OPTIMAL NAVAL PATH PLANNING IN ICE-COVERED WATERS

(DOI No: 10.3940/rina.ijme.2017.a1.390)

V Aksakalli, School of Science, RMIT University, Melbourne, Australia, **D Oz,** Department of Computer Engineering, Marmara University, Istanbul, Turkey, **A F Alkaya,** Department of Computer Engineering, Marmara University, Istanbul, Turkey, and **V Aydogdu,** Department of Maritime Transportation and Management Engineering, Istanbul Technical University, Istanbul, Turkey

SUMMARY

The Northern Sea Route (NSR) links the Atlantic and Pacific oceans through the Arctic and it is critical for global trade as it provides a route between Asia and Europe that is significantly shorter than the alternatives. NSR is soon expected to open for intercontinental shipping due to global warming and thus presents tremendous opportunities for reductions in shipping time, cost, and environmental impacts. On the other hand, facilitating this route requires innovative approaches due to the navigation risks associated with its ice-covered waters. This study presents a graph-theoretical approach for optimal naval navigation in ice-covered sea routes with flexible turn angles based on the idea of large-adjacency grid graphs. Our model allows for asymmetric left and right turn radii as well as turn speeds that vary as a function of the turn angle and it offers natural-looking navigation paths.

NOMENCLATURE

R	The navigation area
R_x	Length of the navigation area
R_y	Width of the navigation area
P	The set of obstacles indexed by p
B_p	Buffer zone associated with obstacle p
f_p	Safety distance associated with obstacle p
s	Starting vertex
t	Termination vertex
V	Vertex set
v_c	Current vertex
v_p	Previous vertex
v_n	Next vertex
E_k	Edge set for each k
a_r	Vessel's minimum right turn radius distance
a_l	Vessel's minimum left turn radius distance
$h(\alpha)$	Vessel's speed as a function of turn angle α
$d(.,.)$	The Euclidean distance function
$m_r(\alpha)$	Length of a proper leg for right turn with angle α
$m_l(\alpha)$	Length of a proper leg for left turn with angle α
Q_m	Control point m for a cubic Bézier curve
λ	Smoothing parameter for a cubic Bézier curve

1. INTRODUCTION

Sea ice in the Arctic region is decreasing at an accelerating rate due to global warming and feedback of the oceanic and atmospheric circulation changes (Shen & Shi, 2011). Northern Sea Route (NSR) links the Atlantic and Pacific oceans through the Arctic and it is critical for international trade as it provides a route between Europe and Asia that is 9000 km shorter than the Panama Canal route and 17000 km shorter than traveling around Cape Horn, South America (Wilson et al., 2004). With the rapid melting of the Arctic sea ice, NSR is soon expected to open for global trade. However, major challenges exist in utilizing NSR due to the region's unusual characteristics including extreme weather conditions and ice-covered waters. These issues have transformed the problem of Arctic vessel navigation into a challenging notion that calls for innovative strategies (Sahin & Kum, 2015). In fact, Ho (2010) states that *''Before the Arctic routes can reliably be used on a large scale for transit by shipping along its passages, more investments are required on infrastructure and the provision of marine services to ensure the safe and secure transit of shipping''*. Thus, there is a need for new naval path planning approaches that specifically account for such challenges.

A wide variety of path planning problems have been studied in the literature and various efficient algorithms have been developed for different problems, including the well-known Dijkstra's and the A* Algorithms for finding shortest paths. However, in real life applications, these algorithms fundamentally differ with respect to vehicles' operational characteristics. From a path planning point of view, perhaps the most critical operational limits are the ones related to the vehicles' turning dynamics, which are, for instance, of utmost importance in vessel navigation. There also exists a significant amount of literature on deterministic shortest paths with turn constraints in continuous as well as discrete settings. Continuous-space studies typically involve calculus of variation with curvature constraints and complex differential equations. These types of approaches, on the other hand, are generally not very suitable for even modest generalizations such as asymmetric turn constraints or reduced turn speeds nor do they scale well with a large number of obstacles. Existing discrete-space studies, on the other hand, impose somewhat simplistic turn constraints. Current state-of-the-art in discrete-space turn-constrained path planning is the one introduced in (Ari et al., 2013). In that work, the authors study the ship navigation problem with the objective of finding the turn-constrained shortest path between two given points in the presence of obstacles with asymmetric turn radii and decreased turning speeds. Their methodology is based on an 8-adjacency grid discretization of the navigation area and

on the idea of vertex replication where immediate navigation history is embedded in each vertex copy. A major limitation of the methodology presented therein is that the navigation is limited to 45-degree turns on the grid. Thus, even though the methodology presented therein is optimal with respect to the 8-adjacency grid discretization, it is likely to be sub-optimal in practice and lead to unrealistic trajectories.

Our goal in this study is to tackle the optimal maritime path planning problem in the presence of ice blocks with respect to safety distance and flexible turn-radius constraints in order to find the shortest feasible naval path between given two coordinates. In particular, we improve upon methodology of Ari et al. (2013) and overcome the stated limitation by allowing for flexible turn angles using the idea of large-adjacency grids (LAGs). Such grids are generalization of the 8-adjacency grids to 8k-adjacency grids where k is a positive integer, such as 16-, 24-, or 32-adjacency grids for $k=2,3,4$ respectively. Due to their adjacency structure, LAGs allow for higher degrees of flexibility in modelling of turn constraints. In particular, larger k values allow for even more flexibility, though at the cost of increased computational burden. Our approach is specifically designed for resolution-independent discretization of the navigation area and it is optimal with respect to the underlying LAG discretization. Our methodology also allows for asymmetric minimum left and right turn radii as well as turn speeds that vary as a function of the turn angle.

Implementation of our methodology requires non-trivial changes to the underlying LAG in order to preserve optimality. Specifically, our approach entails construction of a turn-constrained LAG (TC-LAG) graph where a TC-LAG vertex is defined as a three-tuple of the following: (1) a LAG vertex denoting the physical location of the TC-LAG vertex, (2) the LAG vertex at which the most recent turn was made, and (3) the angle of this turn. During the TC-LAG construction, we deploy a first-in first-out vertex queue, inserting the starting vertex first. For each de-queued vertex, we define all appropriate adjacency structures that do not violate turn constraints and then add encountered vertices to the queue. This process of queuing/ de-queuing is continued until the vertex queue is empty. Once the TC-LAG is constructed, we simply execute Dijkstra's Algorithm on it to find the shortest path between the given starting and termination vertices. Our final step is to smoothen the resulting path via Bézier splines composed of cubic Bézier curves for a more natural looking navigation path (Bartels et al., 1998). We demonstrate our methodology on a ship navigation example where we compare turn-constrained and unconstrained paths for different k values. We also present a proof-of-concept in a full-mission ship-handling simulator where we compare the smoothened optimal path against the actual ship path inside the simulator.

The rest of this manuscript is organized as follows: Section 2 presents a literature review and Section 3 formally defines the turn-constrained path planning problem. Section 4 describes our navigation methodology in detail including the LAG discretization, construction of TC-LAGs, and smoothening of the optimal path. Section 5 presents our vessel navigation example and Section 6 provides the full-mission ship handling simulator application. Our summary and conclusions are presented in Section 7.

2. LITERATURE REVIEW

Numerous studies exist in the literature on shortest path problems with turn constraints. Even though vehicle navigation is inherently a continuous-space problem in general, previous studies on this topic have focused on discretization of the navigation area due to the difficulty of incorporating realistic operational constraints in the continuous setting (Fagerholt et al. 2000; Lee et al. 2002). Incorporation of turn constraints in a continuous environment typically requires nonlinear manoeuvring equations and it is difficult in general (Hilten and Wolkenfelt, 2000). For the most part, turn radius constraints in discrete settings are limited to symmetric one-edge ahead constraints in the literature. Such a limitation requires that resolution of the navigation area be determined by the turn-radius, which is clearly not very practical. These symmetric one-edge ahead turn constraints were modelled in several different ways including (1) vertex replication (Albiach et al., 2008), (2) modification of the Dijkstra's Algorithm (Solka et al., 1995; Gutierrez & Medaglia, 2008), and (3) transformation of the original graph (Mico & Soler, 2011; Pugliese and Guerriero, 2012).

