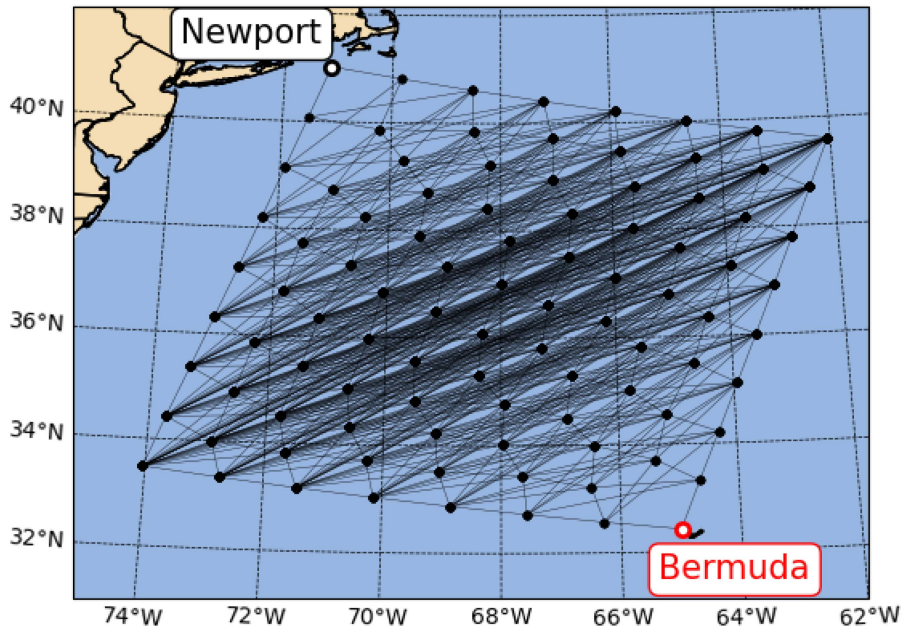

Fig. 3. Real-world scenario. The graph G is displayed using the Lambert conformal conic projection. The lattice structure shown is unidirectional with travel only permitted toward the destination of Bermuda, i.e., traversing backward is not feasible. Each stage s has successor nodes only in stage $s+1$, and each incremental stage has subsequently more candidate nodes until the middlemost stage is reached, wherein the number of candidate nodes decreases until the final destination node is the only one viable in the final stage.

lattice in Fig. 3. The corresponding surface wind barbs[13] and surface current contours/vectors are illustrated in Fig. 4 for a tau[14] of 0 h.

V. Results and Discussion

To demonstrate the benefit of the proposed algorithm, we used the freely available open-source sailing vessel routing optimization software, termed OpenCPN [20] (open-source Chart Plotter Navigation). The results from OpenCPN were obtained using the weather routing optimization plug-in packaged with OpenCPN version 4.6.1, and as translated from C++ into Python. To compare the proposed and baseline algorithms fairly, only the high-level algorithm procedure was translated since OpenCPN relies on the isochrone[15] method for trajectory propagation. The details of the high-level algorithmic procedure contained within OpenCPN are listed in Table IV. Though much of the published literature overlooks the current, the OpenCPN algorithm (herein, referred to as Algorithm II or CPN algorithm) directly considers it in its vector addition operations. The benefit of this method is that the algorithm is fast, taking only one iteration to converge when ocean current is not present.

The proposed SBP algorithm itself takes approximately 1.9 s to obtain an initial trajectory from the origin to the destination, excluding weather ingest routines. If weather ingest routines are included (for surface winds and surface currents), the algorithm runtime increases to approximately 14 s. These runtimes were observed on an Intel Core$^{\mathrm{TM}}$ i7-6650 U CPU Processor 4$\times$ at 2.2 GHz with 16-GB RAM. As the proposed method is coupled with algorithm A*, the worst case complexity of an unbounded search space (i.e., a maximum of S stages to consider) results in $\mathcal{O}(n_\theta n_s^S)$, where n_θ is the number of discretized bearings to consider at each waypoint en route to the destination and n_s is the branching factor (the average number of candidate successors per state). The space complexity of the proposed method is roughly the same as that of other graph search algorithms, as it keeps all generated nodes in memory; however, it differs in that the storage is multiplied by the number of discretized bearings to consider at each waypoint, and the number of forecasts that are applicable to a waypoint (e.g., if there are various ways to reach a waypoint that are at least differing by 6 h regarding the time of arrival, then this constitutes at least two weather forecasts to keep in memory).

A. Illustrative Scenario

Fig. 2 details the path suggested by both the baseline and proposed new algorithms in the illustrative crosswind scenario. The current and true wind vectors at each waypoint are illustrated

[13]Wind barbs are a means of indicating wind direction and intensity. Long and short barbs indicate wind speed, rounded to the nearest 5 kn, while calm wind is displayed as a large open circle. The shaft is used as a tool to visualize the wind angle, pointing to the direction where the wind is coming *from*.

[14]Tau is a meteorological expression that refers to the number of hours postgeneration time of the forecast. A June 15, 2018 forecast reference time of 12Z and tau of 0 h refers to what the weather forecast looks like at 12Z on June 15, 2018. A tau of n hours then refers to n hours after 12Z. In this manner, meteorologists can refer to the exact forecast time, given a reference time. In the example provided, a tau of 3 h refers to the 3-h later forecast of the weather, i.e., 15Z, given the forecast generation time of June 15, 2018 at 12Z.

[15]Isochrone refers to a grid construction method that propagates traversal times out spatially and creates points (or contour lines) that take equal time to arrive at.

TABLE IV
ALGORITHM II: CPN ALGORITHM (BASELINE): OPENCPN ITERATED VELOCITIES AND HEADING CALCULATION

Input:	$\theta_e, V_t, \theta_t, \vec{V}_c$
Initialize:	
$\vec{V}_a^0$	$\leftarrow \vec{V}_t = V_t \angle \underline{\theta_t}$
V_a^0	$\leftarrow V_t$
θ_g^0	$\leftarrow \theta_t$
β_g^0	$\leftarrow 0$
Iterate for $i = 1, 2, \ldots$	
while $\theta_e - \theta_g^{i-1} > 180$	$\theta_e \mathrel{-}= 360$
while $\theta_g^{i-1} - \theta_e > 180$	$\theta_e \mathrel{+}= 360$
β_g^i	$\leftarrow \beta_g^{i-1} + \theta_e - \theta_g^{i-1}$
V^i	$\leftarrow V\left(V_a^{i-1}, \lvert\beta_g^i\rvert\right)$ (From table [21])
θ_v^i	$\leftarrow \theta_t + \beta_g^i$
$\vec{V}^i$	$\leftarrow V^i \angle \underline{\theta_v^i}$
$\vec{V}_g^i$	$\leftarrow \vec{V}^i + \vec{V}_c = V_g^i \angle \underline{\theta_g^i}$
$\vec{V}_a^i$	$\leftarrow \vec{V}_t - \vec{V}_g^i = V_a^i \angle \underline{\theta_a^i}$

