Annals of Glaciology

Article

Cite this article: Schmitz B, Eis C, Bünger HJ, Büskens C (2024). Estimation of travel time reduction for ice-breaking ships when considering ice information data. *Annals of Glaciology* 1–11. https://doi.org/10.1017/aog.2024.28

Received: 31 December 2023
Revised: 26 April 2024
Accepted: 4 June 2024

Keywords:
polar and subpolar oceans; remote sensing; sea ice

Corresponding author:
Bernhard Schmitz;
Email: bschmitz@uni-bremen.de

cambridge.org/aog

Estimation of travel time reduction for ice-breaking ships when considering ice information data

Bernhard Schmitz[1,2], Christine Eis[1], H. Jakob Bünger[2] and Christof Büskens[1]

[1]WG Optimization and Optimal Control, Center for Industrial Mathematics, University of Bremen, Bremen, Germany and [2]Drift+Noise Polar Services, Bremen, Germany

Abstract

In light of the recent increase in polar shipping and potential future increase with continued climate change reliable routing in ice-covered waters becomes increasingly important for environmental, economic and safety concerns. Dependable route suggestions have the potential to reduce travel times through polar waters significantly. We apply the Anytime Repairing A* pathfinding algorithm to classified Copernicus Sentinel 1 radar images to estimate how much travel times can be reduced. For multiple example scenarios, it is quantified how much the travel time is reduced if a ship follows these suggestions compared to navigating without any ice information available exterior to the visual range (VR). It was found that having ice information available is most beneficial in complex ice situations, where it can reduce travel time by up to 34% for a VR of 2 km.

Introduction

The decrease in sea-ice coverage leads to an increasing number of ships operating in the Arctic Ocean. Shipping routes open up in summer but will close in winter (Melia and others, 2017). The exact time of freeze-up is uncertain and varies between different regions, resulting in increased safety issues during the transition period (Li and others, 2021). In areas with favourable ice conditions for shipping, ships can still be damaged by single, undetected and thicker ice floes (Shibata and others, 2013).

Even without the challenges and increased uncertainty resulting from ongoing climate change, navigating ice-covered waters is inherently more difficult than operating under open water/ice-free conditions. Navigation of a vehicle is dependent on the navigator's knowledge of what lays ahead. Weather is the most relevant route-impacting factor under open-water conditions, while ice conditions and drift determine routes in ice cover. Without any knowledge of these impact factors, ship navigators have to rely on what they see outside their windows. The distance within which it is possible to recognize properties of ice depends on several factors. In addition to, e.g. weather and lighting conditions, the height of the bridge over the water (observation angle) or the experience of the observer play an important role. Within this study, we assume that nautical staff can identify different ice features at a visual range (VR) of 2 km of the ship. In addition to optical observations, an ice radar is commonly used, which allows ice to be detected at distances of up to ∼7400 m (Canadian Coast Guard, 2022, Chapter 4.9) depending on the device and weather conditions.

To improve this very limited amount of information, the demand for large-scale ice information increases (Melia and others, 2017). Having access to recent earth observation data can be helpful for planning ahead when navigating icy waters. Reliable near-real-time data enable ice navigators to not only consider the ice situation close to the ship but also take the situation along the whole upcoming journey into account. This will improve safety as dangerous areas like ice ridges or bergy water can be either circumnavigated or other safety measures can be applied. Ice information can also reduce travel time as it reduces the likelihood of unexpected situations where a ship has to turn back or get stuck due to ice conditions. By reducing the travel time required for navigation either more time could be employed for mission related tasks or the target could be reached more quickly.

There already exist numerous approaches for calculating routes in icy waters. They can be categorized based on the input data (ice information), ship performance models, objective function and route optimization techniques (Lehtola and others, 2019; Tran and others, 2023).

Different sources of ice information might be the reason for different spatial and temporal resolutions of the data considered for route calculation. For example, ice charts, ice models or radar data are used. For supporting operational planning, a high geospatial and temporal resolution of the input data is required (Lehtola and others, 2019) as it is necessary to resolve local variations of the ice (Kaleschke and others, 2016). Therefore, this study makes use of classified Copernicus Sentinel 1 radar data that come at a resolution of 160 m, which resolves relevant features in the ice reasonably well. Other approaches tend to use lower resolution input data, for example 1 nm × 1 nm (Lehtola and others, 2019).

As mentioned above, route optimization can help to reduce the time required to cover a certain distance and increase safety. There are also studies that optimize the distance travelled or fuel consumption (Tran and others, 2023). Furthermore, cooperative routing, thus optimal

usage of icebreaker support, was investigated (Topaj and others, 2019). Depending on what is (more) important within a specific situation (or for a study) an objective function is defined. It is also possible to optimize for multiple criteria. However, this comes with the question of how different objectives should be prioritized (Browne and others, 2022). The objective function is an essential attribute to the routing algorithm, and is further described within the 'Methods' section.

Ship performance models are used to quantify how difficult or costly navigation through an area actually is. For this, boundary conditions like ice thickness, ice age and ship parameters are used. As a result, a ship performance model can provide information on the optimal speed for a well-known ship within any given situation or could state whether a leg should be navigated at all as the chance of getting stuck is too high (Lehtola and others, 2019; Tran and others, 2023).

The algorithms most commonly used for ice routing are graph-based (Tran and others, 2023). In this study, we also used an algorithm of this family, namely the A* algorithm (respectively a variant of it). The details are described within the 'Methods' section.

The goal of this study is to evaluate how much travel time can be saved when making use of satellite data as described above or even when using an assistance system which suggests routes. We are trying to achieve that by comparing the results of a fully informed pathfinding algorithm to a simulated ship track. To simulate a ship which is travelling without having ice information data available, the algorithm is modified to just consider ice conditions within the VR of the ship. Route suggestions for four different scenarios are compared to give an estimation on how much travel time could be saved when using additional ice information or even a system that provides route suggestions based on that data.

Methods

Path finding using the A algorithm*

A route suggestion R is a sequence of $N+1$ ($N \in \mathbb{N}$) waypoints $\vec{w}_n$ with $n \leq N$ and $n \in \mathbb{N}_0$. As each node references a geospatial location it is written as vector. The first waypoint $\vec{w}_0$ equals the origin and the last waypoint $\vec{w}_N$ equals the target. To emphasize that the number of waypoints within the final route is unknown until the algorithm is finished, the target node is written as $\vec{\omega}$:

$$R = (\vec{w}_0, \vec{w}_1, \ldots, \vec{\omega}) \tag{1}$$

The A* algorithm (Hart and others, 1968) attempts to find a sequence of nodes that connect $\vec{w}_0$ and $\vec{\omega}$ on a graph Q in such a way that the cost for traversing the edges is minimal. In the following we use $\vec{w}_n \in Q$ to reference waypoints (nodes) which are part of the route suggestion R. All other nodes (that are candidates for becoming part of the final route) are denoted with $\vec{q} \in Q$. As A* is an informed algorithm, the total cost $f(\vec{q})$ for visiting a certain node $\vec{q}$ consist of the cost $g(\vec{q})$ for getting from origin to $\vec{q}$ plus the cost $h(\vec{q})$ of getting from $\vec{q}$ to the target:

$$f(\vec{q}) = g(\vec{q}) + h(\vec{q}) \tag{2}$$

$f(\vec{q})$ is called the objective function which should be minimal. While $g(\vec{q})$ is known, $h(\vec{q})$ has to be estimated by using a heuristic, which we define as:

$$h(\vec{q}) = \frac{\sqrt{(q_x - \omega_x)^2 + (q_y - \omega_y)^2}}{v_{\mathrm{ow}}} \tag{3}$$

$h(\vec{q})$ equals the time for travelling at the line-of-sight with the velocity v_{ow}, which is the velocity of the ship in open water. The component of the geospatial coordinate of each node is referenced with the subscripts x or y. If the heuristic underestimates the cost for reaching the target, A* guarantees to find an optimal solution (Hart and others, 1968), wherefore a lower bound is chosen. Using a heuristic like (3) makes the algorithm prefer nodes from which the goal is expected to be reached more favourably. For the given heuristic h this applies to nodes which are located in the direction of the target. As a consequence, the algorithm (in general) will not visit all nodes of a given area of interest but only relevant ones (Hart and others, 1968). This reduces computational costs in terms of time and memory consumption.

