2024 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
AUTONOMY, ARTIFICIAL INTELLIGENCE, & ROBOTICS TECHNICAL SESSION
AUGUST 13-15, 2024 - NOVI, MICHIGAN

RISK-ADAPTIVE RENDEZVOUS PLANNING FOR RESUPPLY MISSIONS IN THE BATTLEFIELD

Thomas Jonsson Damgaard[1], **Mikael Rittri, PhD**[1], **Patrick Franz**[1]

[1]Carmenta Geospatial Technologies

ABSTRACT

Resupply missions are critical logistical parts of modern warfare. Supply vehicles carrying fuel and ammunition are high-value targets meaning that the route chosen to approach such a mission is sensitive to risk and a critical time of delivery. We address the problem of a supply vehicle that needs to find a secure path to link up with a mobile frontline unit that has a fixed known itinerary. This paper presents a resupply path planning algorithm, the Adaptive Intercepting Path Planning (AIPP) algorithm, that balances risk and travel time to find the most suitable rendezvous point among several. The algorithm generates the least risky route that meets the rendezvous deadline.

Citation: T. Jonsson Damgaard, M. Rittri, P. Franz "Risk-Adaptive Rendezvous Planning for Resupply Missions in the Battlefield," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2024.

1. INTRODUCTION

Combat effectiveness relies on a combination of precision, fire-power a nd l ogistical support from resupplying units. Since the beginning of the Russo-Ukrainian war in February 2022, Russian troops have unleashed an relentless barrage of artillery fire against Ukrainians defenders, with the aim of breaking through or stalling counteroffensives. The Russian capacity to produce and resupply their troops with ammunition has taken NATO by surprise [1]. NATO Secretary-General Jens Stoltenberg has characterized the war in Ukraine as "a battle for ammunition."

In response to Russian aggression, Ukrainian military is utilizing ISR and lethal drones, and Western modern weapon systems [2], to strike and damage Russian ammunition depots located deep within Ukraine territory under Russian control [3]. Targeting and eliminating these depots is an effective combat strategy for disturbing the enemy's supply lines and resource management. Additionally, Ukrainian forces have intensified their efforts to target Russian resupply vehicles [4]. Both ammunition depots and resupply vehicles are critical

targets as they pose significant risks to the delivery of essential resources to the combat troops.

Producing large quantities of ammunition is one challenge, but effectively distributing it to the troops engaged in combat presents a separate critical challenge altogether. Furthermore, modern weapons platforms, include mobile artillery systems, introduce complexities in both supplying new shells and identifying suitable rendezvous points, while ensuring combat effectiveness and readiness. Should a supply mission fail, the weapon system becomes ineffective in combat leaving the frontline troops vulnerable to enemy attacks.

The core objective with this paper is to develop a path planning adapting algorithm that produces the best possible routes to meet up with a moving frontline unit to resupply various necessities. The adaptiveness of the algorithm lies in its ability to balance risk and travel speed, ensuring timely arrival at the frontline units. The algorithm takes into account a set of risk zones, of various kind, to minimize risk exposure while trying to find a punctual path to link up with the moving frontline units.

Assuming the moving frontline units follow a predetermined route (a known combat mission), with specified rendezvous points and associated arrival time of arrival along its route. For clarity, we shall refer to this known route as the itinerary of the unit, while the routes of the supply vehicle shall be called paths to fit the terminology of graph theory. Note that these paths need not be physical roads or trails, just any routes on-road or off-road that the supply vehicle can use. It is crucial to note that the start and end times of the itinerary delineate the time window within which the supply vehicle must rendezvous with the frontline units. This paper will outline the methodology for developing such an adaptive path planning algorithm, considering the dynamics of the frontline units' movements, risk considerations, and the imperative of timely resupply.

2. BACKGROUND

Resupply missions are critical in modern conflicts such as the Russo-Ukrainian war. Modern weapon platforms are highly mobile, resupply vehicles need to link up with moving weapon systems in hazardous zones to deliver crucial and needed supplies. Should a weapon system be left without resupplies of essentials like ammunition or fuel, its combat effectiveness is compromised, leaving the crew vulnerable to therats.