using the arrow key detailed in Table II, along with the apparent wind vector calculated at each traversed waypoint. Each algorithm proposed the same path in this scenario; however, the differences between the two methods were in the pointing of the vessel in transit between the chosen waypoints. Note that OpenCPN does not take into consideration the leeway of a vessel, while our proposed method includes it in the optimization. As such, the path suggested by OpenCPN did not strictly adhere to the grid G. To overcome this and more fairly compare the algorithms, we added an outer loop onto the OpenCPN algorithm to iterate over a feasible range of pointings given a node and a candidate waypoint node, at each step in the optimization. This alteration then accounted for leeway and resulted in the OpenCPN algorithm's recommended route being aligned with the grid G.

As seen in Fig. 2, the current impacted the optimization significantly, allowing for faster traversal across the edges in the second half of the problem space $s \geq 4$ and, in turn, impacting the apparent wind encountered. The apparent wind magnitude encountered in the first half of the stages was 3.56 kn, while in the second half of the stages, it more than tripled to 12.0 kn (partially due to the fair current). This is also illustrated in detail in the free body diagrams, shown in Fig. 5, which also serve a dual purpose as a visual for how the environmental vectors are added or subtracted with respect to the sailing vessel in the scenario considered.

The convergence of the SBP algorithm is demonstrated in Fig. 6, where we invoked the proposed procedure assuming that we intended to travel at a course of 135° to the next waypoint, while assuming the true wind V_t to be 4 kn at an angle of 180° (coming from 0°) with no current vectors. The procedure converged relatively fast due to the radial basis function approximation [24] and took a negligible amount of time to complete. In Fig. 6, iteration 6 was not carried out but rather, upon satisfying the threshold condition (that is, θ_a remained the same or was calculated to be within ϵ from the value computed from the previous iteration), terminated the iteration and returned the necessary values to continue with the optimization.

Since we assumed a uniform graph with waypoints generated at fixed distances, there were only two sets of headings generated corresponding to the two possible conditions—without ocean currents ($s = 0, \ldots, 4$) and with ($s = 5, \ldots, 8$). The exact calculations for each of the angles and velocity magnitudes are shown in Table V. A comment on the way the results of a path optimization algorithm are used is as follows: a modern navigation chartplotter is used by setting a waypoint (and steering so that $\vec{V}_g$ points to it—assuming that the wind and current are constant in a leg) rather than following a heading. This way the error due to sideslip/leeway is eliminated in getting to the desired waypoint. Thus, in this illustrative example, the times for both algorithms will be the same. In the illustrative scenario experiment, the CPN algorithm returned the same path (set of waypoints) as the SBP algorithm; hence, they both achieved an estimated route time of 7 h, 7 min, and 33 s from departure to arrival at the destination (e.g., the finish line). Had a sailing boat traversed via Great Circle as approximated by a set of rhumb line segments, the estimated route time would have been 9 h, 48 min, and 13 s. This illustrates the importance of considering environmental impacts, since a Great Circle route is the *shortest*

TABLE V
STAGE VELOCITIES (KNOTS) AND ANGLES (DEGREES); ILLUSTRATIVE SCENARIO

Stage	V_t	θ_t	V_a	θ_a	V_c	θ_c	V	θ_v	θ	β_a	V_g	θ_g
0 (to 1)	4.00	180°	3.56	218°	0.00	–	3.14	135°	134°	–96.0°	3.14	135°
1 (to 2)	4.00	180°	3.56	218°	0.00	–	3.14	135°	134°	–96.0°	3.14	135°
2 (to 3)	4.00	180°	3.56	218°	0.00	–	3.14	135°	134°	–96.0°	3.14	135°
3 (to 4)	4.00	180°	3.56	218°	0.00	–	3.14	135°	134°	–96.0°	3.14	135°
4 (to 5)	4.00	180°	12.0	208°	5.00	0°	5.63	84.0°	83.8°	–55.8°	7.91	45.0°
5 (to 6)	4.00	180°	12.0	208°	5.00	0°	5.63	84.0°	83.8°	–55.8°	7.91	45.0°
6 (to 7)	4.00	180°	12.0	208°	5.00	0°	5.63	84.0°	83.8°	–55.8°	7.91	45.0°
7 (to 8)	4.00	180°	12.0	208°	5.00	0°	5.63	84.0°	83.8°	–55.8°	7.91	45.0°

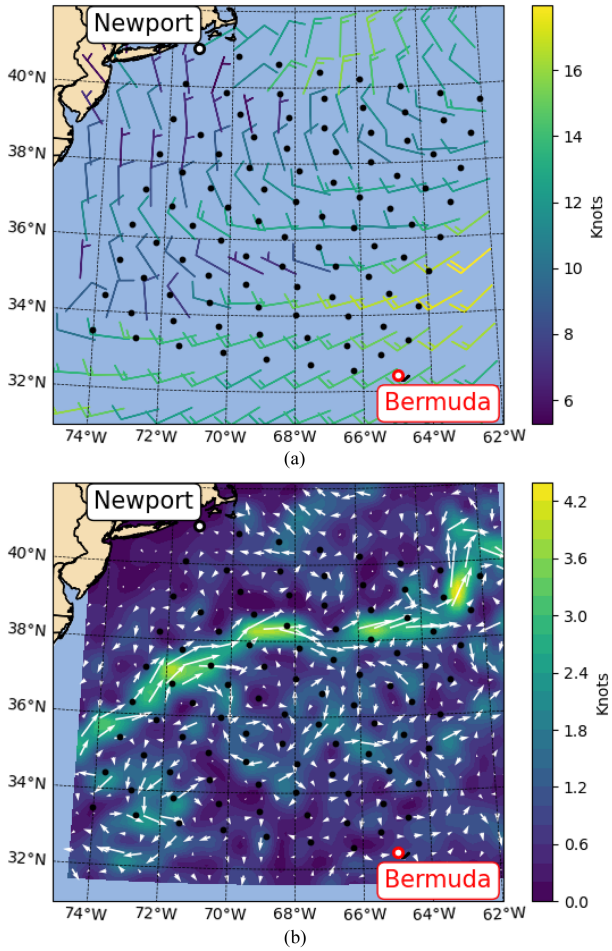

Fig. 4. Lattice waypoints overlayed on top of the real-world scenario's wind and current forecast for the region considered for the 2018 Bermuda Race. Wind barbs and current contours/vectors are illustrated with the Lambert Conformal Conic projection for June 15, 2018, with a forecast reference time of 12Z and a tau of 0 h. Surface wind u and v components are ingested from the NAVGEM and surface current u and v components are according to the Global HYCOM. (a) Real-world scenario wind. (b) Real-world scenario current.