Exploration starts at the given origin node $\vec{w}_0$ and derives the respective costs f for travelling to every child node. Each child node is added to the priority queue (PQ) which is sorted by travel cost in ascending order. In addition to that, the cost for each node is tracked within the node map (NM). Subsequently, following the 'best first' approach, the first node of the priority queue is selected and its child nodes are explored, too. In turn, they are added to the PQ and NM. This is repeated until the first node of the PQ equals the target node. If a child node was visited before, it is added to the PQ and NM only if the current costs are less than the cost of the previous visit. As each node stores a reference to its parent node, it is possible to build the route starting at the target node and tracing the way back to the origin. If the PQ runs empty before hitting the target, the route calculation failed and no route can be suggested. The whole process is visualized in Figure 1.

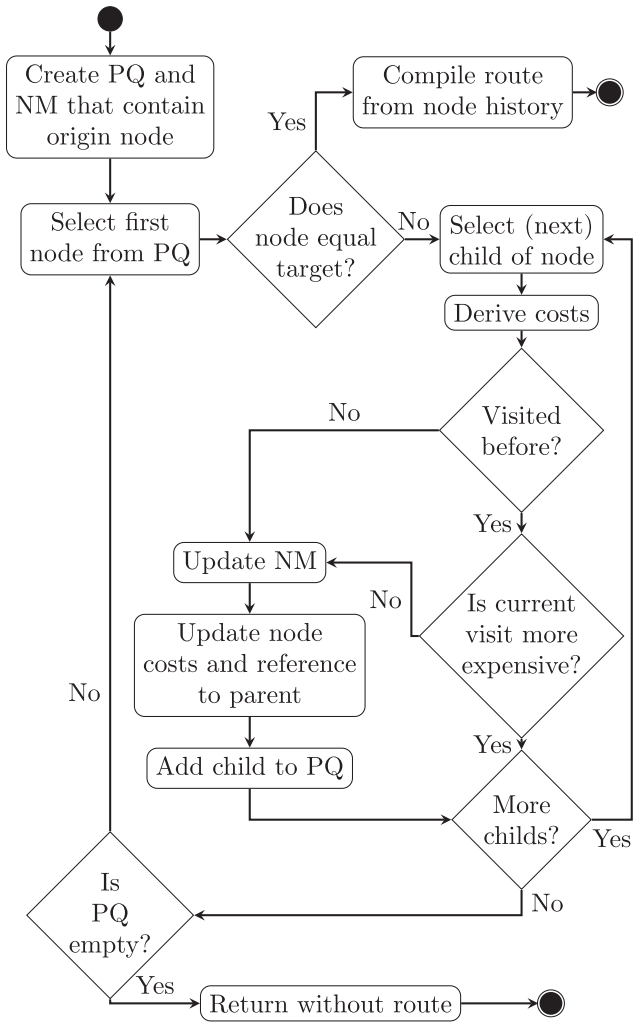

Figure 1. Workflow of the A* algorithm. It makes use of a priority queue (PQ) to make sure that the most interesting node is investigated within the next loop. The node map (NM) is used to track the costs of all nodes for being able to decide if it should be reinvestigated after a revisit.

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

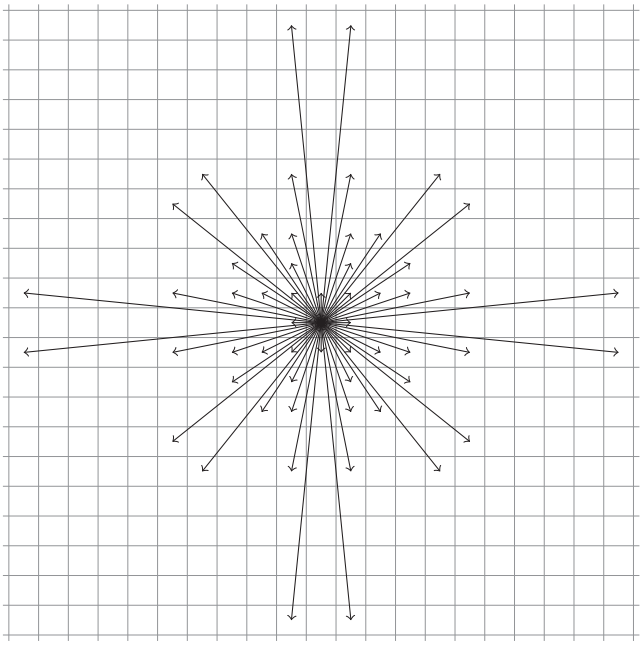

Figure 2. A* move pattern that allows travelling in 56 directions. To keep the angle between possible move directions similar and small the length of some move options is increased. Modified from Guinness and others (2014).

The ice conditions are provided as a raster image, where each pixel can be interpreted as a node. To be able to use these data, it is required to set up edges between the nodes to create a graph which can be traversed. This is done dynamically for each node when it is visited by A* by applying a move pattern which specifies child nodes for the currently selected node. In this study, a pattern with 56 options is used, following Guinness and others (2014). This move pattern, which is visualized in Figure 2, was designed with the goal to minimize the deviations of the angles between the move options. Note that A* works out an an optimal solution for a specific graph. Using another move pattern would result in another graph and therefore another route suggestion.

*Long-distance routing with Anytime Repairing A**

Within heuristic path algorithms (Pohl, 1970) the heuristic $h(\vec{q})$ is weighted by $\epsilon \in \mathbb{R} \geq 1$ to intensify the behaviour of the algorithm to prefer nodes from which the goal can be reached more favourably. Therefore, the objective function (2) turns into

$$f_\epsilon(\vec{q}) = g(\vec{q}) + \epsilon h(\vec{q}) \tag{4}$$

By applying a weight $\epsilon > 1$, the costs for travelling from a certain node to the target could be overestimated, in which case the algorithm no longer guarantees to find the optimal route (Pohl, 1970). In return, a larger weight ϵ increases the prioritization of nodes that are located in the direction or close to the target and therefore speeds up the algorithm as less nodes are visited (Hart and others, 1968). In the literature, a modification of A* that uses this approach is often referred to as weighted A* (e.g. Thayer and Ruml, 2010).

Calculating routes over long distances or with high-resolution data is a complex task because a large number of nodes have to be considered. Users of a route service might expect a route suggestion to be available within a short time frame. When dealing with such complex environments application of the original A* algorithm would not be feasible to serve these expectations. To solve this problem one can use an anytime algorithm which comes with a trade-off between quality and runtime (Zilberstein, 1996). In this approach, the algorithm is expected to present a non-optimal, but feasible, solution R_0 within a short time (e.g. by using a weighted heuristic) but keeps running and returns further solutions R_i with $i \in \mathbb{N}_0$ that get better the longer the algorithm runs. Therefore, the travel cost of R_i is expected to be equal or larger than the costs for travelling R_{i+1}. If the algorithm runs without a time limit and is therefore not interrupted, it finally returns an optimal solution $R_{i_{\max}}$.

A*-based algorithms that work according to this principle include the Anytime Repairing A* (ARA*) (Likhachev and others, 2003) and the Anytime Weighted A* (AWA*) (Hansen and Zhou, 2007). The AWA* uses a constant weight to get a quick solution. If it is not interrupted, all relevant nodes are examined so that the final solution is optimal. In contrast, the ARA* uses a dynamic weight, which is adjusted depending on the solutions found. Nevertheless, ARA* can reuse intermediate results. In addition, the ARA* has a mechanism that prevents nodes from being evaluated multiple times in one iteration. Comparisons between these two algorithms suggest, that ARA* is more appropriate for path-finding with a large number of nodes (Thayer and Ruml, 2010). As this is a requirement which we face in this study, routing is carried out using the ARA* algorithm.

Figure 3 visualizes the workflow of ARA*. It utilizes a waiting list for revisited nodes, which is called revisit list (RL): if a node has been visited before and a cheaper solution for getting to this node is found, it is not added to the PQ but is added to RL instead. Only nodes that are visited for the first time are added to the PQ. This makes the algorithm not re-investigate existing nodes if there are still new nodes to be explored. After the target was found, the costs for reaching the target with that first solution is considered as maximum cost. The algorithm continues investigating the nodes, but ignores all nodes which are more expensive than that. The PQ runs empty after all promising nodes were visited. Afterwards, the PQ is re-initialized with the nodes from the RL. Therefore, for each position on the map, only the cheapest node is considered. In addition to that, nodes with an unweighted total cost that exceeds the cost for reaching the target are ignored, too. This condition is written as

$$f(\vec{q})|_{\epsilon=1} \geq g(\vec{\omega}) = f_\epsilon(\vec{\omega})$$

A new weight is set to the ratio between the cheapest known costs g for travelling from the origin to the target (the best route suggestion found so far) and the minimal unweighted costs of all nodes stored within RL. In case the algorithm is interrupted, the (possibly non-optimal) Route R_i that was calculated last serves as a route suggestion.