Another closely related research area is turn-constrained path planning for military aircraft and unmanned aerial vehicles (Zabarankin et al., 2006; Edison & Shima, 2011; Oz et al., 2013). Such path planning studies, however, are not readily adaptable to general path planning problems due to the fact that aerial mission planners often take into account fuel storage constraints, which fundamentally changes the structure of the underlying problem and makes it computationally intractable (Royset et al., 2009).

A recent study by Babel & Zimmermann (2015) considered computation of feasible short paths for ships in the presence of disk-shaped mines where ship movement is modelled via curves whose curvature change linearly with the curves length. Their methodology is based on random generation of waypoints along with associated random directions and random paths combining them, after which a feasibility check is conducted and the shortest feasible path with respect to this graph is computed. The closest work to ours is that of Ari et al. (2013) where the authors proposed a novel vertex replication based optimal method on an 8-adjacency grid representation of the navigation area for turn-constrained ship navigation. A significant limitation of their approach, however, is that it only allows for 45-degree turns. In our application, we

overcome this limitation via utilization of appropriately defined turn-constrained large-adjacency grid graphs. We are not aware of any previous studies using large-adjacency grid graphs in turn-constrained path planning.

3. THE TURN-CONSTRAINED PATH PLANNING PROBLEM

This section formally defines the turn-constrained path planning problem in the presence of obstacles, or the TC-PP Problem in short. Without loss of generality, we assume a rectangular navigation area and polygon-shaped obstacles. It should be noted that any geometric shape can be represented by a polygon approximation at any level of accuracy (Wu & Leou, 1993). Thus, these two terms, i.e., polygons and obstacles, shall be used interchangeably in this work. We assume the existence of buffer zones around each polygon due to safety considerations since the risk factors outside the vessels which, to a great extent, are attributable to floating obstacles, atmospheric effects and ice, are of outmost importance among the risk factors associated with arctic navigation (Sahin & Kum, 2015). We compute the buffer zones using the buffer zone calculation algorithm in Ari et al. (2013).

For any obstacle $p \in P$, we define its buffer zone B_p as the region whose boundary consists of points that are f_p units away from the closest point on the boundary of p. The vessel is not allowed to enter the buffer zone of any obstacle at any time. Observe that the vessel's turning angle at any point (with respect to its present direction) is upper bounded by the curvature of the circle with respective left and right turn radii. We assume that the vessel's speed depends on the turn angle α where $h(0)$ denotes the vessel's straight navigation speed. The turn speed function concept is explained in detail in Section 4.3.

The *TC-PP Problem* is then defined as follows: Given a set of obstacles P inside the navigation area R and the associated obstacle buffer zones B_p, a starting vertex s and a termination vertex t, find the shortest s,t path for an vessel with speed function $h(\alpha)$, minimum left-turn radius a_l, and minimum right-turn radius a_r.

4. METHODOLOGY

In this section, we first define large-adjacency grids (LAGs), then we describe construction of turn-constrained LAGs and we conclude this section by illustrating how an optimal path can be smoothed in order to emulate the actual navigation of the vessel using Bézier splines.

4.1 LARGE-ADJACENCY GRIDS

Due to the previously-mentioned challenges associated with continuous-space environments, we consider a discrete approximation of the navigation area that is a generalization of the regular 8-adjacency grids as LAGs. In particular, all LAGs are directed graphs in the form of $LAG_k=(V,E_k)$ for $k=1,2,3,\ldots$ such that V consists of all pairs of integers i,j with $1 \le i \le R_x$ and $1 \le j \le R_y$. Here, R_x and R_y are positive integers denoting the length and width of the navigation area respectively. Parameterization of LAGs is achieved by expansion of their adjacency structures via a generalization parameter denoted by k, which we call the "connectedness degree". We call an 8k-adjacency grid as "k-connected LAG" which is defined as the directed graph with vertex set V and edge set E_k such that, for each vertex $v \in V$, there are directed edges between v and all other vertices that are exactly $\sqrt{(i^2 + 1)}$ units away from v for $i=0,1,2,\ldots$. Figure 1 illustrates the adjacency structure for a vertex in $1,2,3$, and 4-connected LAGs respectively.

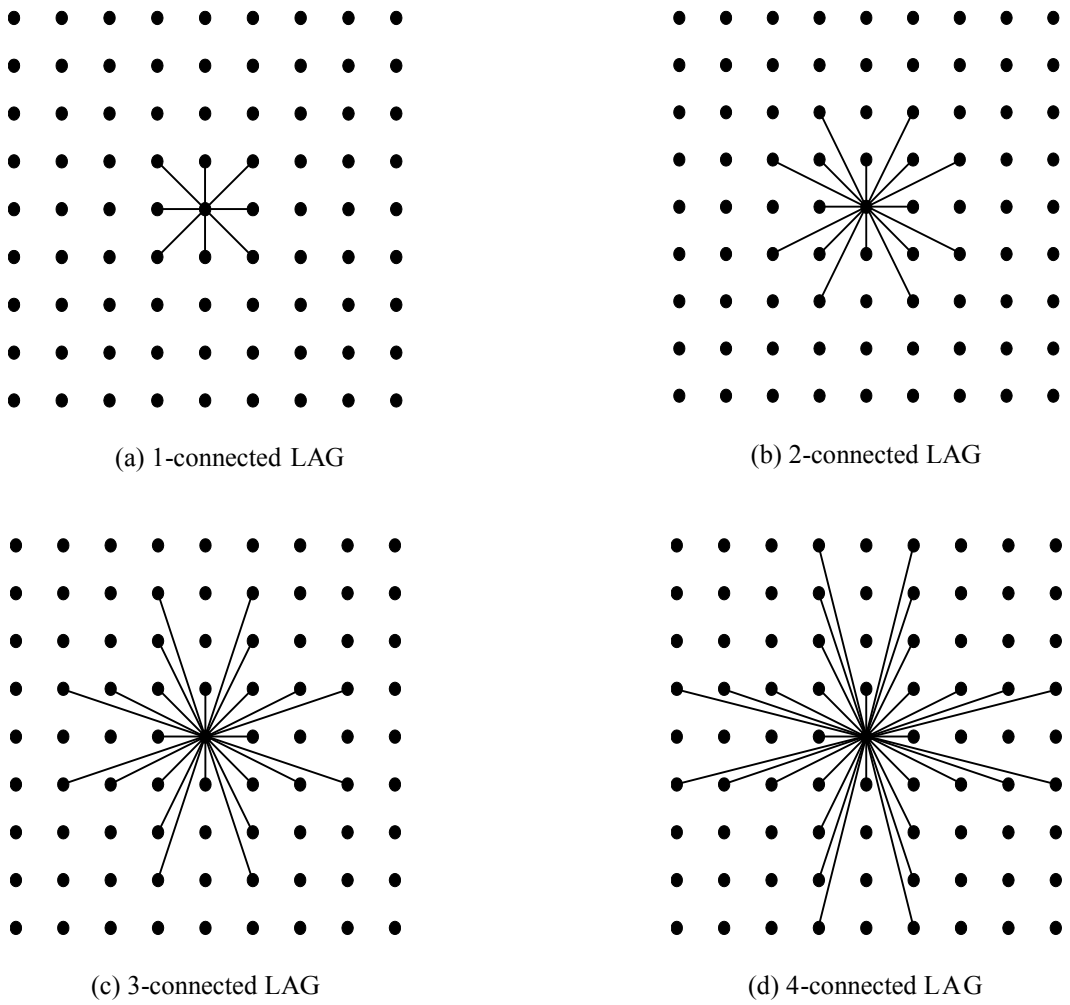
As an example, a 1-connected LAG is the 8-adjacency grid $LAG_1=(V,E_1)$ where there are directed edges between all pairs of the following four types of vertices:

- (i,j) and $(i+1,j)$ with unit length,
- (i,j) and $(i,j+1)$ with unit length,
- (i,j) and $(i+1,j+1)$ with length $\sqrt{2}$ units, and
- $(i+1,j)$ and $(i,j+1)$ with length $\sqrt{2}$ units.