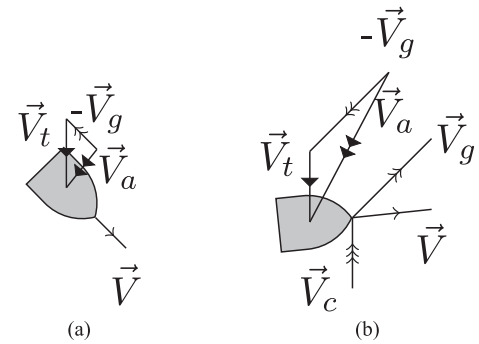

Fig. 5. Free body diagrams corresponding to the proposed SBP algorithm (Algorithm I) for stages $s = 0, \ldots 7$. Stage $S - 1 = 8$ only contained the destination, so, therefore, no further controls were necessary to route the sailing vessel. (a) Stages 1–4. (b) Stages 5–8.

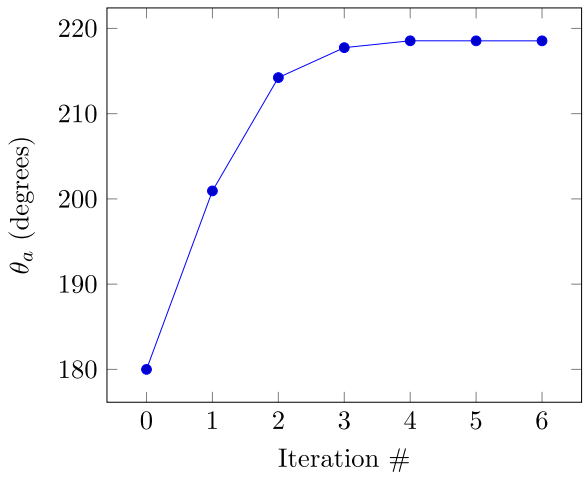

Fig. 6. Convergence of θ_a given the desired mark is at a bearing of 135° (with respect to magnetic North) from the current node and assuming $V_t/\theta_t = 4/180°$ and $V_c/\theta_c = 0, \forall \theta_c$.

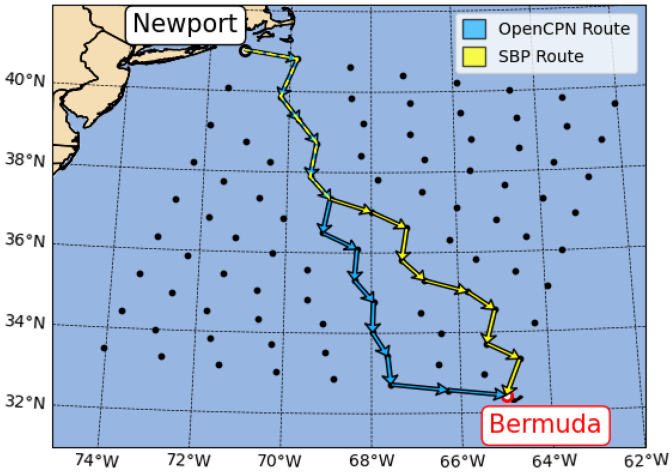

Fig. 7. Real-world scenario. The graph G is displayed using the Lambert conformal conic projection. The lattice structure shown is unidirectional with travel only permitted toward the destination of Bermuda, i.e., traversing backward is not feasible. The edges recommended by the SBP algorithm are shown in yellow (light), while the edges recommended by the CPN algorithm are shown in deep sky blue (darker), with the direction of travel indicated.

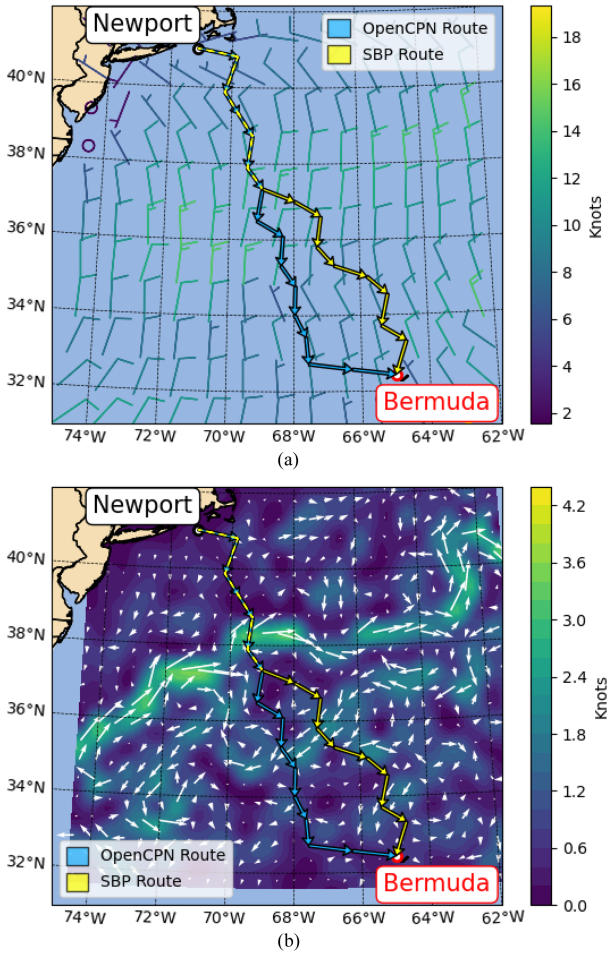

Fig. 8. Recommended routes output by the CPN and SBP algorithms overlayed on top of the surface wind barbs and current contours/vectors using the Lambert conformal conic projection. The surface winds are shown as forecasted with a reference date and time of June 15, 2018, at 12Z with a tau of 30 h. Surface wind u and v components are ingested from the NAVGEM. Surface current u and v components are according to the Global HYCOM. (a) Real-world scenario surface winds. (b) Real-world scenario surface currents.