Limited VR routing

The goal of this study is to estimate how much the travel time could be reduced when using a route suggestion compared to travelling without any ice information available but the ice conditions within the VR of the ship. This could also serve as comparison between travelling with sea-ice information like satellite data available versus travelling without information available.

To simulate how a ship relying on visual navigation would travel, the algorithm is modified to assume the most expensive ice condition $c_{\max}$ for all nodes $\vec{q}$ exterior to the VR r of the current ship location $\vec{w}_0^{(j)}$:

$$\tilde{c}(\vec{q}) = \begin{cases} c(\vec{q}) & \text{if} \left|\vec{q} - \vec{w}_0^{(j)}\right| \leq r \\ c_{\max} & \text{otherwise} \end{cases} \tag{5}$$

For all scenarios described in this study, the most expensive ice

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

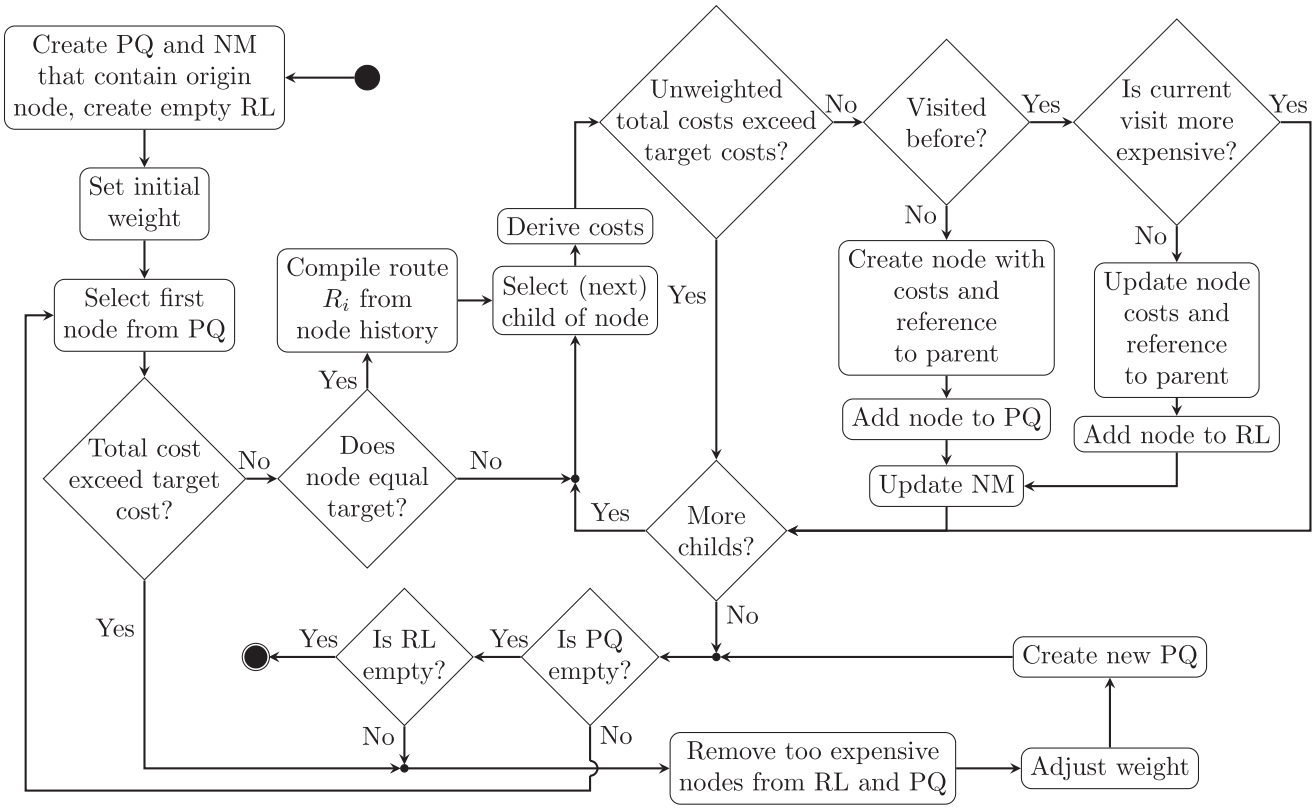

Figure 3. Workflow of the ARA* algorithm (Likhachev and others, 2003) as it has been implemented for this study. In addition to a priority queue (PQ) and a node map (NM) a revisit list (RL) is used as a waiting list for nodes that have been visited before and should not immediately been evaluated again. In contrast to the A* algorithm, the ARA* algorithm makes use of a dynamically weighted heuristic and returns multiple routes R_i which get better the longer the algorithm runs.

condition is multiyear ice. The ice conditions $c(\vec{q})$ are derived from satellite data as mentioned within the next section. Equation (5) is implemented as a filter for querying node cost, thus ARA* is no longer allowed to access the (full information) graph directly but is executed for a limited graph. The filter is initialized with the ship position $\vec{w}_0^{(j)}$ and VR radius r in such a way that its centre coincidences with the ship's position. Thereby, the ARA* algorithm finds the fastest route within VR, but makes sure that there is a sensible trade-off between the interior and the (high) exterior travel cost. Other options are discussed later on.

For calculation of the ship's track, the ARA* algorithm is invoked multiple times for subsequent ship positions $\vec{w}_0^{(j)}$, which is denoted by the index j. First, a route

$$R^{(0)} = \left(\vec{w}_0^{(0)}, \vec{w}_1^{(0)}, \ldots, \vec{\omega}\right)$$

is calculated from origin $\vec{w}_0^{(0)}$ (initial ship position) to target $\vec{\omega}$. Once this route is known, the ship moves to a new position $\vec{w}_1^{(0)}$ which is the second waypoint of the initial route $R^{(0)}$ and equals the first waypoint (origin) $\vec{w}_0^{(1)}$ of the next route. The filter is reinitialized with the changed ship position $\vec{w}_0^{(1)}$. We assume r to remain constant during a specific journey. Subsequently, the algorithm is invoked again for calculating the route

$$R^{(1)} = \left(\vec{w}_0^{(1)}, \vec{w}_1^{(1)}, \ldots, \vec{\omega}\right)$$

As visualized in Figure 4 this is repeated until the first waypoint of a route $\vec{w}_0^{(j_{\max}+1)}$ (equals $\vec{w}_1^{(j_{\max})}$) is equal to $\vec{\omega}$, which indicates that the target has been reached. The final route is given by the series of ship positions:

$$R = \left(\vec{w}_0^{(0)}, \ldots, \vec{w}_0^{(j_{\max})}, \vec{\omega}\right)$$

For efficiency reasons, each node that is exterior to the VR is restricted to have the target node as only child node, which means the ship moves along the line-of-sight from that node to the target (compare Fig. 5). This adds another move option, which is cheaper than summing up smaller non-line-of-sight move options. However, as the track exterior to the visible range is

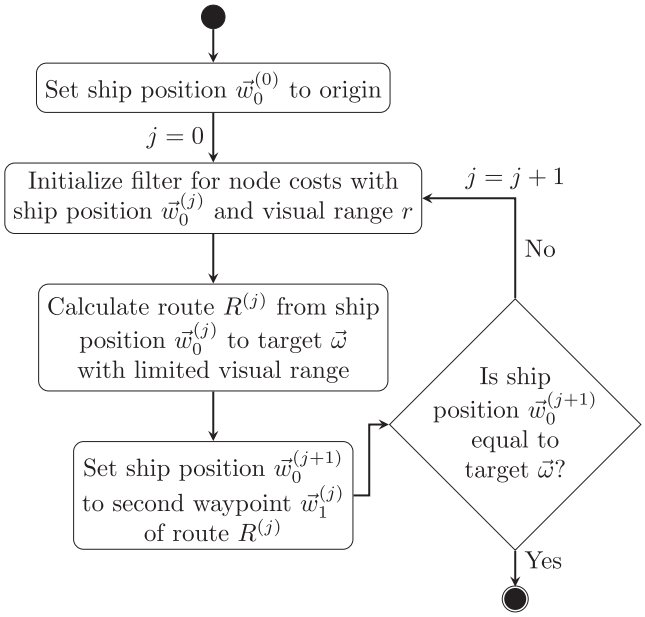

Figure 4. Strategy for calculation of limited VR routes. Each time the ship moves to a new position $\vec{w}_0^{(j)}$ the filter for querying node cost according to (5) has to be (re)initialized and the ARA* algorithm is invoked for calculating a route from $\vec{w}_0^{(j)}$ to the target $\vec{\omega}$.