The problem at hand can be precisely formulated within the framework of graph theory. Since we need to consider both travel time and risk, we have a multi-objective optimization problem, but we will use scalarization to reduce it to a single-objective optimization problem. Since such scalarization involves a weighted sum with weights that are hard to choose well, a common approach is to let the algorithm try many different settings for the weights and then ask a human expert to choose manually between the alternative solutions.

In our scenario, where each rendezvous has a deadline, we can categorize alternative supply paths into punctual and tardy ones. By discarding the tardy paths, we can automatically identify the least risky punctual path. It is an advantage to include many alternative rendezvous points since they make it easier to find a low-risk punctual path, and pathfinding to multiple goal points can be done nearly as efficiently as to a single goal. As mentioned in the introduction section, we assume that the frontline unit has a fixed itinerary that gives its future positions in a time interval in which the supply vehicle must link up with it.

Our objective is to identify the least risky punctual path for the supply vehicle to reach a rendezvous position on the fixed itinerary. This itinerary remain unchanged because the frontline units' route take precedence, given its assumed greater importance. While adjusting the itinerary to optimize the resupply mission in terms of speed or risk reduction could be beneficial, such consideration

lie though beyond the scope of this paper but could be a target for a future research paper.

2.1 Problem Statement

A vechicle is driving on an initial path P_I to a link up location R_1, a rendezvous point. Their mission starts at a timestamp c_{start} The rendezvous point R_1 has a georeferenced coordinate position and a deadline for arrival c_1. The path P_I balances travel time t_I and accumulated risk r_I, a set of risk areas has been defined in the operative region. By following path P_I, the vehicle will arrive at rendezvous point R_1 at time $c_{\text{start}} + t_I$ to resupply the moving frontline unit.

Unfortunately, the vehicle has lost time due to unforeseen conditions, resulting in P_I no longer being a feasible solution, the vehicle will not link up the moving frontline unit.

This paper's solution, the AIPP (Adaptive Interception Path Planning) algorithm, will recalculate a new solution, the path P_A, using the supply vehicle's current location and new starting timestamp c_0. The moving frontline unit has a known itinerary providing us a set of n possible rendezvous points $\{R_1, R_2, ..., R_n\}$, where R_1 is the initial rendezvous point.

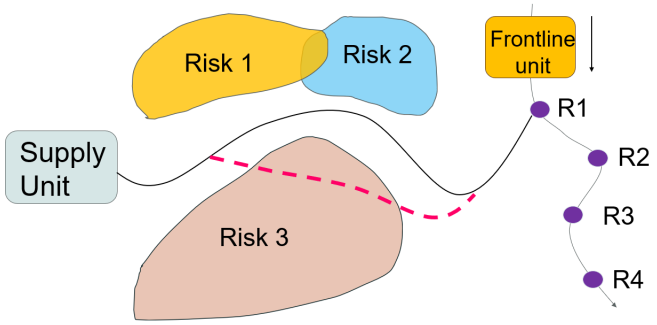

Figure 1: *If the black path to R1 is not punctual, the AIPP algorithm can find an alternative route (the red dotted line) by cutting into a risk zone.*

Figure 1 illustrates a supply vehicle intercepting a mobile frontline unit to re-supply ammunition. The frontline unit has a set of rendezvous points R_1, R_2, R_3 and R_4. The original solution in black reaches the point R_1 at the original time estimate, although, due to unforeseen road conditions, the supply vehicle is off track to intercept on time. The AIPP algorithm finds an alternative path, dashed red line, that enters a risk area to cut corners and reach the final destination on schedule to link up with the moving frontline unit.

The solution P_A will minimize total accumulated risk while still reaching a rendezvous point punctually. The results can be presented visually with alternative paths to all rendezvous points it can reach in time but only the one with least total amount of risk taken will be considered the winner by the algorithm.