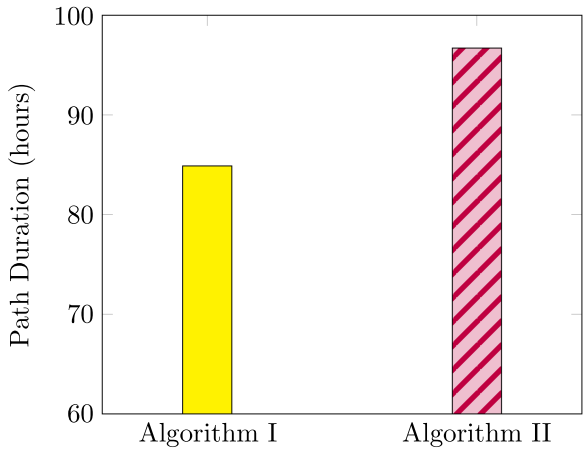

Fig. 9. Comparison of the estimated traversal times of the corresponding paths recommended by Algorithms I and II for the real-world scenario. The total time calculated to travel along the path suggested by Algorithm I, given the recommended waypoints, was ≈ 84.9 h, while the corresponding estimated travel time to sail the route recommended by Algorithm II was ≈ 96.7 h.

route between two waypoints, but not necessarily the *fastest* when there is weather and currents.

Owing to the simplicity of the scenario explored, we did not demonstrate how changing weather condition impacts the optimization. To investigate weather impacts on each algorithm's recommended solution paths, we posed a second scenario involving real-world meteorological and oceanographic data, with discussion of our findings in Section V-B.

B. Real-World Scenario

Fig. 7 details the paths suggested by the baseline and proposed algorithms in the real-world scenario. Each algorithm recommended the same path up until the sixth stage; however, they then deviated in the sequence of waypoints suggested until the final destination was reached. To fairly compare the algorithms in this real-world scenario, we augmented the OpenCPN algorithm to adhere to the grid G by iterating over a range of proposed headings per waypoint candidate and choosing the heading that guaranteed arrival at the candidate node, given the anticipated leeway.

The primary distinction between the two routes begins at the sixth stage, and thus, to better investigate this difference, we plot the surface wind and current forecasts at the time of arrival at the waypoint corresponding to this stage in Fig. 8(a) and (b). As illustrated, the majority of the remaining SBP algorithm-recommended route is aligned with the current vectors from this stage onward, expediting the sailing boat's transit toward the destination. The SBP algorithm is able to prolong sailing distance in advantageous surface current regions, i.e., aligned with the destination. Exploiting this surface current information, in concert with prevailing surface winds in the region, enables the SBP algorithm to achieve a faster overall anticipated race time.

The results pertaining to the estimated transit time from stage $s = 0$ to $s = 16$ are shown in Fig. 9. The CPN algorithm returned a solution that estimated the time to destination to be 4 days,

42 min, and 21 s in duration, while the SBP algorithm recommended a route with a concomitant, estimated duration time of 3 days, 12 h, 52 min, and 40 s—a difference of roughly 11 hours, 49 min, and 41 s of elapsed time in arriving at Bermuda (i.e., the finish line). In comparing the two algorithms, the different headings and deviations in waypoint recommendations of the proposed iterative procedure (SBP algorithm) amounted to more than 12% improvement over the baseline algorithm's recommended route, a relatively large margin of victory in yacht racing.

VI. Conclusion

An iterative procedure to obtain the recommended pointing of a sailing vessel given the ocean current and true wind vectors, paired with a fastest-path algorithm, was derived and shown to be superior to an open-source baseline software. Using observed or forecast weather conditions as input, we presented an iterative procedure that takes into consideration the primary impacting factor in fastest-path sailing vessel routing (the wind), while also taking into consideration a major, but often unused, impactor (the ocean current).

In addition to the proposed iterative procedure, two algorithm comparison test cases were presented that validated the goodness of the proposed procedure, while using open-source software as a baseline for solution quality. An illustrative crosswind scenario was posed to demonstrate the utility and intuition guiding the proposed procedure in an idealized setting, while a real-world scenario was explored to validate the proposed algorithm's improvement in route recommendation over that of the baseline's. The SBP algorithm was able to exploit both real surface current and wind forecast information to achieve a faster anticipated race time over the CPN algorithm, which could only use surface wind information. Our findings indicate that, if they require a high-fidelity fastest-path recommendation that considers leeway motion of the vessel, the proposed procedure will precisely get the sailing vessel from one waypoint to the next, while recommending a sequence of waypoints leading to the desired destination. The key to practicality is the feature of rerunning the SBP algorithm to 1) match the starting time since the optimization runs backward from the end point (finish line) at an initially assumed end time (needed for the weather forecast along the way) and 2) to incorporate updated weather and current data while on the way.

Appendix A
SBP Algorithm Additional Information and Input Data

A. Algorithm Overview

In this section, we present an overview of the SBP algorithm, detailed in Table III. Assuming that a sufficiently dense grid G is available, we invoke multistage dynamic programming (see Appendix C for details) and, in turn, examine the candidate node $\mathbf{x}_k(s+1)$, chosen from the set of neighbor nodes (possible marks) connected by an edge $e \in E$ to the current location $\mathbf{x}_j(s)$, with the corresponding edge course $\theta_e(\mathbf{x}_j(s), \mathbf{x}_k(s+1))$. The required course is known and used in conjunction with a tolerance ϵ, i.e., the allowable deviation off course when traversing from stage s to stage $s+1$.

The SBP algorithm is used over a range of angles enveloping the desired course θ_e. This range amounts to the possible angles to point the vessel to align the velocity over ground angle with that of θ_e. Each proposed pointing θ in this range is input to the algorithm, along with the true wind $\vec{V}_t$, the current $\vec{V}_c$, and a table of performance prediction ("polars"; see Appendix A-B and [21] for an example).

The apparent wind vector at iteration 0, $\vec{V}_a^0$, is initialized to the true wind vector $\vec{V}_t$, namely,

$$V_a^0 \underline{/\theta_a^0} = \vec{V}_a^0 \leftarrow \vec{V}_t = V_t \underline{/\theta_t}. \tag{30}$$

Then, the apparent wind with respect to the centerline of the sailing vessel, β_a^0, is computed as a function of θ_a^0 and the proposed pointing θ as follows:

$$\beta_a^0 \leftarrow f_u\left(\theta_a^0, \theta\right). \tag{31}$$

This completes the initialization.