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

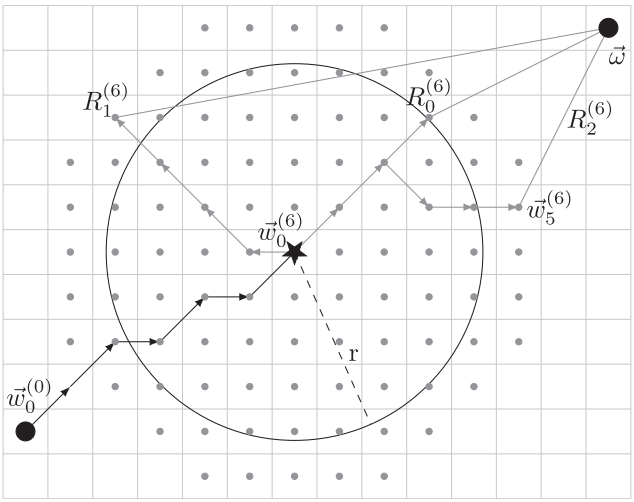

Figure 5. Route is calculated from an origin $\vec{w}_0^{(0)}$ to a target $\vec{\omega}$. For the ship position after six iterations $\vec{w}_0^{(6)}$ and a VR r the nodes visualized by grey dots are considered for routing to the target. For nodes within the VR (circle) ARA* is applied as usual, while nodes exterior to the VR have the target as the only child. The image shows three possible routes $R_0^{(6)}$, $R_1^{(6)}$ and $R_2^{(6)}$ from the current ship position to the target. The subscript number is the sequential number of the ARA* solution, thus cost for $R_1^{(6)}$ are greater or equal to the cost for $R_2^{(6)}$. For simplicity, this visualization assumes a move strategy that allows horizontal, vertical and diagonal movement only.

not considered when building the final limited VR route, this is neglected.

For VRs below 10 km, solutions were calculated for all multiples of 1 km and for 7400 m because that is the assumed VR when using an ice radar. The computational costs rise with increasing VR, as ARA* has to consider larger areas. Therefore, route suggestions are calculated for multiples of 2 km only up to a maximum VR of 20 km.

Scenarios

Information on ice conditions within the test regions Baffin Bay and Hudson Bay is given by Copernicus Sentinel 1 (S1) data which was processed by the European Space Agency (ESA) in February 2020 and January 2023. Classifications for that acquisitions are derived and provided by the German Aerospace Center (DLR). The resolution of these classifications is four times lower than the resolution of the respective Sentinel 1 acquisitions, which is 40 m (Murashkin and Frost, 2021). This enables a routing algorithm to consider small features with a size of 160 m, for example small leads.

Both classified images contain six different classes: multiyear ice, first-year ice, new ice, rough ice, smooth surface leads and rough surface leads. To reduce complexity, the number of classes considered for route calculation is reduced to three: two classes for ice and one class for open water. Table 1 names the classes used for route calculation and also specifies the assumed travel velocities within each class. Within this study, all routes are calculated based on these velocities. This is an experimental setup. In real-life applications ship parameters and operation practices have to be considered for correct time estimation.

Table 1. Velocities for different ice types and water used to compute travel costs between two locations

Class name	Content	Velocity m s^{-1}
Multiyear ice	Multiyear ice	1.0
Other ice	First-year ice, new ice, rough ice	2.5
Water	Smooth surface leads, rough surface leads	8.2

Within the two selected regions, four different routes are calculated. Each route is defined by its origin and target location. The line-of-sight tracks as well as the ice conditions in the surrounding of the tracks are visualized in Figures 6a, b. Each scenario name is prefixed with an abbreviation of its region name. The '3' indicates that three ice or water types are considered. We selected the scenarios in a way that they cover areas with homogeneous and inhomogeneous ice conditions for travelling through. Additionally, we investigated situations which allow travelling within leads and also situations where a ship cannot just travel in leads but has to enter the ice. For example, the scenario bb3_lead should allow the routing algorithm to make use of some leads, especially at the end of the track while bb3_crossing crosses some leads. bb3_crossing and bb3_lead travel through few homogeneous areas, while hb3_noland mostly travels through homogeneous areas, which are in both cases classified as 'other ice'. In contrast to that, bb3_southnorth mostly travels multiyear ice, which encloses some (smaller) features of water or other ice. These descriptions are valid for the line-of-sight tracks, however the route suggestions will, in general, deviate from the line-of-sight to reduce travel time. Table 2 provides the line-of-sight distance per scenario. In the following, we will consider hb3_noland as scenario with homogeneous ice conditions and the other three scenarios bb3_lead, bb3_crossing and bb3_southnorth as rather inhomogeneous.

For later comparison, the travel times for travelling along the line-of-sight were calculated (Table 2). This was not done with ARA* but by accumulating the distances travelled within the different ice conditions and applying the ice condition specific velocities. Note that the line-of-sight is considered as the Euclidean distance on a plane (WGS84/NSIDC Sea Ice Polar Stereographic projection) which is also used by the ARA* algorithm as implemented here.

Results and discussion

Travel times of the route suggestions for each scenario as calculated with the ARA* algorithm are visualized in Figure 7. The travel times for the final route suggestions are also given in Table 3. As described, the ARA* algorithm returns multiple route suggestions for decreasing weights. While the initial weight is 1 for all scenarios, the subsequent weights are depending on the travel time of the previous found route. The time it takes to find a route suggestion highly depends on the scenario. For the scenarios with rather mixed patches of ice and leads, especially for bb3_lead the progress of travel time reduction is not as continuous as it is for hb3_noland which contains rather homogenous ice conditions. For the scenarios with rather complex, mixed ice and water surfaces, especially for bb3_lead, the progress of travel time reduction is not as continuous as for hb3_noland. Comparing the travel time of the first solution to the travel time of the last solution shows that ARA* was able to reduce travel time by 5% (hb3_noland) to 7.5% (bb3_southnorth). We also found that, following the optimal route, a ship could save 22.4% (hb3_noland) to 50.8% (bb3_lead) travel time compared to following the line-of-sight. This is visualized in Figure 8. Note that this figure shows the travel time for following the line-of-sight relative to the travel time of the optimal route.

The calculation of the second solution for bb3_lead takes ~50% of the total calculation time. This is in strong contrast to the other scenarios in which the second solutions are determined much faster. After finding the second solution, the travel time determined for bb3_lead decreases significantly. Such a sudden decrease in travel time cannot be observed for the other scenarios, or can only be observed in a mild form. Calculating the optimal route for bb3_lead takes less time than calculating the optimal