A 2-connected LAG, on the other hand, is the 16-adjacency grid $LAG_2=(V,E_2)$ where E_2 is equal to E_1 augmented by directed edges between all pairs of the following four types of vertices with lengths of $\sqrt{5}$ units:

- (i,j) and $(i+2,j+1)$,
- (i,j) and $(i+1,j+2)$,
- $(i,j+1)$ and $(i+2,j)$, and
- $(i,j+2)$ and $(i+1,j)$.

Note that the permissible turn angles in a LAG are determined by its connectedness degree. For instance, 1-connected LAGs only allow for 45° turns. Permissible angles for other LAGs can be found using simple geometry. For instance, 2-connected LAGs allow for $\arctan(1/2) \approx 26.6°$ turns (with respect to straight navigation path) in addition to 45°. Likewise, 3-connected LAGs allow for $\arctan(1/3) \approx 18.4°$ turns (in addition to 26.6° and 45°), whereas 4-connected LAGs additionally allow for $\arctan(1/4) \approx 14°$ turns. Observe that each increase in k will roughly allow for an additional $(45/k)°$ turn. As an example, assuming that the vessel is traversing on a straight navigation path, a 6-connected LAG allows for $9.5°$, $11.3°$, $14°$, $18.4°$, $26.6°$ and $45°$ turns. For other incoming directions, the vessel can turn with an angle that is any pairwise difference of these 6 angles. For instance, if the vessel is traversing an edge with a 9.5° angle, it can make a $11.3\text{-}9.5 = 1.8°$ turn. For computational efficiency, we do not explicitly consider turns larger than 45° as such turns can be expressed as a sequence of smaller angle turns.



(a) 1-connected LAG

(b) 2-connected LAG

(c) 3-connected LAG

(d) 4-connected LAG

Figure 1: Illustration of the adjacency structure in 1, 2, 3, and 4-connected LAGs respectively

Clearly, higher values of k will allow for even more flexible turns, but this will also result in many more edges being defined, and consequently increase computational time requirements. Once a k value is decided and the navigation area is discretized via a k-connected LAG, one vertex in the graph LAG_k is designated as the starting vertex s and another is designated as the termination vertex t. The vessel is to traverse from s to t in LAG_k while avoiding any edges intersecting with buffer zones subject to minimum turn radius constraints.

4.2 CONSTRUCTION OF TURN-CONSTRAINED LAGS

Our implementation of turn constraints entails construction of what we call a turn-constrained LAG (TC-LAG), which is essentially another graph constructed from the underlying LAG that only allows for feasible course alterations with respect to the turn constraints. Once the TC-LAG is constructed, we execute Dijkstra's Algorithm to find the shortest s,t path. During the TC-LAG construction process, we pay particular attention to define adjacency structures in a manner that all feasible traversals are embedded and no unfeasible moves are permitted. Thus, our methodology is guaranteed to find the turn-constrained optimal path with respect to the underlying LAG discretization.

The idea in construction of TC-LAG is that vertices of this new graph embed the immediate navigation history so that the next feasible moves can be determined. The reason for this idea is that where the vessel can feasibly traverse next depends on where the vessel is coming from. Specifically, the immediate navigation history we track is the vertex corresponding to the most recent turn and the angle of that turn. Given the graph LAG_k, we denote the corresponding turn-constrained graph as $LAG'_k=(V',E'_k)$. The vertices $v' \in V'$ are defined as three-tuples in the form of $<v_c,v_p,\alpha>$ where $v_c,v_p \in V$ and $\alpha \in [0°,45°]$ is the angle of the most recent turn. The subscripts "c" and "p" stand for "current" and "previous" respectively. In particular, the first vertex v_c in the tuple denotes the physical location of the TC-LAG vertex. The second vertex v_p denotes the physical location of the most recent turn prior to the vessel's arrival at v_c. The vertex v_p is used to keep track of how far the vessel has travelled since the last turn. Note that

the LAG vertices v_c and v_p are not necessarily adjacent in the underlying LAG graph. As described below, the distance between v_c and v_p and the turn angle α are used to identify which moves are feasible from v_c with respect to the turn constraints.

When constructing the TC-LAG graph LAG'_k, we deploy a first-in first out TC-LAG vertex queue and use a regular LAG graph LAG_k as a guide graph. The construction algorithm starts with an empty queue and inserts the TC-LAG vertex $<s,null,0>$ into this queue where s is the starting LAG vertex. At each iteration of the algorithm, a vertex $v' = <v_c,v_p,\alpha>$ is pulled from the queue and processed as follows:

Step 1. Using the guide LAG graph, all feasible LAG neighbours of the physical location vertex v_c are identified based on the immediate traversal history as specified by v_p and α, and new TC-LAG vertices corresponding to these neighbors are created.

Step 2. These newly created TC-LAG vertices are added to the queue.

Step 3. These vertices as well as the corresponding TC-LAG edges with appropriate time lengths are added to V'_k and E'_k respectively.

The algorithm continues with the queuing/ dequeuing process described above until the vertex queue is empty. This process specifically ensures that all possible feasible paths from start to termination are embedded in the TC-LAG and no illegal moves are allowed.

In Step 1 above where new TC-LAG vertices $v'=<v_c,v_p,\alpha>$ are created, suppose that a candidate LAG neighbor of v_c is denoted by v_n and the turn angle at v_c in the direction of v_n is denoted by α_n. Here, the subscript "n" stands for "next". Provided that v_n is a feasible neighbor, the TC-LAG vertex $v'_n := <v_n,v_c,\alpha_n>$ is created and the edge (v',v'_n) is added to E'.

There are three possibilities for a feasible move: a straight traversal and turns in the same and opposite directions as the previous turn respectively. Figure 2 illustrates these three possibilities. In Figure 2(a), the neighbor v_n represents a straight traversal from v_c and therefore no feasibility check is necessary.

In Figure 2(b), v_n represents a turn in the opposite direction as the previous turn. The feasibility check in this case is carried out as follows: we first observe that the edge (v_p,v_c) is the second leg of a turn with angle α. The minimum distance that needs to be travelled so that v_n becomes a feasible move can be calculated using simple geometry as shown in Figure 3. In this particular case where the previous turn was a right turn, we define *the minimum right-turn distance function* as follows:

$$m_r(\alpha) := \frac{a_r}{\tan\left(\frac{180-\alpha}{2}\right)} \qquad \text{(eq.1)}$$

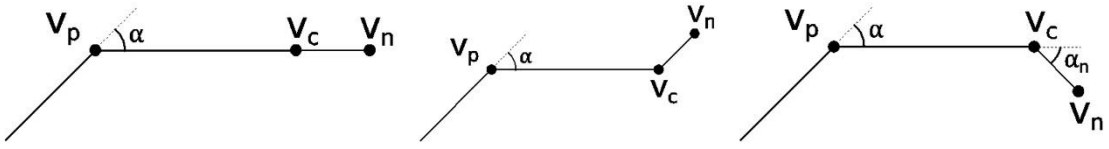
Similarly, for left turns, $m_l(\alpha)$ is defined as:

$$m_l(\alpha) := \frac{a_l}{\tan\left(\frac{180-\alpha}{2}\right)} \qquad \text{(eq.2)}$$

Thus, v_n is a feasible neighbor of v_c only if:

$$d(v_p, v_c) \geq m_r(\alpha) + m_l(\alpha_n) \qquad \text{(eq.3)}$$

where d denotes the Euclidean distance function.