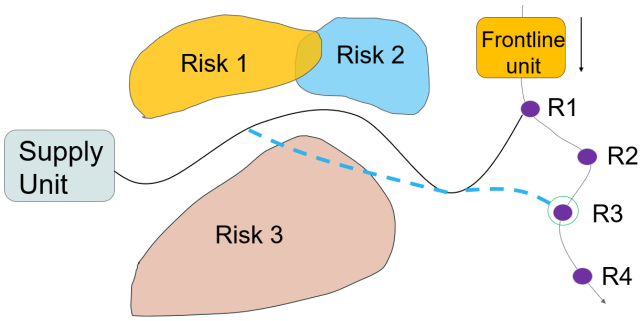

Figure 2: *Here, the AIPP algorithm has found the safest path by choosing the most suitable rendezvous point.*

Figure 2 illustrates a supply vehicle intercepting a mobile frontline unit to re-supply ammunition. Based on figure 1, however, this algorithm optimizes the alternative (blue dashed) path to select the earliest rendezvous point (R_3) with the least accumulated risk. This solution takes the least amount of risk and will link up with the moving frontline unit before it moves beyond the last possible rendezvous point.

3 THEORY

3.1 Graph Theory

In graph theory [5], a graph G is commonly written as $G = (V, E)$, where V represents the

vertices (also known as nodes) and E represents the edges (also known as links). A vertex is defined as a position inside of a graph. An edge is defined as a pair of vertices. In this paper we use directed graphs, where the edges are ordered pairs of vertices.

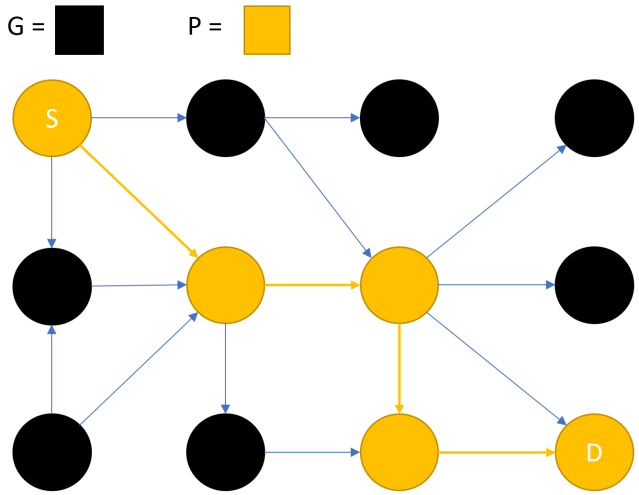

Figure 3: *A graph G in black and a path P from vertex S (start) to vertex D (destination) in yellow.*

A path (or a route) in a graph is a sequence of vertices, $P = (v_1, v_2, ..., v_n) \in V \times V \times ... \times V$ such that (v_i, v_{i+1}) is an edge of the graph for all $i < n$. A *path planning optimization problem* is a problem where we seek one or more routes and where we prefer routes with a lower *cost* as defined by an *objective function*. This is commonly known as the shortest path problem [6], but we will use the more general term least-cost path, to avoid confusion with concrete geometric distances. The objective function for the least-cost path problem is defined as,

$$\sum_{1 \leq i < n} f\left(v_i, v_{i+1}\right) \tag{1}$$

This objective function sums the cost of traversing each edge in the sequence P from v_1 to v_n where f gives the cost value measured in arbitrary units (freely defined). Negative costs make sense in certain applications but we will assume non-negative costs.

3.2 Solving Multi-Objective Optimization

A *multi-objective optimization* problem is a problem where we want to find a solution x that minimizes multiple objective functions f_i over a set $\mathbb{X}$ of possible solutions, where $i = 1, 2, ..., k$ for some $k \geq 2$.

$$\min_{x \in \mathbb{X}} \left(f_1(x), f_2(x), ..., f_k(x)\right) \tag{2}$$

Multi-objective problems are difficult since each objective function is a separate dimension. It is often impossible to find a solution that minimizes each objective function individually, and there is no linear ordering of solutions. However, one can construct a linear ordering by so-called *scalarization*. This paper uses two different scalarization methods, linear scalarization [7] and the ϵ-constraint method [8].

The linear scalarization method creates a weighted sum of the set of objective functions to reduce it to a one-dimensional problem,

$$\min_{x \in \mathbb{X}} \sum_i w_i f_i(x),$$

where $i = 1, 2, ..., k$ and the w_i are weights. The weights are often normalized to make their sum 1 but that is not necessary; we will use a different convention where w_1 is always 1.