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

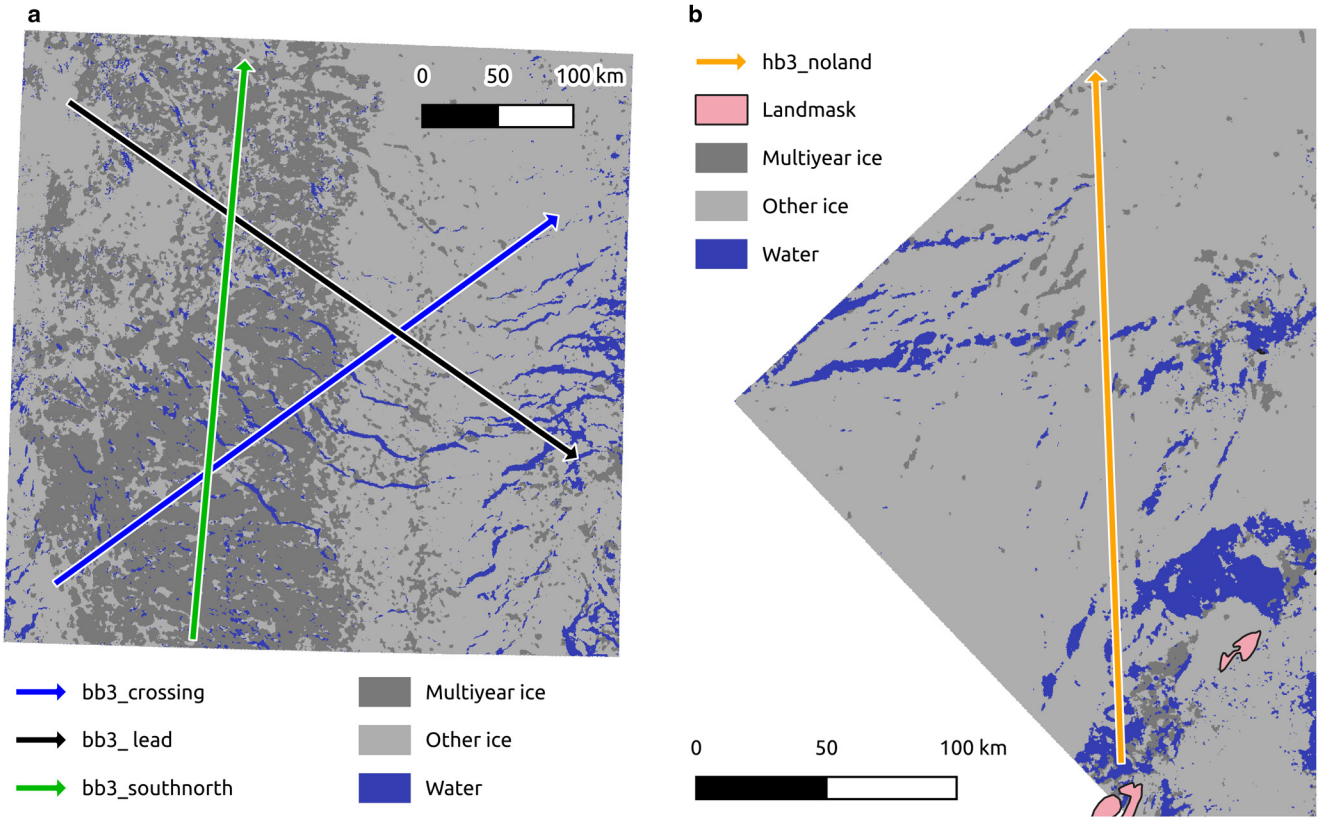

Figure 6. Visualization of the scenarios for route calculation. Contains modified Copernicus Sentinel 1 data (2023), processed by ESA. The underlying Sentinel 1 data were classified by the German Aerospace Center (DLR), according to Murashkin and Frost (2021): (a) scenarios at Baffin Bay. Underlying S1 acquisition: S1A_EW_GRDM_1SDH_20230117T212054_20230117T212154_046829_059D73_6422. (b) Scenario at Hudson Bay. Underlying S1 acquisition: S1A_EW_GRDM_1SDH_20230117T114207_20230117T114307_046823_059D46_6ABB.

solution for hb3_noland. This is noticeable because the line-of-sight distance between origin and target for bb3_lead is ∼147% of the distance for hb3_noland. A possible explanation for this behaviour is the presence of leads that extend in the direction of travel at bb3_lead and which are exploited by the algorithm. In the other scenarios, leads in the direction of travel do not occur at all or to a much lesser extent. This suggests that the ice conditions within the area of interest have an influence on the performance of the algorithm.

Figure 7 also shows that for hb3_noland the final weight ϵ is significantly larger than 1. In this case, the optimal solution was already found for $\epsilon = 1.22$. Although the algorithm continued, it did not work out faster routes for smaller weights.

The time it takes until the optimal route suggestion is delivered is certainly too long, a potential user of a route service does not wait for several days until a route is suggested. More important, the ice situation will be different after some hours, so the value of the route suggestion is questionable. Computations for this study were carried out on a system with 4 cores with 8 threads at a base frequency of 3.5 GHz and 768 GB RAM. However, the current implementation does not make use of multiple threads, i.e. is single threaded. Within this study we do not compare calculation times, but focus on the resulting route suggestions. Therefore, we can leave this issue aside.

Table 2. Line-of-sight (LOS) distances and times for travelling along them

Scenario	LOS distance km	LOS travel time
hb3_noland	281.3	29 h 45 min 32 s
bb3_southnorth	386.22	80 h 19 min 3 s
bb3_lead	412.36	60 h 4 min 43 s
bb3_crossing	413.94	70 h 49 min 7 s

LOS refers to the Euclidean distance on a plane between the origin and target of each scenario.

The optimal route suggestions calculated with the ARA* algorithm which was provided with full information regarding the ice conditions are compared to the limited VR ARA* solutions. Figure 8 visualizes the travel times for these route suggestions relative to the travel time of the unlimited VR route suggestions for all four scenarios, depending on the VR. The travel times of the results for the scenarios with rather heterogeneous ice conditions (bb3_lead, bb3_southnorth and bb3_crossing) show a trend of reduced travel times for increasing VRs. This relation is not as strong for the more homogeneous scenario hb3_noland. The experiment shows that limiting the knowledge of the surrounding ice condition to a radius of 7400 m increases travel time by ∼12% when travelling in areas with rather homogeneous ice conditions like hb3_noland. For the same situation, a VR of 2 km increases travel time by 18%. The mean values of the Baffin Bay scenarios are used to indicate the increase in travel time for more complex scenarios. That gives an increase of 37% for a VR of 7400 m and 47% for a VR of 2 km compared to the solution with unlimited knowledge.

Figure 9 shows the relation between the route length and the VR. Again, the ARA* solution which considers all ice conditions (unlimited range) is utilized for displaying all other solutions relative to it. Lower VR route suggestions are, in general, shorter than suggestions with larger VR or unlimited VR. The shortest route follows the line of sight, ignoring any ice information available (visibility range equals zero). For more complex scenarios the length difference between the line-of-sight solution and the unlimited VR solution gets larger.

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

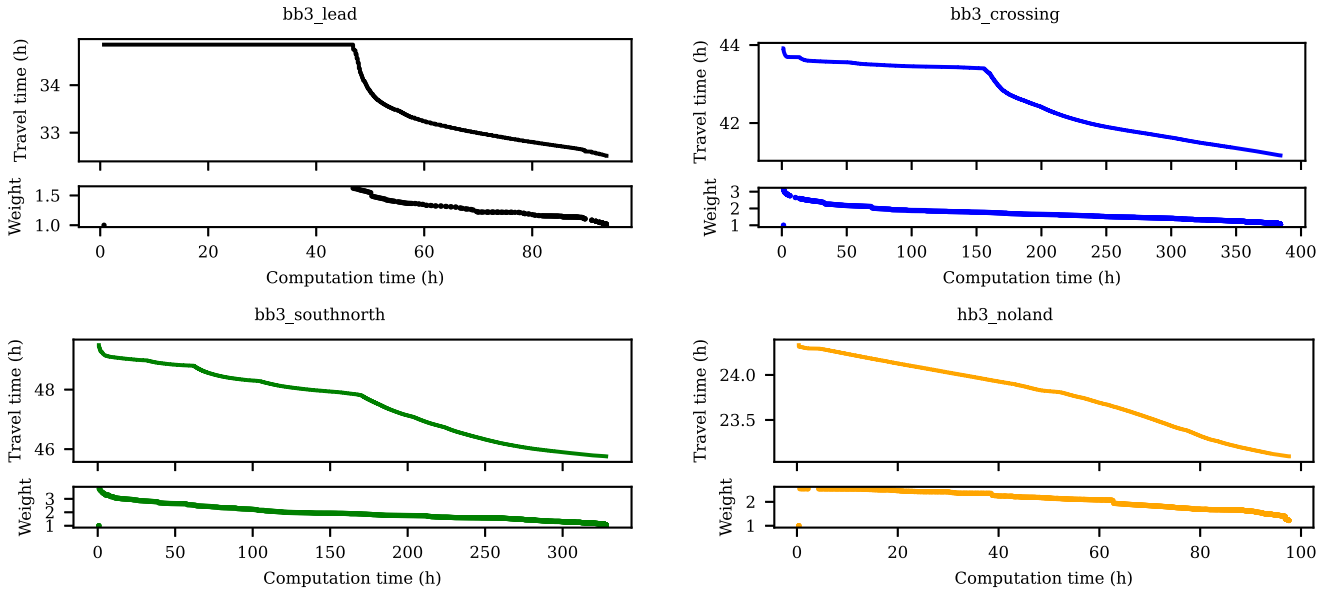

Figure 7. Route suggestions for the given scenarios as calculated by a ARA* algorithm with unlimited VR and an initial weight of $\epsilon = 1$. For each scenario the travel times decrease over time and with smaller weight. The final travel time and route length is also given in Table 3.

In regions with inhomogenous ice regimes the required travel time decreases with an increase in VR (Fig. 8). For homogeneous regions this effect is not that strong. This suggests that a routing or ice information system is more beneficial if there are different features around which need to be avoided (like multiyear ice) or which could be utilized (like open water leads) and if the VR is small. This is also in line with the observations which were made regarding the route length: the higher the VR is, the more remote features can be detected, which helps the navigator to favour more distant but potentially more suitable features over close ones that end up in difficult conditions eventually. This is possible as the algorithm does not optimize for travel distance but for travel time. These extra distances sum up to an increased route length. This effect is largest for the unlimited VR routes, therefore the travelled distance for this suggestion exceeds the length of other suggestions.