(a) Straight traversal (b) Opposite direction (c) same direction

Figure 2: Illustration of the three feasible moves following a right turn.

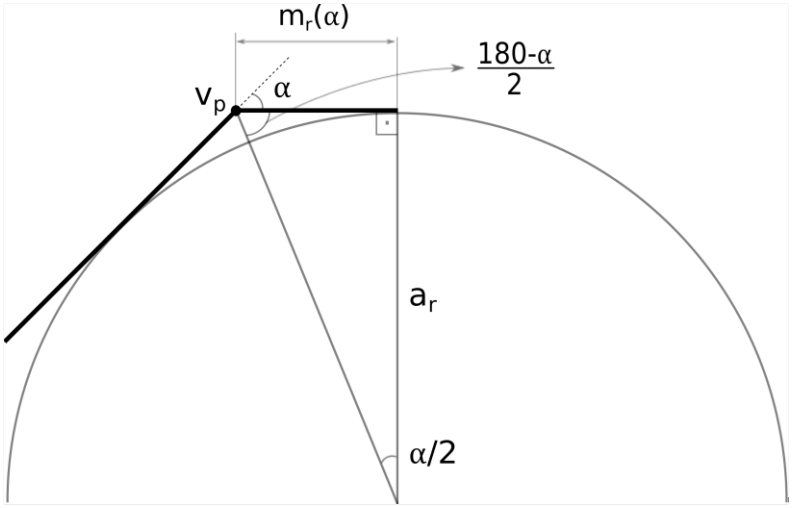


Figure 3: Calculation of the minimum right-turn distance.

Next, in Figure 2(c), v_n denotes a right turn in the same direction as the previous one. In order to check for feasibility, we first note that the edge (v_p,v_c) is the second leg of a turn with angle α, and it is also the first leg of a turn with angle α_n. The vertex v_n is a feasible neighbour only if the distance between v_p and v_c is at least the sum of the respective minimum leg lengths. Hence, the feasibility condition is given by:

$$d(v_p, v_c) \geq m_r(\alpha) + m_r(\alpha_n) \qquad \text{(eq.4)}$$

Feasibility conditions involving left turns can be defined in a similar manner using $m_l()$ instead of $m_r()$ above.

4.3 CALCULATION OF TC-LAG EDGE TIME LENGTHS

In general, a vessel's speed decreases during course alterations. We therefore consider a general case where the vessel's speed is a function of the vessel's turn angle. The speed function shall be denoted by $h(\alpha)$ where α is the angle of the vessel's turn. Here, $h(0)$ is the speed for straight (non-turn) traversal. In case the vessel's speed is constant during the entire traversal, $h(\alpha) := h(0)$ for $\alpha \geq 0°$. For decreased turn speeds, $h(\alpha)$ specifically denotes the vessel's minimum instantaneous turn speed in between the turn legs as described below.

For a given turn angle α, we first note that the turn is achieved in two legs on the LAG graph. We call a leg proper if its Euclidean length is respectively $m_r(\alpha)$ for a right turn and $m_l(\alpha)$ for a left turn. Our modeling of the turn speeds is illustrated in Figure 4 for a right turn. The line segment (p_1,v) corresponds to the first proper leg and the line segment (v, p_2) corresponds to the second proper leg where v is a LAG vertex, yet p_1 or p_2 is not necessarily so. Observe that $d(p_1,v) = d(v, p_2) = m_r(\alpha)$. Upon the vessel's arrival at the point p_1, its speed is $h(0)$. Along the first proper leg, that is, on the line segment (p_1, v), the vessel's speed decreases from $h(0)$ all the way down to $h(\alpha)$. Along the second leg, i.e., on the line segment (v, p_2), the vessel starts gaining speed, and by the time it arrives at p_2, the vessel reaches its original speed of $h(0)$. For convenience and simplicity, we assume that the vessel's average speed along both the first and second proper legs is denoted by $h_{leg}(\alpha)$ and calculated as:

$$h_{leg}(\alpha) := \frac{h(0)+h(\alpha)}{2} \qquad \text{(eq.5)}$$

We make the observation that a vessel loses about one fifth of its speed after a $45°$ (American Bureau of Shipping, 2006). Hence, we use the following model for $h(\alpha)$ in our vessel navigation example:

$$h(\alpha) := h(0)\left(1-\left(\frac{1}{5}\right)\left(\frac{\alpha}{45^o}\right)\right), \quad 0 \leq \alpha \leq 45^o \qquad \text{(eq.6)}$$

It should be noted that turn speeds and rates of acceleration and deceleration change considerably from one vessel to another. Consequently, the above model is likely to require some fine-tuning in different applications.

Time lengths of TC-LAG edges are defined based on the speed model described above. In particular, time lengths of edge segments corresponding to proper legs are calculated with respect to the average speed $h_{leg}(\alpha)$ in Equation 5 and time lengths of other edge segments are calculated with respect to the speed $h(0)$.

4.4 SMOOTHENING OF THE OPTIMAL PATH

Once the time-wise shortest s,t path is found on the TC-LAG graph using Dijkstra's Algorithm, the final step is to smoothen the path for a more natural look. For the smoothening process, we use classical Bézier splines composed of cubic Bézier curves. Bézier curves are commonly used in computer graphics, font engines, and 2D/ 3D digital animation to smoothen curves (Bartels et

al., 1998). We remark that once smoothened, the shortest *s*,*t* path may intersect a buffer zone, albeit slightly. Should this be the case, a possible provision would be an appropriate increase in the safety distance for the associated ice block and rerunning of the algorithm.

An *m*-th order Bézier curve is defined by a set of *m+1* control points. Thus, a cubic Bézier curve is specified by four control points, which we denote by $Q_0,\ldots,Q_3$. The first and last control points are the end points of the curve. A Bézier curve is always contained in the convex hull of its control points. On the other hand, the intermediate control points only provide directional information and typically do not lie on the curve. A cubic Bézier curve is parameterized as follows:

$$B(\lambda) = (1-\lambda)^3 Q_0 + 3(1-\lambda)^2 \lambda Q_1 + 3(1-\lambda)\lambda^2 Q_2 + \lambda^3 Q_3, \quad \lambda \in [0.1] \qquad \text{(eq.7)}$$

Optimal path is denoted by the sequence of vertices $<s=v_0,v_1,\ldots,v_{m-1},t=v_m>$. In order to smoothen this path using cubic Bézier curves, we need four points on the path for each curve, which we choose as follows:

- For the first Bézier curve, we set $Q_0 = s$, $Q_1 = v_1$, $Q_2 = v_2$, $Q_3 = v_{2,3}$ where $v_{i,j}$ is defined as the midpoint of the edge (v_i,v_j) for $1 \le i,j \le m$.
- For the last Bézier curve, if m is odd, we set $Q_0 = v_{m-3,m-2}$, $Q_1 = v_{m-2}$, $Q_2 = v_{m-1}$, $Q_3 = t$. If *m* is even, we set $Q_0 = v_{m-2,m-1}$, $Q_1 = v_{m-1}$, $Q_2 = v_{m-1,t}$, $Q_3 = t$.
- For Bézier curves in the middle, we set $Q_0 = v_{k,k+1}$, $Q_1 = v_{k+1}$, $Q_2 = v_{k+2}$, $Q_3 = v_{k+2,k+3}$ where $2 \le k \le n-4$.

Once these curves are constructed, they are concatenated to form one continuous Bézier spline representing the optimal path the vessel is to navigate.