The iteration begins with interpolation of the vessel speed V and the associated leeway λ from the performance prediction tables

$$V^i \leftarrow V\left(V_a^{i-1}, \left|\beta_a^{i-1}\right|\right) \tag{32}$$

$$\lambda^i \leftarrow \lambda\left(V_a^{i-1}, \left|\beta_a^{i-1}\right|\right). \tag{33}$$

The real vessel pointing is calculable once the leeway is known. This information is used to find the vessel's velocity vector $\vec{V}$

$$\theta_v^i \leftarrow \theta - \lambda^i \operatorname{sgn}(\beta_a^{i-1}) \tag{34}$$

$$\vec{V}^i \leftarrow V^i \underline{/\theta_v^i} \tag{35}$$

where "sgn" is a function that takes the value 1 if β_a^{i-1} is positive, and -1, otherwise. Since the velocity vector of the current and the vessel velocity vector in the water are known, the velocity over ground $\vec{V}_g$ is their sum

$$V_g^i \underline{/\theta_g^i} = \vec{V}_g^i \leftarrow \vec{V}^i + \vec{V}_c. \tag{36}$$

The apparent wind vector $\vec{V}_a$ is calculable as the difference between the true wind $\vec{V}_t$ and $\vec{V}_g$ as follows:

$$V_a^i \underline{/\theta_a^i} = \vec{V}_a^i \leftarrow \vec{V}_t - \vec{V}_g^i. \tag{37}$$

Finally, the apparent wind angle with respect to the centerline of the vessel is computable via f_u once $\vec{V}_a$ is known, namely,

$$\beta_a^i = f_u\left(\theta_a^i, \theta\right) \tag{38}$$

where f_u is defined in (8). The procedure then repeats until $|\theta_a^i - \theta_a^{i-1}| < \epsilon$ or a number of iterations threshold is reached.

Upon termination, $\vec{V}$, $\vec{V}_g$, and $\vec{V}_a$ are returned. Based on the associated θ_g and the desired course θ_e, we can accept or reject the solution. If we reject the solution, we try another feasible pointing within the range generated around θ_e. If we accept the solution, we move on to compute the next node available from the set of neighbor nodes not yet visited.

TABLE VI
PREDICTED BOAT SPEED V AND LEEWAY λ FOR TRUE WIND SPEED $V_t = 4$ KN

	V_t	θ_t	V	VMG	V_a	β_a	λ
Upwind	4.0	30.0	1.864	1.614	5.69	20.6	5.81
	4.0	33.0	2.228	1.869	5.99	21.3	4.58
	4.0	36.0	2.547	2.061	6.24	22.1	3.86
	4.0	39.0	2.833	2.201	6.45	22.9	3.37
	4.0	42.0	3.088	2.295	6.62	23.8	3.01
	4.0	45.0	3.317	2.346	6.76	24.7	2.74
	4.0	46.8	3.435	2.353	6.83	25.2	2.61
	4.0	50.0	3.642	2.341	6.93	26.2	2.40
	4.0	60.0	4.118	2.059	7.03	29.5	1.94
	4.0	70.0	4.405	1.507	6.89	33.0	1.62
	4.0	80.0	4.533	0.787	6.54	37.0	1.36
	4.0	90.0	4.520	0.000	6.03	41.5	1.14
	4.0	100.0	4.345	−0.754	5.37	47.2	0.94
Downwind	4.0	110.0	4.253	−1.454	4.74	52.5	1.00
	4.0	120.0	4.040	−2.020	4.02	59.5	0.84
	4.0	130.0	3.666	−2.356	3.25	70.3	0.66
	4.0	135.0	3.453	−2.442	2.90	77.5	0.56
	4.0	140.0	3.234	−2.477	2.58	86.2	0.47
	4.0	141.3	3.175	2.479	2.50	88.8	0.44
	4.0	150.0	2.789	−2.415	2.11	108.7	0.29
	4.0	160.0	2.374	−2.231	1.95	135.3	0.16
	4.0	170.0	2.143	−2.110	1.93	158.9	0.08
	4.0	180.0	1.999	−1.999	2.00	180.0	0.00

B. Performance Prediction Table

Assuming a 36-ft fast cruising boat, specifically a Beneteau First 36.7-Racing Keel, the vessel characteristics detailed in Table VI were assumed and used as input for both the algorithms. This table, which is more comprehensive than the polars, also shows the apparent wind for the various angles of the true wind and the VMG, i.e., the velocity of the vessel against the wind, as well as the leeway. All values are for the best choice of sails (a spinnaker for downwind) and optimal sail trim.

APPENDIX B
ADDITIONAL SCENARIOS FOR EVALUATION OF SBP

A. Downwind Scenario

We present an illustrative scenario to demonstrate the feasibility of our approach for the case when the bearing of the destination node relative to the origin node is approximately equal to that of the direction of the true wind, θ_t. In downwind sailing, aligning the vessel pointing with that of the true wind is not optimal due to the physics of a sail. A sailing vessel will travel faster when using a spinnaker "pulled" by the wind.

Instead of a "dead run" with the wind, it is favorable to switch between port and starboard tacks to allow the wind to "pull" the vessel. The path[16] recommended by the SBP algorithm (assuming for simplicity $V_c = 0$, $\lambda \approx 0$, i.e., $\vec{V}_g = \vec{V}$), which consists of broad reach legs, is illustrated in Fig. 10. The corresponding free body diagram for each recommended tack in the simulation is detailed in Fig. 11.

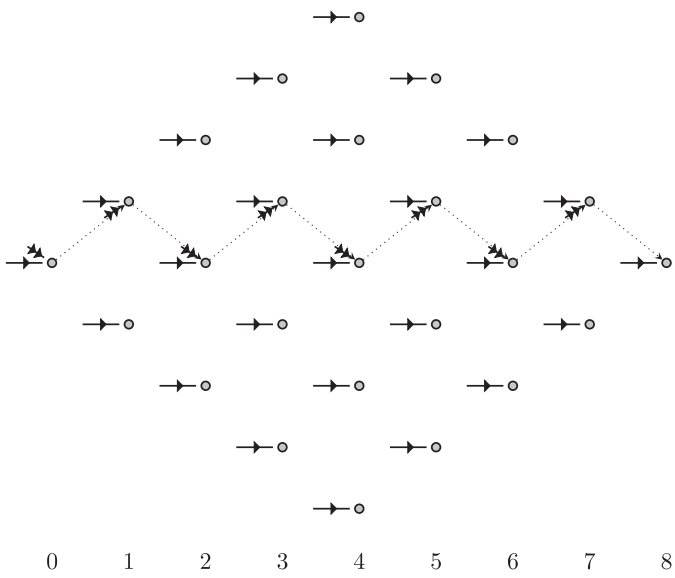

Fig. 10. Downwind scenario. The environmental vectors at each possible state within each grid stage generated between the start and destination points for the scenario of traversing downwind with no current and no leeway. The true wind vectors are illustrated with a single solid arrow, and the apparent wind vectors are shown with a double solid arrow. The edges traversed by the sailing vessel between waypoints are dotted with the direction of travel indicated.