The ϵ-constraint method optimizes a selected objective function f_j by forcing every other objective function to be less than or equal to a specific boundary value ϵ_i,

$$\min_{x \in \mathbb{X}} f_j(x)$$
$$\text{such that } f_i(x) \leq \epsilon_i \text{ for } i = 1, 2, ..., k;\ i \neq j.$$

That is: one accepts a solution x only if it satisfies all the ϵ constraints, and such solutions are linearly ordered by their values of $f_j(x)$.

3.3 Moving Frontline Unit

The supply vehicle is linking up with a moving frontline unit to provide much needed supplies, such as ammunition and fuel. In this paper, it is assumed that the frontline unit has a known fixed itinerary, something along the lines of a combat mission, in which we know approximately where the frontline unit will be at a specific time. We assume there are a total of n rendezvous points each with one timestamp c_i with $i = 1, 2, ..., n$. The mission is assumed to start at a timestamp c_{start}. If no rendezvous point is reached in time then the resupply mission has failed. Any path that reaches the rendezvous point in time is defined as a punctual path.

Given a set of punctual paths, the optimal path is defined as the path that has the least accumulated risk.

3.4 Risks

The travel time is not our only concern, since paths in a battlefield are risky and we want to minimize risk, too. So, we use two separate functions that give two costs for an edge from v_i to v_{i+1}:

- $t(v_i, v_{i+1})$ gives the travel time,
- $r(v_i, v_{i+1})$ gives the risk.

Travel time can be measured in seconds, for example, but we will not the specify the unit or the physical dimension of risk. One could try to define risk as the probability of mission failure, but that would lead to difficult questions about whether the failure probabilities of different edges should be assumed to be independent. So, we will just assume that risk is measured by a non-negative number, and that the total risk of a path is the sum of its edge risks.

This is an example of *multi-objective* path finding [7] since we have two objectives and no given exchange rate between them – that is, although one can assume a linear exchange rate saying that 60 seconds of travel time is as costly as 42 units of risk, for example, that would be an arbitrary judgement. In a general setting, one can have two alternative paths where one is faster and the other is safer, and it is unspecified which of them is best without a given exchange rate. However, in our setting we assume a deadline for the travel time of each path, which means it *is* possible to define the best alternative without any exchange rate between time and risk: any path that does not meet the deadline is unacceptable, and among the others the one with minimal risk is optimal. Finding such an optimal path is still a challenge since many interesting alternatives need to be generated and examined, and to understand our approach, it is useful to temporarily ignore the deadlines. The next section will describe the set of interesting alternatives in a setting without deadlines: the *Pareto* front.

It can be useful to handle more than one kind of risk, but at this theoretical stage we do not have to introduce more than one cost function for risk. This is because the theoretical model is fine-grained and assigns an amount of risk to each individual edge of the graph, so terrain areas that are extra dangerous can be modelled by assigning a larger amount of risk to the graph edges inside them. However, in Section 4 where we will describe a way to model terrain and road geodata as a graph, we will introduce an alternative way to describe risks via safety factors that are assigned to entire terrain areas, not to individual graph edges. For simplicity, our basic algorithm description will then assume that the same value of the safety factor is used for all risky terrain areas, but in Section 7.1 we discuss a generalization to several safety factors.

3.5 Pareto Paths

We start by looking at the base case with one starting position and one end goal.

The main difficulty in multi-objective optimization is that alternative solutions cannot always be ranked. In our case each path has two different costs, the total travel time t and the total risk r. Given two paths P_1 and P_2, can we say

whether one is definitely better? Yes, sometimes: if P_1 has both shorter travel time and lower risk, then it is clear that P_1 is better than P_2, so since P_1 is available P_2 is not interesting. We say that P_1 *dominates* P_2, according to Pareto efficiency [7]. But if P_1 has shorter travel time while P_2 has lower risk, then both can be interesting. The word "interesting" is informal: an interesting path is one that is not dominated by any other, and the technical term is that such a path is *Pareto optimal*. The set of all Pareto optimal paths is known as the *Pareto front*, and we call them *Pareto paths*.