5. VESSEL NAVIGATION APPLICATION

We now illustrate how our methodology can be used to find optimal paths for vessels navigating in ice-covered waters. In our example, we consider a *5×3* km navigation area with unit edge length of 100 meters, i.e., $R_x = 50$ and $R_y = 30$. We use a standard 100,000 tones merchant vessel with a cruising speed of *k=10* knots, a right-turn radius of 545 meters, and a left-turn radius of 475 meters (American Bureau of Shipping, 2006). We consider an ice concentration of about *20%*. In order to generate a random ice field with this concentration, we do the following: (1) generate random points inside the navigation area, (2) construct the corresponding Voronoi tiles representing ice blocks, and then (3) shrink them by *80%*. For each ice block, we set the buffer zone safety distance to 120 meters. Figure 5 depicts the navigation area filled with polygon-shaped ice blocks with buffer zones around these blocks.

We conduct two sets of computational experiments within the navigation area described above: (1) unconstrained shortest paths on regular LAGs, and (2) turn-constrained shortest paths on TC-LAGs. We consider *k* values ranging from 1 to 6. We break down the computational time for each *k* into two: time required for constructing the underlying LAG/ TC-LAG graph and the time required for the Dijkstra run. For each experiment, we visually illustrate the paths found for different *k* values and also tabulate execution times, path lengths, and graph sizes. Our goal here is to gain insight into impact of the *k* value on execution time and solution quality in both constrained and unconstrained cases. These experiments were run on a PC with 32 GB memory and an Intel Core i7 processor with a 3.8 GHz clock speed.

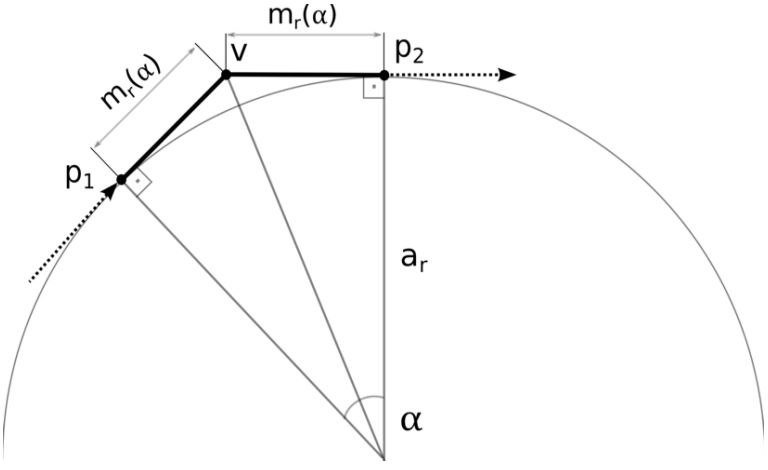


Figure 4: Illustration of proper legs for a right turn. The line segments (p_1,v) and (v,p_2) are proper right turn legs and both have a Euclidean length of $m_r(\alpha)$.

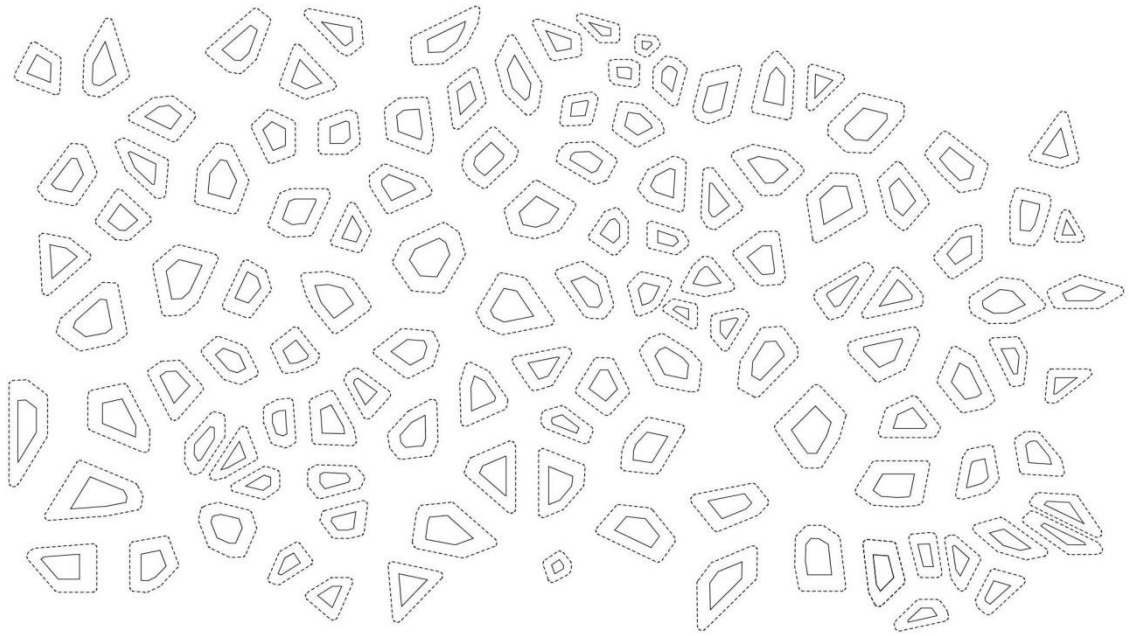


Figure 5: The navigation area filled with polygon-shaped ice blocks with buffer zones around them in the ice navigation example.

Table 1: Comparison of results for unconstrained and turn-constrained shortest paths obtained by Dijksra's Algorithm on LAG and TC-LAG graphs, respectively, for different *k* values.

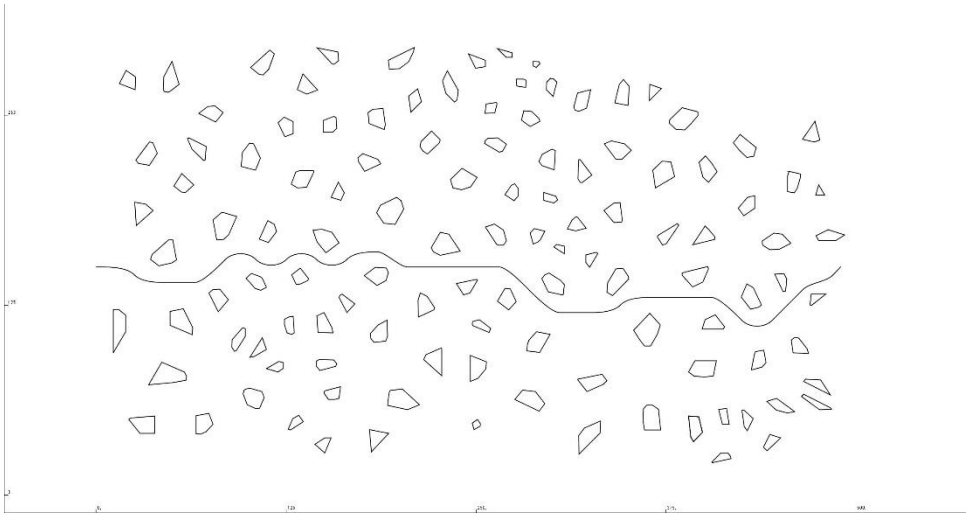
Unconstrained Dijkstra on a LAG graph						**Turn-constrained Dijkstra on a TC-LAG graph**					
k	Graph Constr. (sec)	Dijkstra (sec)	Path Length (min)	No. Vertices	No. Edges	k	Graph Constr. (sec)	Dijkstra (sec)	Path Length (min)	No. Vertices	No. Edges
1	3.44	0.22	18.56	1,500	11,524	1	170.47	2.35	19.36	264,685	596,961
2	6.52	0.28	17.42	1,500	22,580	2	997.12	14.37	17.75	918,794	3,441,841
3	9.49	0.31	16.85	1,500	33,324	3	2,923.45	46.17	17.40	1,751,689	9,672,551
4	11.99	0.36	16.47	1,500	43,736	4	5,038.52	84.90	16.81	2,322,264	16,295,633
5	15.23	0.40	16.47	1,500	53,876	5	6,971.08	124.07	16.81	2,669,858	22,119,898
6	17.55	0.42	16.47	1,500	63,684	6	10,883.26	187.25	16.81	3,343,571	31,830,312