In a few cases, the travel times for a solution with a larger VR exceed the travel time for another solution with a smaller VR. This is because some larger VR routes are attracted by features that are not visible to the lower VR routes when being at the same location. Even though using these features is faster now, it could be more expensive later on when travelling subsequent areas which are unknown by now. If the lower VR route decides in the same location for a different path, this could enable the usage of e.g. leads within the subsequent legs, which are not reachable by the larger VR route. In turn, the travel time is lower within that part of the route. Figure 10 illustrates this by an example spotted within the bb3_lead scenario. In the area

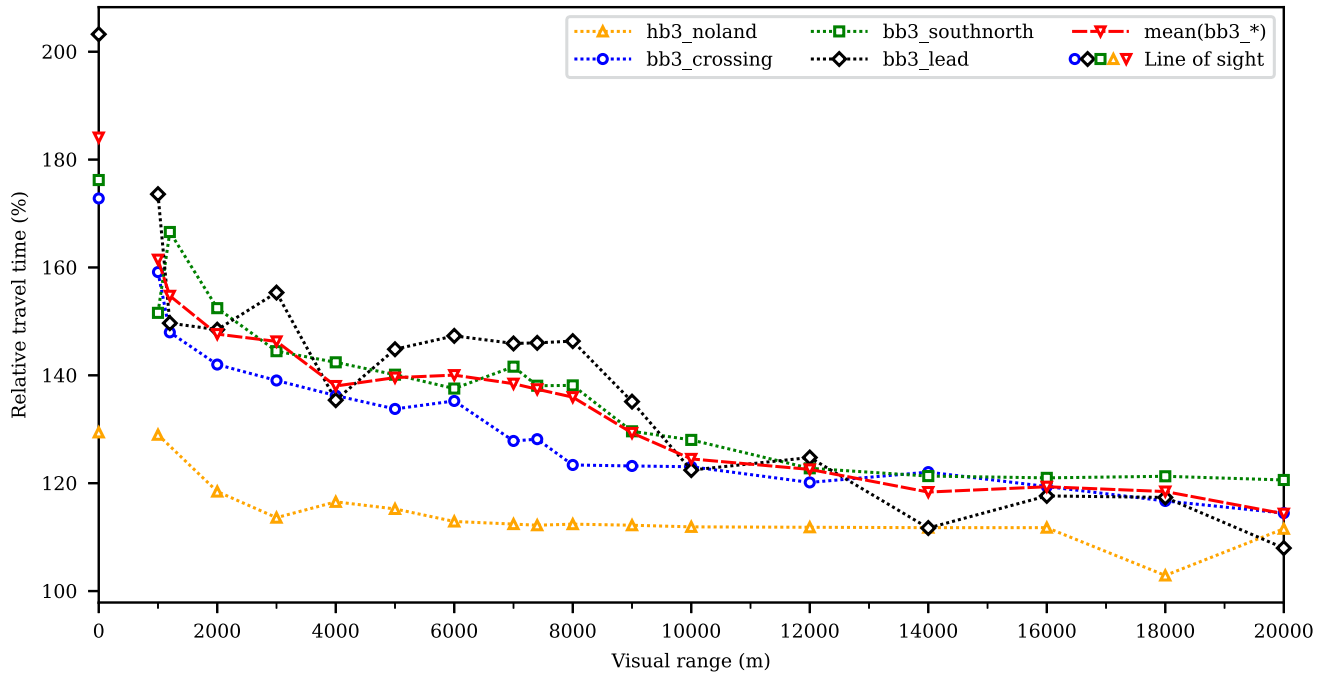

Figure 8. Relative travel time by VR for different scenarios. In total, 100% equals the travel time as calculated with full information available. The red, dashed line is the mean travel time for bb3_lead, bb3_southnorth and bb3_crossing. Travel times were calculated only for VR highlighted by a marker.

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

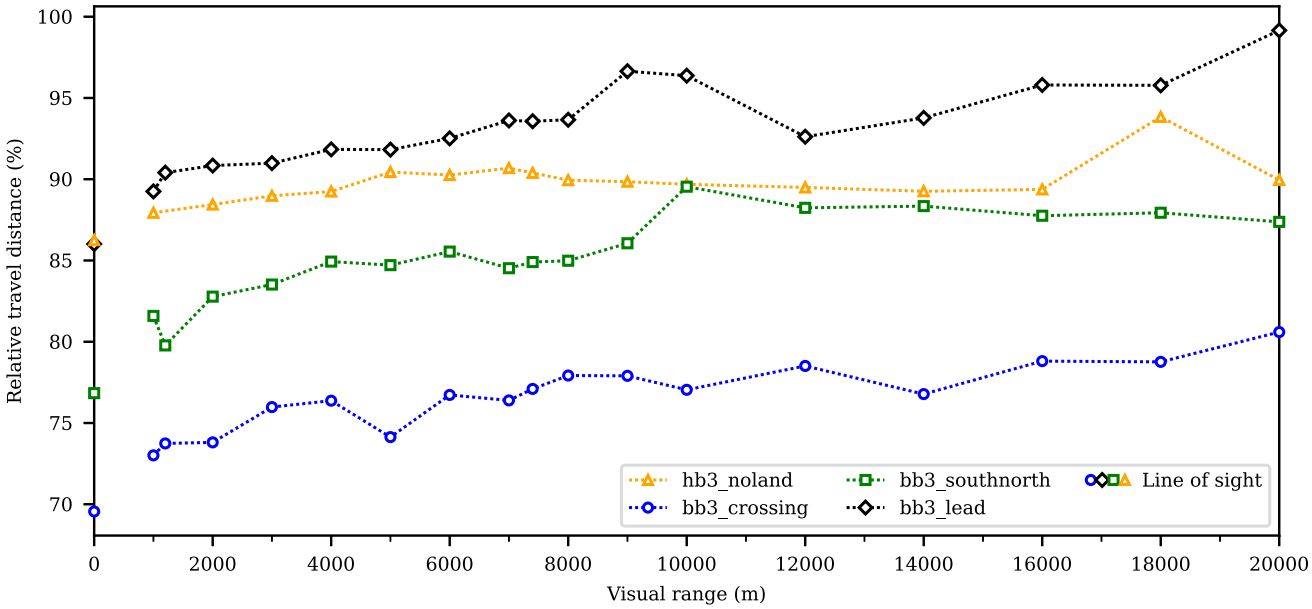

Figure 9. Relative travel distance by VR for different scenarios. In total, 100% equals the travel distance when unlimited information is available at each waypoint. Suggestions created with lower VR in general are shorter than larger VR routes. The length of all limited VR solutions do not exceed the length of an unlimited VR route. The markers at the vertical axis denote the length of the line of sight between origin and target. Travel distances were calculated only for VR highlighted by a marker.

between the 'division waypoint' and the 'reunion waypoint' the 10 km VR solution is ~14% (~2.5 h) faster compared to the 12 km VR solution. Figure 11 is an excerpt of Figure 10 which shows the area at the 'division waypoint'. When being there, for a ship travelling with a VR of 12 km, it looks like it is best to make use of the 'other ice' patch, which starts in the centre of Figure 11 and extends to the east. The 12 km VR solution decides against the open-water patch in the middle of the image because it already knows that this open-water patch does not extend further towards the target. Note that the lead in the south is not visible at that time for both, 10 and 12 km VR. As a ship with a VR of 10 km located at the 'division waypoint' does not know that the central open-water patch does not extend further towards the target, it makes use of it. When arriving there, the lead in the south gets visible (as indicated by the red or white dashed circles). It turns out that it is more expensive to use the 'other ice' patches like the 12 km VR solution does (red arrows) than to use the southern lead (white arrows). The consequences of