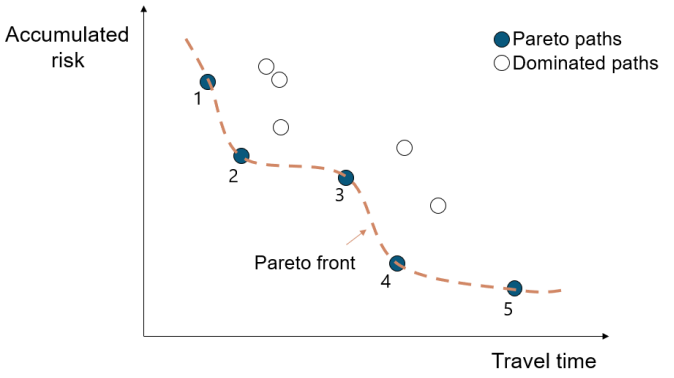

Figure 4: *Pareto front.*

In Figure 4, the diagram shows Pareto paths plotted using the objective functions as axes, with travel time along the horizontal axis and accumulated risk along the vertical axis. The Pareto paths are illustrated with circles filled in blue while the other, dominated paths have transparent filling. The dashed orange line illustrates the Pareto front which is partly concave around Pareto path 3.

We propose to solve this multi-objective optimization problem by the common method of linear scalarization [7], which means that the multiple costs are converted to a single cost so that alternative solutions can be ranked. For each path, we will scalarize its travel time t and its risk r by making a weighted sum:

$$t + wr, \tag{3}$$

where the non-negative weight w can be regarded as an exchange rate that converts risk units into penalty seconds. The absence of a weight for t does not cause any loss of generality. So, at one extreme, one can generate bold paths that ignore all risks by setting w to zero. And at the other extreme, one can generate conservative paths that avoid all risks by setting w to infinity and using the convention that $\infty \times 0 = 0$.

A standard algorithm for pathfinding, like Dijkstra's algorithm or A*, can now be used to find the path with the least scalarized cost, and this will be a Pareto path in the multi-objective setting [9]. The intention is to vary the value of w to generate several alternative Pareto paths to each rendezvous point.

One limitation of linear scalarization is that is not possible to obtain Pareto paths along a concave part of the front. In figure 4, Pareto paths 1, 2, 4 and 5 can be found using linear scalarization although Pareto path 3 cannot. There are methods that can generate all Pareto paths [10, 11], but they are usually slower and we have not explored them.

3.6 Pareto Paths With A Deadline

We will now add a deadline, which is an example of an ϵ-constraint scalarization [8].

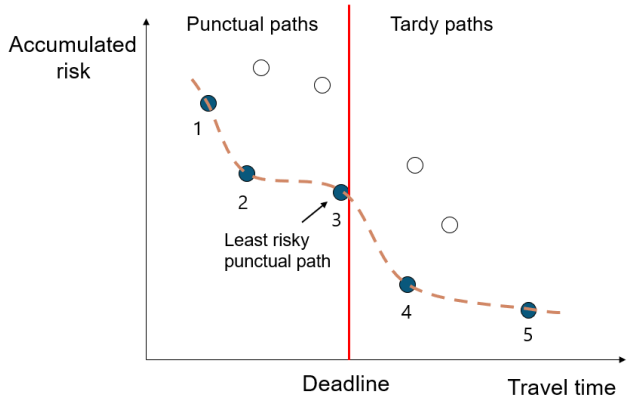

Figure 5: *Pareto front with the additional deadline vertical line. Any Pareto path to the right of the line is considered tardy while anything on the left is punctual.*

As Figure 5 shows, the introduction of a deadline divides all paths into punctual and tardy paths, and the least risky of the punctual paths is the best one. One will still need a way to generate Pareto paths, and we still intend to use scalarization with varying weights to do so, but the advantage of the deadline is that the results can then be ranked automatically and the human operator does not need to examine many Pareto paths manually.

In the example in Figure 5, the least risky punctual path is Pareto path 3 which cannot be obtained by linear scalarization since it is on a concave part of the Pareto front, highlighting the limitation of our method. Instead, Pareto path 2 would be the least risky punctual path that our method would obtain.

4 THE AIPP ALGORITHM

This section describes the fundamental parts of the *Adaptive Intercepting Path Planning* (AIPP) algorithm. Based on the theory in section 3, the algorithm will be introduced in more practical terms.