5.1 UNCONSTRAINED SHORTEST PATHS ON REGULAR LAGS

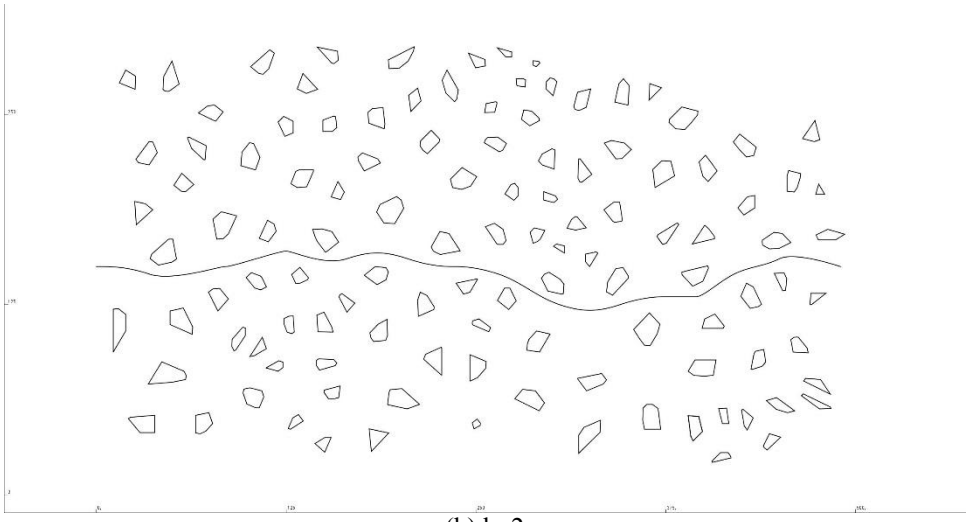
We first run Dijkstra's Algorithm on regular LAG graphs to obtain unconstrained shortest paths, which are illustrated in Figure 6 for *k=1,3,4,5,6* respectively. Our purpose with this exercise is to illustrate that regardless of turn constraints, LAG discretization can result in shorter and more realistic paths. Computational results for this set of experiments are included in Table 1. In the table, optimal path lengths are given in minutes. The last two columns show the number of LAG vertices and edges for the respective *k* values. As can be seen in the table, optimal path lengths do not change for $k \geq 4$, yet both computational times slightly increase with *k*, illustrating the fact that LAG graphs offer a significant degree of flexibility in balancing computational burden against solution quality. A visual inspection of Figure 6 reveals that higher *k* values result in not only shorter, but also more natural paths, particularly avoiding the unpleasant zigzagging pattern seen in the case of *k=1* that is common in the literature.

5.2 TURN-CONSTRAINED SHORTEST PATHS ON TC-LAGS

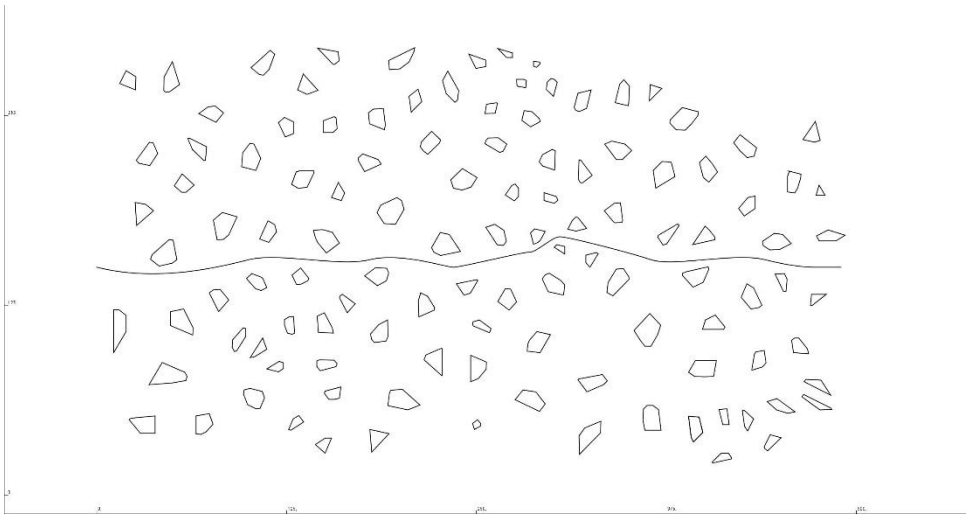
Next, we run Dijkstra's Algorithm on *k-connected* TC-LAG graphs in order to obtain turn-constrained shortest paths, as illustrated in Figure 7 for *k=1,2,4,5,6* respectively. As can be seen in this figure, increasing *k* results in shorter, more realistic, yet still feasible paths. Table 1 includes the computational results for this set of experiments. We observe that path lengths improve progressively up to *k=4*, after which increasing *k* does not yield any additional benefits. In particular, the case here for *k=1* that only allows *45°* turns corresponds to the methodology of Ari et al. (2013) for the same problem. In this particular case, allowing for additional turn angles with a 4-connected LAG discretization results in a *15%* decrease in path length.