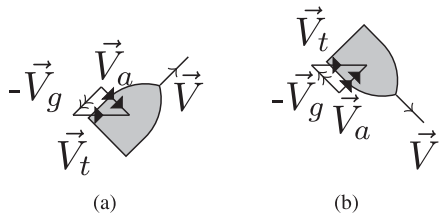

Fig. 11. Free body diagrams corresponding to the proposed SBP algorithm for tacking when sailing downwind ($\vec{V}_g = \vec{V}$ because the current and leeway are assumed to be zero). (a) Port tack. (b) Starboard tack.

The control values and relevant angles and speeds of the vessel are tabulated in Table VII.

Since the algorithm does not penalize tacking, the path has more tacking than a good sailor would do. Penalizing tacking would reduce their frequency in the recommended path. Also, a more dense grid would yield a faster time to the destination. However, given the simplified grid, the traversal time from start to end and *with* tacking was approximately 11 h, 18 min, and 33 s, while *without* tacking (i.e., a "dead run") the calculated course time was roughly 15 h, 38 min, and 13 s.

[16]The optimum VMG downwind is at 140°, while Figs. 10 and 11 show the path at 135° with a VMG 2% below the optimum. To get the optimum, the grid has to be about ten times more dense.

TABLE VII
VELOCITIES (KNOTS) AND ANGLES (DEGREES) ON PORT AND STARBOARD TACK WHEN THE DESTINATION IS DEAD AHEAD

Tack	V_t	θ_t	θ	V_a	θ_a	V	θ_v	V_g	θ_g
Port	4.00	90°	45.0°	3.60	124°	2.83	45.0°	2.83	45.0°
Starboard	4.00	90°	135°	3.60	56.3°	2.83	135°	2.83	135°

TABLE VIII
TACK VELOCITY COMPONENT COMPARISON WHEN TRAVELING UPWIND

Tack	V_t (knots)	θ_t	θ	V_a (knots)	θ_a	V (knots)	θ_v	V_g (knots)	θ_g
Port	4.00	270°	45.0°	7.10	255°	2.62	45.0°	2.62	45.0°
Starboard	4.00	270°	135°	7.10	285°	2.62	135°	2.62	135°

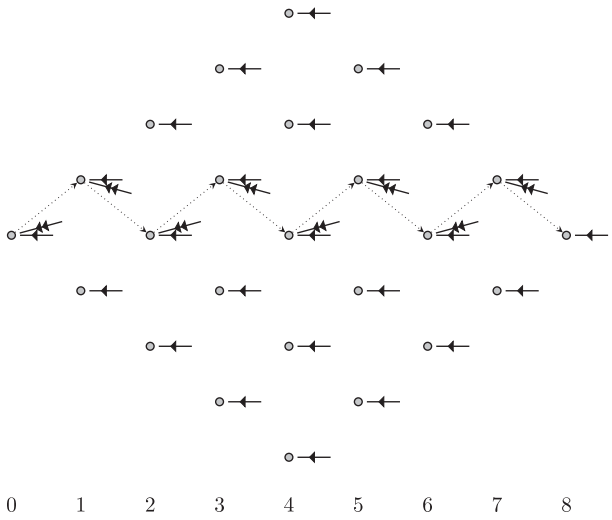

Fig. 12. Illustrative upwind scenario. The environmental vectors at each possible state within each grid stage generated between the start and destination points for the scenario of traversing upwind with no current and no leeway. The true wind vectors are illustrated with a single solid arrow, and the apparent wind vectors are shown with a double solid arrow. The edges traversed by the sailing vessel between waypoints are dotted with the direction of travel indicated.

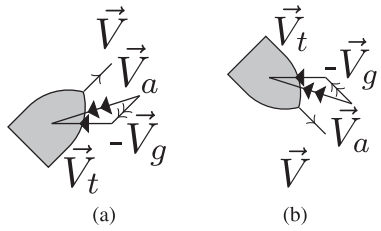

Fig. 13. Free body diagrams corresponding to the proposed SBP algorithm for tacking when sailing upwind. (a) Port tack. (b) Starboard tack.

B. Upwind Scenario

In addition to a downwind sailing scenario, we present a sample path recommended by the SBP algorithm when sailing upwind, i.e., when the difference between the bearing of a destination node relative to the origin node and the angle of the true wind, θ_t, is approximately 180°. In upwind sailing, it is physically impossible to point the vessel in the direction of oncoming winds and proceed toward a mark. Depending on the vessel characteristics, the minimum angle to sail into the wind may vary. For the class of sailing vessel studied in this article, the vessel pointing must differ by at least 30° relative to the oncoming true wind, as detailed in Table VI. The best true wind angle for maximum VMG is around 45°.

Pointing the vessel into the wind is referred to putting the vessel "in irons," so, unlike the downwind scenario, tacking is the only option, switching between port and starboard tacks to allow the wind to "pull" the vessel with maximum VMG to its goal. The path recommended by the SBP algorithm (again, assuming for simplicity $V_c = 0$, $\lambda \approx 0$, i.e., $\vec{V}_g = \vec{V}$) is illustrated in Fig. 12. The corresponding free body diagram for each recommended tack in the simulation is detailed in Fig. 13.

The control values and relevant angles and speeds of the vessel are tabulated in Table VIII.

APPENDIX C
MULTISTAGE PATH SELECTION VIA DYNAMIC PROGRAMMING FOR PATH OPTIMIZATION

Dynamic programming is an iterative procedure that, when followed and given certain assumptions, will provide an optimal solution (in this case, the fastest path). We assume the following.

1) The problem space is discretizable (the path is within a grid).
2) The cost function is additive.