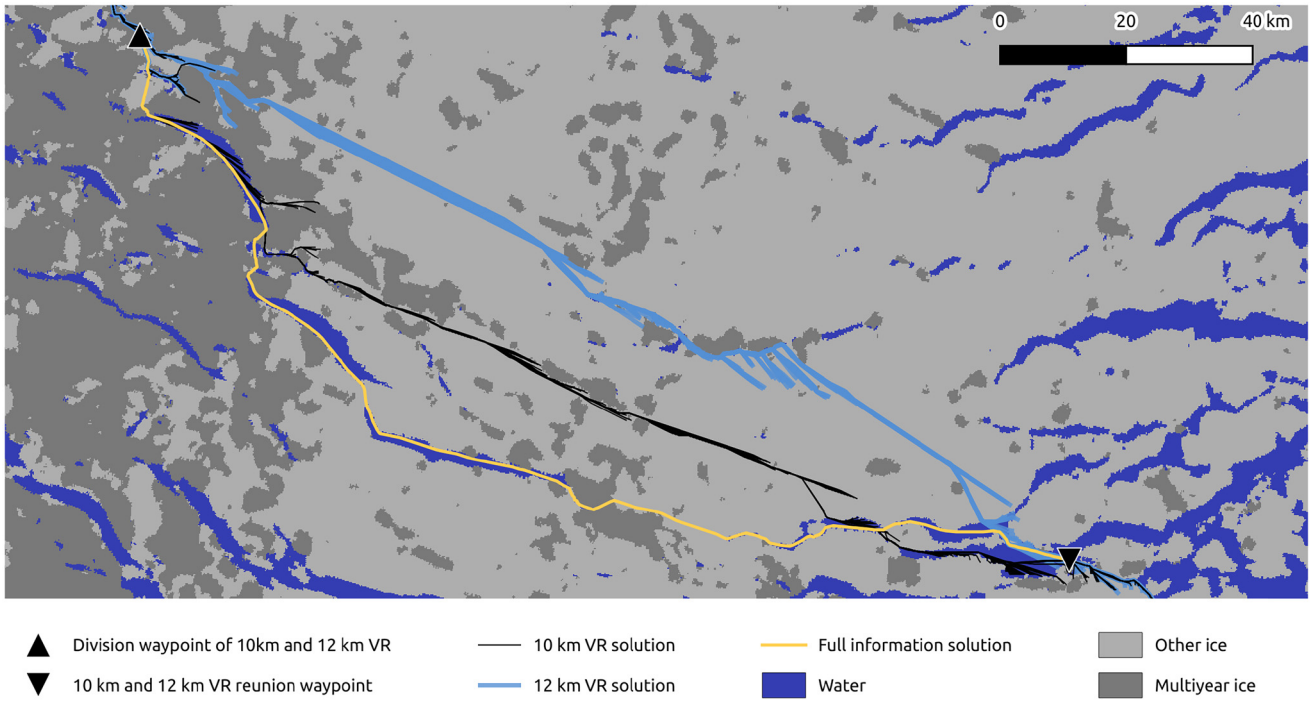

Figure 10. Visualization of the 10 and 12 km limited VR intermediate solutions which run from origin to target (both not within in the image) as defined by the scenario. Between the division and reunion waypoint, both solutions differ completely: the 10 km VR suggestion makes use of some leads, which results in reduced travel times compared to the 12 km VR solution. For comparison, a full information route which runs from the division to the reunion waypoint is displayed. This route utilizes even more leads, which further reduces the travel time.

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

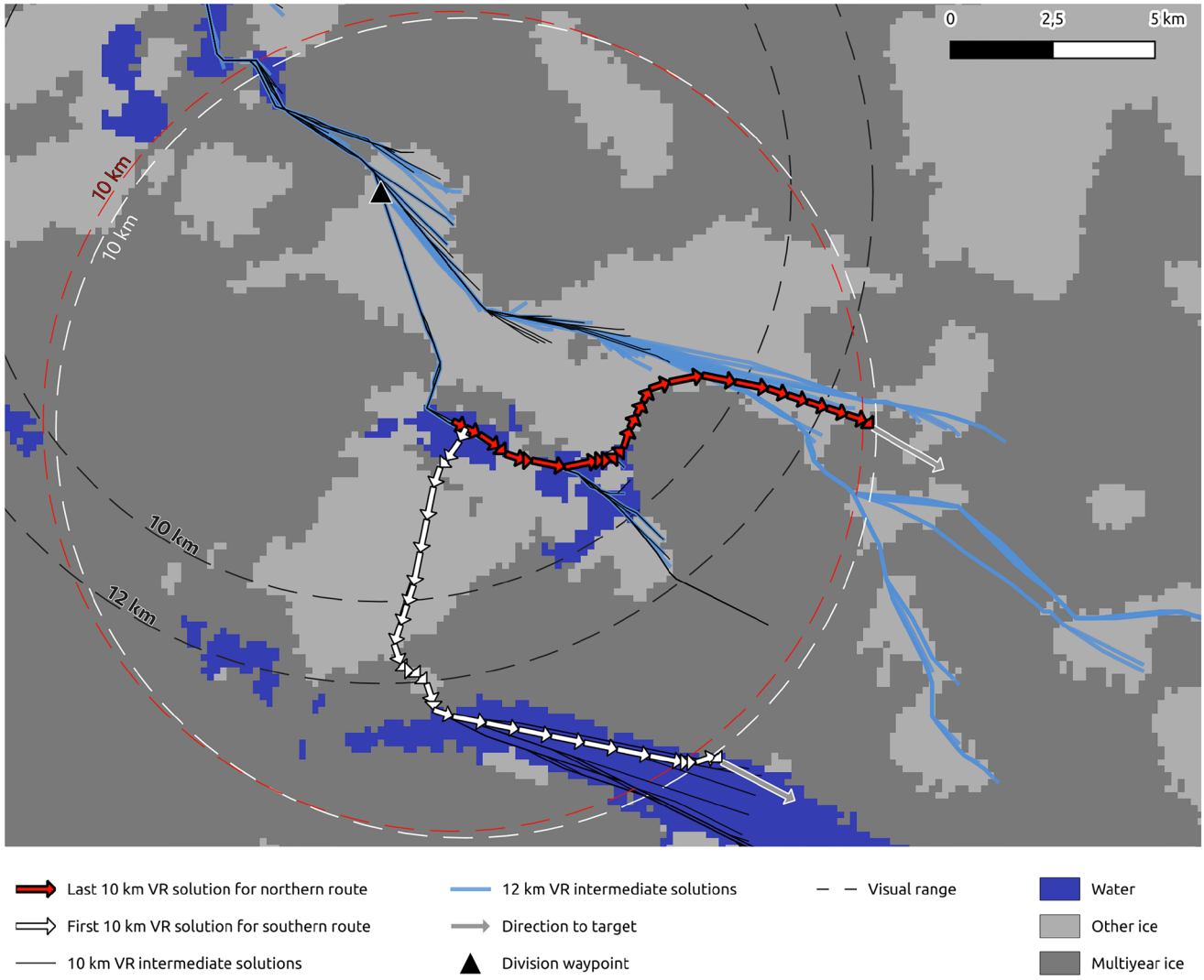

Figure 11. Comparison of excerpts of 10 and 12 km VR solutions that go from north to east or south. It indicates that the 10 km VR solution is not attracted by an 'other ice' area that extends from the image centre to the east. That is because it is not within its visible range (black dashed circles) at the last common waypoint (black triangle). The last 10 km intermediate VR solution which suggests going the northern route is marked with red arrows and the visible area for the corresponding ship position is visualized by the red dashed circle. The first 10 km intermediate VR solution which suggests the southern route is marked by white arrows and the visible area is highlighted by the white circle. After leaving the VRs both solutions are heading to the final target following the line of sight which is indicated by the grey arrows. The blue and black lines visualize other intermediate solutions. These lines show that both solutions were attracted by the water area in the centre of the image first. Interestingly, the 12 km VR solution was never attracted by the large water patch in the south.

not utilizing the central open-water patch are visible in Figure 10: the 10 km VR solution (black) can make use of subsequent leads which are not reachable by the 12 km VR solution as they are too far in the south. Therefore, the 12 km VR solution is certainly best in the area of Figure 11 which is visible to that solution, but is more expensive within the subsequent track (which was not considered when making the decision). One can conclude that the 10 km VR solution found a better solution for the area between the division waypoint and the reunion waypoint by chance due to not knowing ice conditions exterior of its limited VR. A route calculated for an unlimited VR between the division waypoint and the reunion waypoint would, in the beginning, be almost equal to the 10 km VR solution. In addition to the 10 km VR solution, the unlimited VR solution can use additional water patches that are further in the south, which also reduces the travel time. Finally, the travel time of the 10 km VR solution are 132% and the travel time of the 12 km VR solution are 154% of the unlimited VR solution. Travelling along the unlimited VR solution takes 11 h 30 min.

As described in (5) within this study we assume maximum cost c_{max} for travelling areas beyond VR. However, other options for setting the travel cost for the nodes beyond VR are possible. An option which does not make sense is assuming open water beyond VR (c_{min}). This would make the algorithm try to escape the VR as fast as possible, without considering the direction to the target that much. It would neglect extra cost that are created by not travelling in the direction of the target because travelling beyond VR is that cheap. Especially, it would prefer travelling in a suboptimal direction over entering the ice, even in a situation where it would make sense to do so. A more realistic choice could be setting the exterior travel costs for each node to the mean interior costs c_{mean}. This would be like assuming that the ice situation

Table 3. Required time for travelling and length of the routes suggested by the unlimited ARA* algorithm

Scenario	Route length km	Travel time
hb3_noland	326.15	22 h 59 min 36 s
bb3_southnorth	502.65	45 h 34 min 49 s
bb3_lead	479.41	32 h 30 min 44 s
bb3_crossing	595.13	40 h 59 min 2 s

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

continues similar to how the situation is within the VR. If there is lots of open water, that might allow the ship to approach the target less directly. Such a situation could occur at an ice edge, where the ship might travel more parallel to the edge until it finds a better location to enter the ice.