(a) k=1

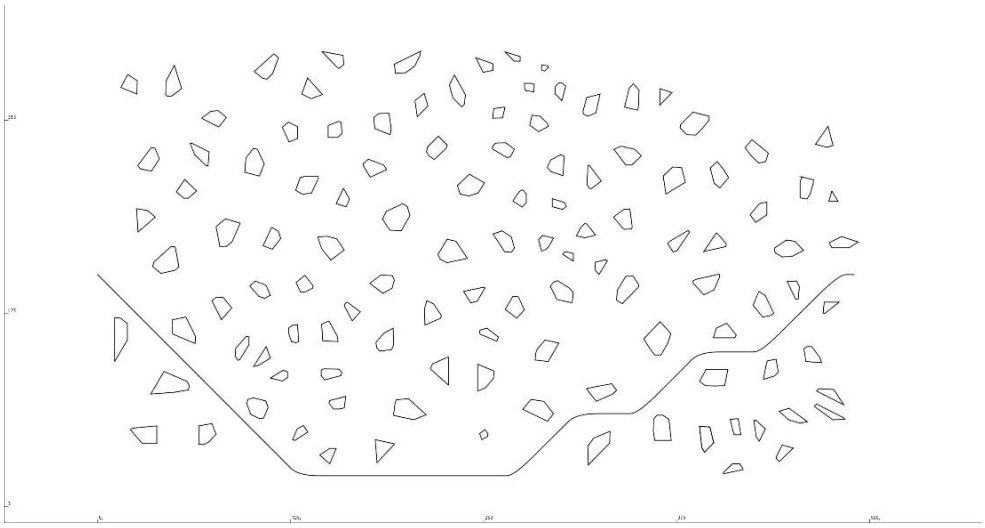


(b) k=2

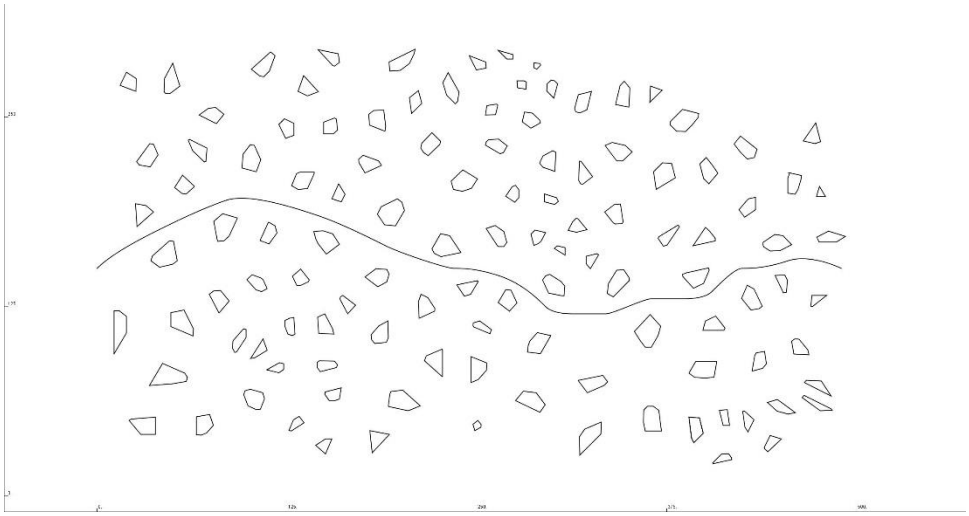


(c) k=4,5,6

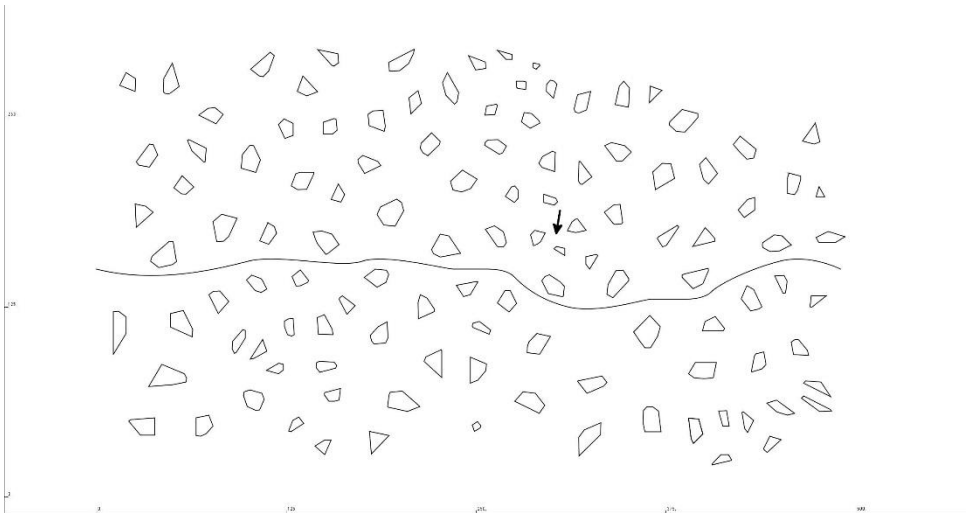
Figure 6: Smoothened unconstrained shortest paths on regular LAG graphs for different values of *k*.



(a) k=1



(b) k=2



(c) k=4,5,6

Figure 7: Smoothened turn-constrained shortest paths on TC-LAG graphs for different values of k. For $k \geq 4$, while the unconstrained shortest path passes through the region between the three ice blocks shown by the arrow, it is too much of a sharp turn for the vessel, and therefore it is avoided in the turn-constrained shortest path.

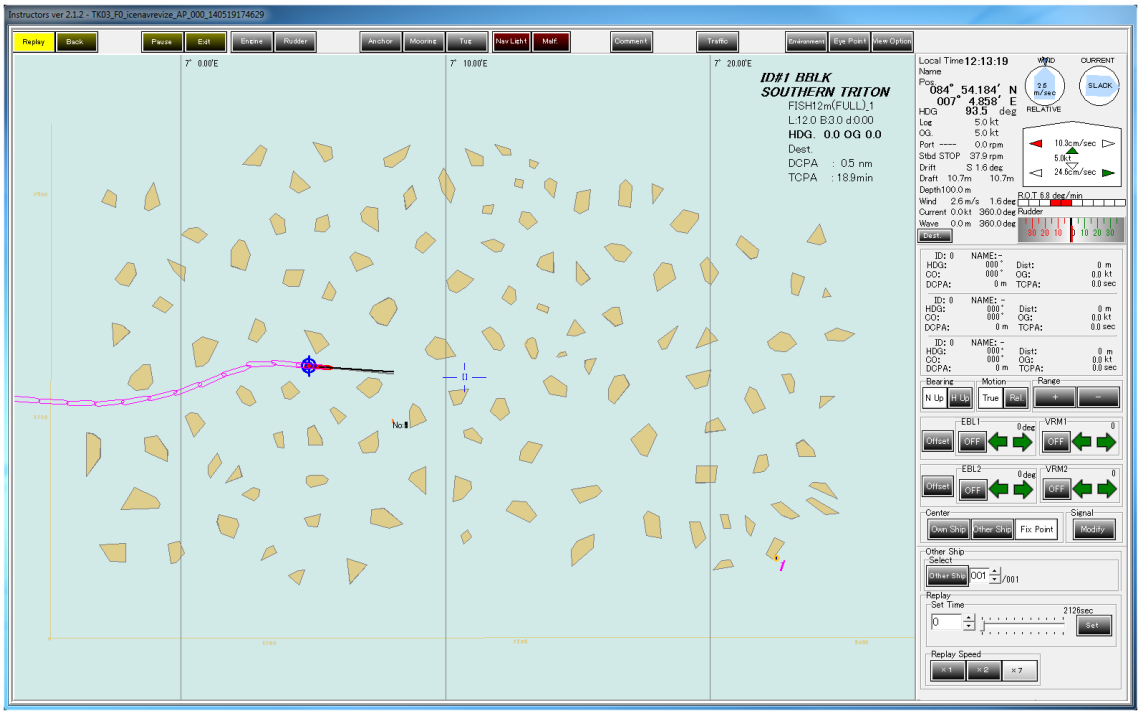


Figure 8: The navigation area inside the simulator console and progress of the ship's navigation.

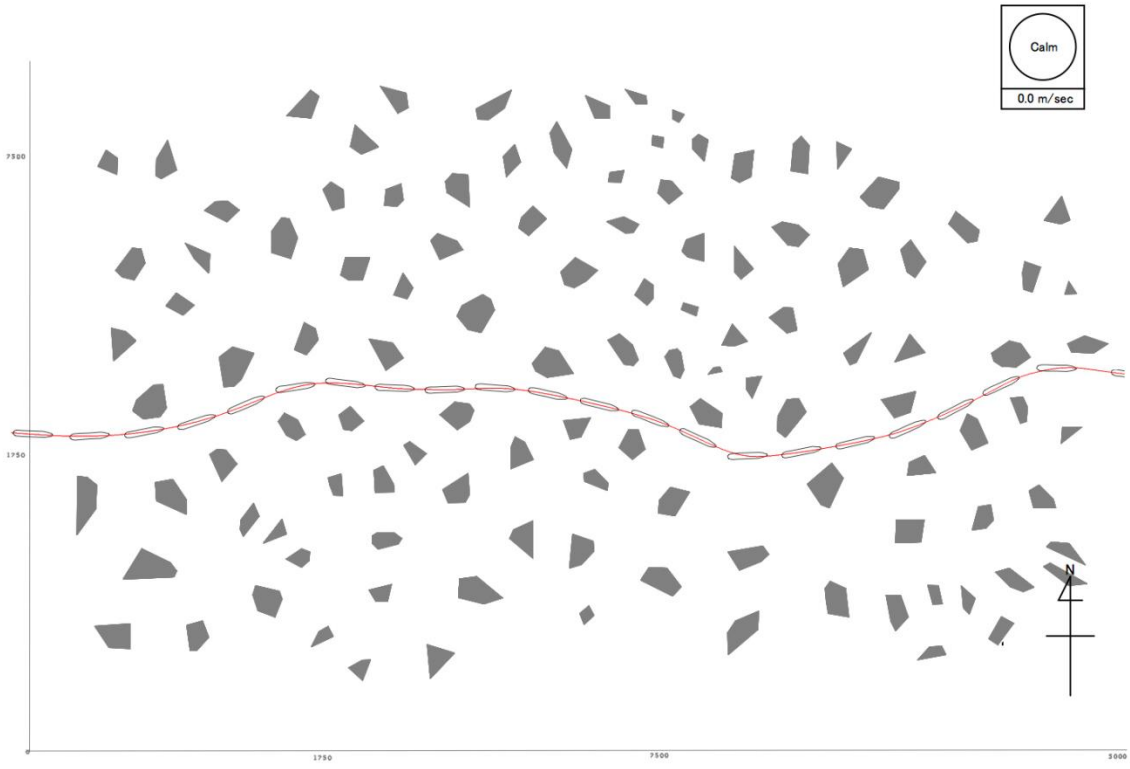


Figure 9: The complete manually navigated path inside the simulator.