For the basic multistage dynamic programming problem with S stages, we apply S controls (decisions) to go from the initial stage, denoted $\mathbf{x}(0)$ to the end stage $\mathbf{x}(S-1)$. To proceed from stage s to $s+1$, we must apply a control $\theta(s)$. In doing so, we incur a cost $c[\mathbf{x}(s), \mathbf{x}(s+1), \theta(s), \mathbf{R}]$, defined in Section II-A. From our second assumption (i.e., the cost function is additive), the optimal total cost is thus

$$c_S[\mathbf{x}(S-1)] + \sum_{s=0}^{S-2} c[\mathbf{x}^*(s), \mathbf{x}^*(s+1), \theta^*(s), \mathbf{R}] \qquad (39)$$

where $c_S[\mathbf{x}(S-1)]$ is the terminal cost of the end stage (if any), $\mathbf{x}^*(s)$ and $\mathbf{x}^*(s+1)$ are the optimally chosen states in the respective stages s and $s+1$, and $\theta^*(s)$ is the pointing of the vessel that takes it (given the prevailing wind and current) from $\mathbf{x}^*(s)$ to $\mathbf{x}^*(s+1)$. By "optimal" in the fastest-path sailing vessel routing problem, we mean the shortest overall transit time. The transit time from one stage to the next is given by (25). Using backward dynamic programming, we know the goal stage to be $\mathbf{x}(S-1)$, which contains a single state, that is, the goal node. Hence, $\mathbf{x}^*(S-1)$ is known. Proceeding backward, we find that the minimum cost over all traversable states in the previous stage is

$$\mathbf{x}^*(S-2) = \arg\min_j c[\mathbf{x}_j(S-2), \mathbf{x}^*(S-1), \theta^*(S-2), \mathbf{R}]. \tag{40}$$

The optimal node is selected for each stage by

$$\mathbf{x}^*(s) = \arg\min_j c[\mathbf{x}_j(s), \mathbf{x}^*(s+1), \theta^*(s), \mathbf{R}] \quad s = S-2, \ldots, 0 \tag{41}$$

and

$$\mathbf{x}^*(S-1) = 0. \tag{42}$$

Knowing the optimal cost from one node to the next stage, we aim to solve for the optimal path over all stages. To do so, we can condense the problem to that of finding the best cost from each node to the next stage and calculating the optimal "cost-to-go" J^* from there onward. The problem then reduces to (5), reproduced as follows:

$$J^*(\mathbf{x}_j(s)) = \min_{\mathbf{x}_k(s+1), \theta(s)} \left(c[\mathbf{x}_j(s), \mathbf{x}_k(s+1), \theta(s), \mathbf{R}] + J^*(\mathbf{x}_k(s+1)) \right) \tag{43}$$

$$= \min_{\mathbf{x}_k(s+1), \theta(s)} \left(\frac{d[\mathbf{x}_j(s), \mathbf{x}_k(s+1)]}{f_g(\theta(s), \mathbf{R})} + J^*(\mathbf{x}_k(s+1)) \right) \tag{44}$$

where f_g is defined in (24).

In practical applications of dynamic programming, there are sometimes opportunities to reduce the problem space further by exploiting domain-specific information to estimate the costs to go. In lieu of (44), we may use an approximating function $\tilde{J}(\mathbf{x}_k(s+1))$, which we assume to be both positive and optimistic, to estimate the true cost to go $J^*(\mathbf{x}_k(s+1))$ from the next node $\mathbf{x}_k(s+1)$

$$\hat{J}(\mathbf{x}_j(s)) = \min_{\mathbf{x}_k(s+1), \theta(s)} \left(\frac{d[\mathbf{x}_j(s), \mathbf{x}_k(s+1)]}{f_g(\theta(s), \mathbf{R})} + \tilde{J}(\mathbf{x}_k(s+1)) \right). \tag{45}$$

The approximation of (26) as (45) amounts to the utilization of algorithm A* [25], [26] to aid in solving the fastest-path sailing vessel problem for the minimum time path planning. This algorithm limits the search for the sake of speedup.

REFERENCES

[1] *Royal Bermuda Yacht Club and the Cruising Club of America*, Newport Bermuda Race, Newport, RI, USA, 2017. [Online]. Available: http://bermudarace.com/

[2] A. Philpott and A. Mason, "Optimising yacht routes under uncertainty," in *Proc. 15th Cheasapeake Sailing Yacht Symp.*, 2001, pp. 2001–009.

[3] D. S. Ferguson and P. Elinas, *A Markov Decision Process Model for Strategic Decision Making in Sailboat Racing*. Berlin, Germany:Springer, 2011, pp. 110–121.

[4] S. P. Ladany and O. Levi, "Search for optimal sailing policy," *Eur. J. Oper. Res.*, vol. 260, no. 1, pp. 222–231, 2017.

[5] F. Tagliaferri, A. Philpott, I. Viola, and R. Flay, "On risk attitude and optimal yacht racing tactics," *Ocean Eng.*, vol. 90, pp. 149–154, 2014.

[6] M. Życzkowski, P. Krata, and R. Szłapczyński, "Multi-objective weather routing of sailboats considering wave resistance," *Polish Maritime Res.*, vol. 25, no. 1, pp. 4–12, 2018.

[7] R. Stelzer and T. Pröll, "Autonomous sailboat navigation for short course racing," *Robot. Auton. Syst.*, vol. 56, no. 7, pp. 604–614, 2008.

[8] L. Xiao, J. C. Alves, N. A. Cruz, and J. Jouffroy, "Online speed optimization for sailing yachts using extremum seeking," in *Proc. IEEE Oceans*, 2012, pp. 1–6.

[9] H. Saoud, M. D. Hua, F. Plumet, and F. B. Amar, "Optimal sail angle computation for an autonomous sailboat robot," in *Proc. IEEE 54th Conf. Decis. Control*, 2015, pp. 807–813.

[10] D. Wrede, J. Adam, and J. Jouffroy, "Online optimization of different objectives in robotic sailing: Simulations and experiments," in *Proc. IEEE Conf. Control Appl.*, 2015, pp. 876–881.

[11] J. Cabrera-Gámez, J. Isern-González, D. Hernández-Sosa, A. C. Domínguez-Brito, and E. Fernández-Perdomo, "Optimization-based weather routing for sailboats," in *Robotic Sailing*. Berlin, Germany: Springer, 2013, pp. 23–33.

[12] T. Akiyama, J.-F. Bousquet, K. Roncin, G. Muirhead, and A. Whidden, "An engineering design approach for the development of an autonomous sailboat to cross the atlantic ocean," *Appl. Sci.*, vol. 11, no. 17, 2021, Art. no. 8046.

[13] N. M. Kristensen, "Weather routing: Sensitivity to ensemble wind and current input," *master's thesis*, Dept. Geosci., Univ. Oslo, Oslo, Norway, 2010.

[14] V. Futch and A. Allen, "Search and rescue applications: On the need to improve ocean observing data systems in offshore or remote locations," *Front. Mar. Sci.*, vol. 6, 2019, Art. no. 301.

[15] N. Salvesen, E. Tuck, and O. Faltinsen, "Ship motions and sea loads," in *Proc. Soc. Nav. Architects Mar. Eng. Annu. Meeting*, 1970, pp. 250–287.

[16] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Belmont, MA, USA:Athena Scientific, 1995.

[17] Y. Bar-Shalom and R. Sivan, "On the optimal control of discrete-time linear systems with random parameters," *IEEE Trans. Autom. Control*, vol. AC-14, no. 1, pp. 3–8, Feb. 1969.