Note that the described methods use objective functions which optimize for time, while ice navigators would consider additional criteria like safety or fuel consumption, too. For now, there is no data on fuel consumption available to the authors, therefore, calculating routes optimized for fuel consumption was not possible. For each ice condition, a velocity was assumed as noted in Table 1. These velocities are rough estimates and will vary for each ship. This situation will improve when using real ship characteristics, which also include for example, information on fuel consumption and velocity per ice type. Safety concerns are considered indirectly only, for example, the traversal of thicker ice, where the ship could get stuck, is punished because of the reduced velocity.

In the presented scenarios we considered static ice conditions to estimate how additional information from satellite images and extended VR can affect travel time. To ultimately provide useful route suggestions to operators on ice going ships additional factors, such as sea-ice drift, have to be considered as well. By application of the ARA* algorithm as implemented in this study, already small-scale changes of the ice condition (e.g. a lead is opening up) would require to calculate a new solution from scratch, as there is no logic to handle changed costs for affected nodes. To prevent this, for example the Anytime Dynamic A* algorithm, which is a combination of ARA* and D* Lite (Likhachev and others, 2005), could be used. However, for large-scale, significant changes of the ice condition this approach might also require a full recalculation of the route (Likhachev and others, 2005). Another possibility is to use a strategy similar to the cooperative A* algorithm and interpret segmented pieces of ice as entities moving in a space–time tensor (Silver, 2005). This solution supports route calculation in view of large-scale changes of the ice conditions but requires that future, time-dependent positions of relevant segments are known before the route calculation begins (or at least before the ship would reach the affected location). Such information could be obtained by using ice drift models. Contrary to the cooperative A* algorithm, occupied positions would not necessarily be classified as obstacles, but as more difficult to navigate depending on the ice condition. Similar to this approach, the ICEPATHFINDER framework (Lehtola and others, 2019) makes use of a speed map, which is updated with drift forecast data at discrete time steps (each covers 6 h). As a consequence, the cost for travelling to a certain node is dependent on time. Another approach that also uses an ice condition map that is updated periodically but utilizes a three-dimensional ant colony algorithm for pathfinding is described by Zhang and others (2022). Further development to support route calculation for dynamic ice conditions is to be considered in future work.

Independent of how reliable a route suggestion might be, navigators on the ground will always have the final call, because they have the advantage of seeing the conditions first hand. As a result, the route a real ship would take may be different to the one provided by an algorithm. However, this is true for both, the limited and the unlimited VR solution.

Conclusion

In this study, we simulated a ship navigating in ice-covered waters without any information on surrounding ice conditions exterior to the VR. This was done by using an ARA* pathfinding algorithm which utilized classified Sentinel 1 data that comes at a resolution of 160 m. Four ship tracks were simulated to meet different ice conditions in the two selected regions, Baffin Bay and Hudson Bay. The simulations are calculated for multiple VRs up to 20 km. Optimal routes that utilize full ice information are calculated for the same scenarios, using the same algorithm. By comparing the results of both approaches, we found that, for all scenarios, the travel times for a certain track decreased for increasing visual range. In contrast to that, the travelled distance increased for increasing VR. We also found that using ice information or an ice assistance system that can suggest routes is most beneficial when sailing in waters with inhomogeneous ice conditions. It was assumed that 2 km is a realistic VR for what navigators can see from the bridge during navigation. It turned out that following the optimal route suggestion for travelling in homogeneous ice conditions, a ship could save 15% of the travel time compared to travelling with 2 km VR. When travelling in areas with more complex, inhomogeneous ice conditions, it could save 32% of the travel time. When assuming that a ship is utilizing an ice radar which enables a VR of 7400 m it could save 10% (homogeneous case) to 27% (inhomogeneous case) of the travel time when following the optimal route compared to travelling with the information from the ice radar.

Acknowledgements. We acknowledge the Federal Ministry for Digital and Transport of Germany for their support via mFUND for the project FAST-CAST 2 (grant No.: 19F2191A).

References

Browne T and 5 others (2022) A method for evaluating operational implications of regulatory constraints on Arctic shipping. *Marine Policy* **135**, 104839. doi: 10.1016/j.marpol.2021.104839

Canadian Coast Guard (2022) Ice navigation in Canadian waters. https://www.ccg-gcc.gc.ca/publications/icebreaking-deglacage/ice-navigation-glaces/page01-eng.html, accessed: 2023-12-13.

Guinness RE and 7 others (2014) A method for ice-aware maritime route optimization. In *2014 IEEE/ION Position, Location and Navigation Symposium – PLANS 2014*, pp. 1371–1378.

Hansen EA and Zhou R (2007) Anytime heuristic search. *Journal of Artificial Intelligence Research* **28**, 267–297. doi: 10.1613/jair.2096

Hart PE, Nilsson NJ and Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* **4**(2), 100–107. doi: 10.1109/TSSC.1968.300136

Kaleschke L and 27 others (2016) Smos sea ice product: operational application and validation in the Barents Sea marginal ice zone. *Remote Sensing of Environment* **180**, 264–273. Special issue: ESA's Soil Moisture and Ocean Salinity Mission – Achievements and Applications. doi: 10.1016/j.rse.2016.03.009

Lehtola V, Montewka J, Goerlandt F, Guinness R and Lensu M (2019) Finding safe and efficient shipping routes in ice-covered waters: a framework and a model. *Cold Regions Science and Technology* **165**, 102795. doi: 10.1016/j.coldregions.2019.102795

Li X and 5 others (2021) Arctic shipping guidance from the CMIP6 ensemble on operational and infrastructural timescales. *Climatic Change* **167**(1), 23. doi: 10.1007/s10584-021-03172-3

Likhachev M, Gordon G and Thrun S (2003) ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *Proceedings of (NeurIPS) Neural Information Processing Systems*, Vol. **16**, pp. 767–774.

Likhachev M, Ferguson D, Gordon G, Stentz A and Thrun S (2005) Anytime Dynamic A*: An Anytime, Replanning Algorithm. In *Proceedings of 15th International Conference on Automated Planning and Scheduling (ICAPS)*.

Melia N, Haines K, Hawkins E and Day JJ (2017) Towards seasonal Arctic shipping route predictions. *Environmental Research Letters* **12**(8), 084005. doi: 10.1088/1748-9326/aa7a60

Murashkin D and Frost A (2021) Arctic Sea ICE Mapping Using Sentinel-1 SAR Scenes with a Convolutional Neural Network. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pp. 5660–5663.

Pohl I (1970) Heuristic search viewed as path finding in a graph. *Artificial Intelligence* **1**(3), 193–204. doi: 10.1016/0004-3702(70)90007-X

Shibata H, Izumiyama K, Tateyama K, Enomoto H and Takahashi S (2013) Sea-ice coverage variability on the Northern Sea routes, 1980–2011. *Annals of Glaciology* **54**(62), 139–148. doi: 10.3189/2013AoG62A123

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press

Silver D (2005) Cooperative pathfinding. In *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'05, 117–122, AAAI Press.

Thayer J and Ruml W (2010) Anytime heuristic search: Frameworks and algorithms. In *Proceedings of the 3rd Annual Symposium on Combinatorial Search, SoCS 2010*, Vol. **1**, pp. 121–128. https://doi.org/10.1609/socs.v1i1.18181

Topaj A, Tarovik O, Bakharev A and Kondratenko A (2019) Optimal ice routing of a ship with icebreaker assistance. *Applied Ocean Research* **86**, 177–187. doi: 10.1016/j.apor.2019.02.021

Tran TT, Browne T, Musharraf M and Veitch B (2023) Pathfinding and optimization for vessels in ice: a literature review. *Cold Regions Science and Technology* **211**, 103876. doi: 10.1016/j.coldregions.2023.103876

Zhang C, Zhang D, Zhang M, Zhang J and Mao W (2022) A three-dimensional ant colony algorithm for multi-objective ice routing of a ship in the Arctic area. *Ocean Engineering* **266**, 113241. doi: 10.1016/j.oceaneng.2022.113241

Zilberstein S (1996) Using anytime algorithms in intelligent systems. *AI Magazine* **17**(3), 73–83. doi: 10.1609/aimag.v17i3.1232

https://doi.org/10.1017/aog.2024.28 Published online by Cambridge University Press