6. SIMULATOR APPLICATION

Similar to aircrafts, automated navigation systems are available for ships for navigation in open waters. However, common practice in restricted waterways is to switch to hand-steering mode due to the vessel's limited maneuvering capability. Thus, in a real-world application, the optimal path obtained above in our ice navigation example needs to be traversed manually by a qualified helmsman. In this section, we present a proof-of-concept by having a licensed captain manually steer the optimal path in a full-mission vessel handling

simulator (FMSHS). Our purpose with this simulation is to illustrate real-world feasibility of our methodology. The FMSHS we use is a Japanese Marine Science branded one that is capable of fully simulating various vessel maneuvers in restricted waterways.

Figure 8 shows navigation area inside the simulator console and progress of the vessel's navigation. The complete manually navigated path inside the simulator is shown in Figure 9. We observe that the manual path follows the smoothened graph-theoretical path very closely, suggesting that the optimal graph-theoretical path in our example is consistent with the complex maneuvering dynamics of the vessel under consideration.

7. SUMMARY AND CONCLUSIONS

In this study, we utilize large-adjacency grid (LAG) graphs to solve the problem of vessel navigation in ice-covered waters with turn-constrains. Our approach allows for considerably higher degrees of flexibility regarding the turn angles leading to practical and optimal paths that can be undertaken by ships in reality in order to navigate through ice in Arctic waters. We demonstrate that regardless of turn constraints, particularly after a Bézier spline smoothening, a LAG discretization offers short, practical and natural-looking navigation paths. We validate our methodology on a vessel navigation example in ice-covered waters and we present a proof-of-concept in a full-mission vessel handling simulator application.

Regarding implementation of turn constraints, we first construct a new graph based on a regular LAG that embeds all feasible moves and prohibits any illegal ones, which we call a TC-LAG. Once constructed, we run Dijkstra's Algorithm on the TC-LAG to find the optimal turn-constrained path. Our modeling of turn constraints allows for resolution-independent discretization of the navigation area and it is optimal with respect to the underlying LAG discretization. This model also allows for asymmetric minimum left and right turn radii as well as reduced turn speeds as a function of the turn angle. In our limited computational experiments, we observed that k=4 seemed to be sufficient for practical purposes. Increasing k beyond 4 resulted in relatively higher run times with little to no additional benefits in solution quality.

As can be seen in our vessel navigation example, the number of vertices and average degree of the graph in TC-LAGs can be rather high, resulting in increased run times for the Dijkstra's Algorithm. As a remedy to this problem, future research may consider the following: (1) use fast heuristics-based searches such as A* instead of Dijkstra's Algorithm, or (2) run two sequential searches: the first search being in the LAG (using A* and perhaps Euclidean distance as the heuristic function), and the next search being in the TC-LAG with the heuristic function being derived from the costs obtained in the LAG search.

In this work, environmental forces such as winds, waves, or sea currents were assumed to be non-existent. Nonetheless, such forces often play a crucial role in vessel navigation. Future research might also investigate modeling and incorporation of such forces in the model.

8. ACKNOWLEDGEMENTS

This study was supported by TUBITAK (The Scientific and Technological Research Council of Turkey) (Grant Number 113M489). Earlier work of V. Aksakalli was conducted while in the Department of Industrial Engineering at Istanbul Sehir University.

9. REFERENCES

1. ALBIACH, J., SANCHIS, J.M., SOLER, D. (2008). *An asymmetric TSP with time windows and with time-dependent travel times and costs: an exact solution through a graph transformation*. European Journal of Operational Research, 189, 789–802.
2. ARI, I., AKSAKALLI, V., AYDOGDU, V., KUM, S. (2013). *Optimal ship navigation with safety distance and realistic turn constraints*. European Journal of Operational Research, 229, 707–717.
3. BABEL, L., ZIMMERMANN, T. (2015). *Planning safe navigation routes through mined waters*. European Journal of Operational Research, 241, 99–108.
4. BARTELS, R.H., BEATTY, J.C., BARSKY, B.A. (1998). *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, San Francisco, CA.
5. DIJKSTRA, E.W. (1959). *A note on two problems in connection with graphs*. Numerische Mathematik, 1, 269–271.
6. EDISON, E., SHIMA, T. (2011). *Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms*. Computers & Operations Research, 38, 340–356.
7. FAGERHOLT, K., HEIMDAL, S., LOKTU, A. (2000). *Shortest path in the presence of obstacles: an application to ocean shipping*. Journal of the Operational Research Society, 51, 683–688.
8. GUTIERREZ, E., MEDAGLIA, A.L. (2008). *Labeling algorithm for the shortest path problem with turn prohibitions with application to large-scale road networks*. Annals of Operations Research, 157, 169–182.
9. HILTEN, M.J.V., WOLKENFELT, P.H.M. (2000). *The rate of turn required for geographically fixed turns: A formula and fast-time simulations*. Journal of Navigation, 53, 146–155.

10. HO, J. (2010). *The implications of Arctic sea ice decline on shipping*. Marine Policy, 34, 713–715.
11. LEE, H., KONG, G., KIM, S., KIM, C., LEE, J. (2002). *Optimum ship routing and its implementation on the Web*. Lecture Notes in Computer Science, 125–136.
12. American Bureau of Shipping, (2006). Guide for vessel maneuverability. *Houston, TX*.
13. MICO, J.C., SOLER, D. (2011). *The capacitated general windy routing problem with turn penalties*. Operations Research Letters, 39, 265–271.
14. OZ, I., TOPCUOGLU, H.R., ERMIS, M. (2013). *A meta-heuristic based three-dimensional path planning environment for unmanned aerial vehicles*. Simulation, 89, 903–920.
15. PUGLIESE, L.D.P., GUERRIERO, F. (2012). *Shortest path problem with forbidden paths: the elementary version*. European Journal of Operational Research, 227, 254—267.
16. ROYSET, J.O., CARLYLE, W.M., WOOD, R.K. (2009). *Routing military aircraft with a constrained shortest-path algorithm*. Military Operations Research, 14, 31–52.
17. SAHIN, B., KUM, S. (2015). *Risk assessment of Arctic navigation by using improved fuzzy-ahp approach*. International Journal of Maritime Engineering, 157-A4, 241–250.
18. SHEN, C., SHI, W. (2011). *Review of climate change in the Arctic*. Energy Procedia, 2466–2473.
19. SOLER, D., MARTINEZ, E., MICO, J.C. (2007). *A transformation for the mixed general routing problem with turn penalties*. The Journal of the Operational Research Society, 59, 540–547.
20. SOLKA, J.L., PERRY, J.C., POELLINGER, B.R., ROGERS, G.W. (1995). *Fast computation of optimal paths using a parallel Dijkstra's algorithm with embedded constraints*. Neurocomputing, 8, 195–212.
21. WILSON, K.J., FALKINGHAM, J., MELLING, H., ABREU, R.D. (2004). *Shipping in the Canadian Arctic: other possible climate change scenarios*. In: Proceedings of the International Geoscience and Remote Sensing Symposium.
22. WU, J.S., LEOU, J.J. (1993). *New polygonal approximation schemes for object shape representation*. Pattern Recognition, 26, 471–484.
23. ZABARANKIN, M., URYASEV, S., MURPHEY, R. (2006). *Aircraft routing under the risk of detection*. Naval Research Logistics, 53, 728–747.