[18] Y. Bar-Shalom and E. Tse, "Dual effect, certainty equivalence, and separation in stochastic control," *IEEE Trans. Autom. Control*, vol. AC-19, no. 5, pp. 494–500, Oct. 1974.

[19] D. Sidoti et al., "A multiobjective path-planning algorithm with time windows for asset routing in a dynamic weather-impacted environment," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 12, pp. 3256–3271, Dec. 2017.

[20] OpenCPN chart plotter navigation, 2017. [Online]. Available: https://opencpn.org/

[21] *Performance Prediction—Design #446, First 36.7 574 Racing Keel for Chantiers Beneteau S.A.*, FARR Yacht Design, Ltd., Annapolis, MD, USA, 2001. [Online]. Available: http://www.blur.se/polar/first367_performance_prediction.pdf

[22] T. Whitcomb, "Navy global forecast system, NAVGEM: Distribution and user support," in *Proc. 2nd Sci. Workshop ONR DRI: Unified Parameterization Extended Range Prediction*, 2012.

[23] G. Halliwell, R. Bleck, and E. Chassignet, "Atlantic ocean simulations performed using a new hybrid-coordinate ocean model," in *Proc. EOS, Fall AGU Meeting*, 1998, pp. 1–30.

[24] J. Travers, R. Hetland, and T. Oliphant, "scipy.interpolate.Rbf." Accessed: Jan. 31, 2022. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.Rbf.html

[25] P. E. Hart, N. J. Nilsson, and B. Raphael, "Correction to "A formal basis for the heuristic determination of minimum cost paths"," *ACM SIGART Bull.*, no. 37, pp. 28–29, 1972.

[26] N. J. Nilsson, *Principles of Artificial Intelligence*. San Mateo, CA, USA: Morgan Kaufmann, 2014.

David Sidoti received the B.S., M.S., and Ph.D. degrees from the University of Connecticut, Storrs, CT, USA, 2011, 2016, and 2018, respectively, all in electrical and computer engineering.

Since 2018, he has been a Computer Scientist with the Meteorological Applications Development Branch, Marine Meteorology Division, U.S. Naval Research Laboratory, Monterey, CA, USA. He is an NRLMMD's primary Machine Learning subject matter expert and contributes to a broad portfolio of Machine Learning projects relating to atmospheric and oceanographic forecasting. He was also responsible for transitioning several software for waterspace allocations and undersea munitions deployments operations to Program Executive Office—Integrated Warfighting Systems 5 Echo. He has authored or coauthored three book chapters, 30 journal articles, and 34 conference papers.

Dr. Sidoti was a corecipient of the Tammy Blair Award for best student paper at 2016 International Conference on Information Fusion. He was recognized and awarded a Distinguished Scholar Jerome and Isabella Karle's Fellowship upon joining U.S. Naval Research Laboratory in 2018. In 2022, he received the Assistant Secretary of the Navy Research, Development, and Acquisition's Dr. Delores M. Etter Top Scientists and Engineers of the Year 2022 Emergent Scientist Investigator Award for his transitioned undersea optimization work.

Krishna R. Pattipati (Fellow, IEEE) received the B.Tech. (Hons.) degree in electrical engineering from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1975, and the M.S. and Ph.D. degrees in systems engineering from the University of Connecticut (UConn), Storrs, CT, USA, in 1977 and 1980, respectively.

From 1980 to 1986, he was with ALPHATECH, Inc., Burlington, MA, USA. He is currently with the Department of Electrical and Computer Engineering, UConn, where he is also the Distinguished Professor Emeritus and the Collins Chair Professor of Systems Engineering. He is a cofounder of Qualtech Systems, Inc., Rocky Hill, CT, a firm specializing in advanced integrated diagnostics software tools (TEAMS, TEAMS-RT, TEAMS-RDS, and TEAMATE) and serves on the board of Aptima, Inc., Woburn, MA. His research interests include proactive decision support, uncertainty quantification, smart manufacturing, autonomy, knowledge representation, and optimization-based learning and inference. A common theme among these applications is that they are characterized by a great deal of uncertainty, complexity, and computational intractability.

Dr. Pattipati was selected by the IEEE SYSTEMS, MAN, AND CYBERNETICS SOCIETY as the Outstanding Young Engineer of 1984. He received the Centennial Key to the Future Award. From 1998 to 2001, he was the Editor-in-Chief for IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS. He was a corecipient of the Andrew P. Sage Award for the Best IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS Paper for 1999, the Barry Carlton Award for the Best IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS Paper for 2000, the 2002 and 2008 NASA Space Act Awards for "A Comprehensive Toolset for Model-based Health Monitoring and Diagnosis," and "Real-time Update of Fault-Test Dependencies of Dynamic Systems: A Comprehensive Toolset for Model-Based Health Monitoring and Diagnostics," and the 2003 AAUP Research Excellence Award at UConn. He is Elected Fellow of the Connecticut Academy of Science and Engineering.

Yaakov Bar-Shalom (Fellow, IEEE) was born in 1941. He received the B.S. and M.S. degrees from Technion, Haifa, Israel, in 1963 and 1967, respectively, and the Ph.D. degree from Princeton Univ., Princeton, NJ, USA, in 1970, respectively all in electrical engineering.

He is currently a Board of Trustees Distinguished Professor with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA, and he is also the M. E. Klewin Professor with the University of Connecticut. He has authored or coauthored more than 650 papers in his research areas and in stochastic adaptive control and eight books including *Estimation With Applications to Tracking and Navigation* (Hoboken, NJ, USA: Wiley, 2001) and *Tracking and Data Fusion* (Hoboken, NJ, USA: Wiley, 2011). His current research interests include estimation theory, target tracking, and data fusion.

Dr. Bar-Shalom was a corecipient of the M. Barry Carlton Award for the best paper in IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS in 1995 and 2000 and the recipient of the 2008 IEEE Dennis J. Picard Medal for Radar Technologies and Applications and the 2012 the Connecticut Medal of Technology. He has been listed by academic.research.microsoft as no. 1 in Aerospace Engineering based on the citations of his work. He is also the recipient of the 2015 International Society of Information Fusion (ISIF) Award for a Lifetime of Excellence in Information Fusion, renamed in 2016 as "ISIF Yaakov Bar-Shalom Award for Lifetime of Excellence in Information Fusion." He was an Associate Editor for IEEE TRANSACTIONS ON AUTOMATIC CONTROL and *Automatica* and a General Chairman of 1985 American Control Conference and 2000 International Conference on Information Fusion. He was the President of the ISIF in 2000 and 2002 and Vice-President Publications from 2004 to 2013.