4.1 From Geodata To Graph

So far, the problem has been described in terms of graphs, but the algorithm will be easier to understand in terms of more concrete data structures. Our prototype implementation assumes that terrain information will be expressed as three kinds of aligned rasters (gridded coverage data) giving elevations, soil strength and roughness, while the road network will be expressed as line geometries with attributes. Then we can define the search graph to have one vertex at the center of each terrain raster cell. Each edge will then go from one cell to one of its eight neighboring cells, and both the speed and the travel time for an edge can be calculated from information in the rasters and from the capabilities of the terrain vehicle type. Road line geometries can either be rasterized into a fourth kind of raster, or else road line segments can be treated as special graph edges that link two non-adjacent raster cells. A risky area can be represented either as a polygon or as a thematic raster. Such an area is not directly assigned a risk r as described in section 3.4. Instead, the areas are assigned a safety factor which is a real number between 0 for a completely unsafe area and 1 for a completely safe area. For any edge inside a risky area, its speed will be multiplied with the safety factor to get a penalized speed, which is used to calculate the penalized travel time for the edge, which was written as $t + wr$ in section 3.5. For example, if the safety factor is 0.25, the penalized travel time is four times longer than the true travel time, so each true minute spent traversing such an area corresponds to a penalized time of 4 minutes, or we can say that to each true minute will be added 3 penalty minutes. In other words, we do not try to define a risk r explicitly and then a separate weight w as in section 3, because the risk r was expressed in an unspecified unit anyway. To summarize, when the true travel time is t and the safety factor is s, the penalized travel time is t/s which is used as the scalar cost during pathfinding (instead of $t + wr$ in section 3.5). When the safety factor s is 0, the formula above causes division by zero which we interpret as producing an infinite cost: in other words, no paths may ever go through areas where $s = 0$. A pathfinding algorithm will optimize for the minimal penalized time, but it must also calculate the true travel time, since it would make no sense to compare the penalized time with the rendezvous deadline.

4.2 Bisecting The Safety Factor Interval

The AIPP algorithm is iterative.

As the first iteration, pathfinding is done with a safety factor of 0, so that all risky areas are treated as fatal and therefore avoided. If this iteration gives any path to a rendezvous position that meets the deadline, the algorithm can stop with success, since we cannot do better than meeting the deadline and avoiding all risks. If there are several paths that meet their deadlines, we can of course choose the one that is most ahead of its deadline.

Otherwise, in a second iteration, pathfinding is done with a safety factor of 1, so that all risky areas are treated as fully safe and not avoided at all. If even this bold approach fails to give any paths that meet their deadlines, the algorithm can stop with failure. Of course, in a real-life scenario, one could then try to adjust the itinerary of the moving frontline unit or try to find another supply vehicle in a better start position – but such considerations are beyond our problem statement.

But suppose that the conservative safety factor $s = 0$ gives no punctual paths, while the bold safety factor $s = 1$ does give at least one punctual path. The algorithm should not stop here, because although we have a punctual path, it could be unnecessarily risky. That is, we want to find the optimal value for s, which is the lowest value that produces a punctual path, and all we know so far is that such a value exists somewhere in the semi-closed interval $(0, 1]$. So, we bisect the interval by doing a new pathfinding with $s = 0.5$. If this value gives any punctual path, we know that the optimal value must be in the semi-closed interval $(0, 0.5]$, otherwise it must be in the semi-closed interval $(0.5, 1]$. By iterating this bisecting process, we can box in the optimal value.

Whenever a value of s gives at least one punctual path to some rendezvous point, we can discard the rendezvous points that did not get a punctual path with s, because the next value of s will be lower and those points cannot get punctual paths when the pathfinder becomes more cautious.

But when should we stop bisecting? There is no absolute answer to that, because even if the interval has become very short and even if the last few successful iterations have all produced identical paths, it is still possible that further bisecting would find a better path. So, one can either just decide on some fixed number of iterations in advance, or else provide an interactive interface that displays the solutions from each iteration as soon as they are ready and allows a human operator to stop further bisections at any time.

An example walkthrough of this algorithm can be found in section 5.1.

4.3 The Pathfinding Component

So far, the algorithm description has not mentioned the basic problem of finding vehicle paths in terrain – we assume that pathfinding is done by a software component whose details are beyond the scope of this paper. However, since the pathfinder is crucial for performance, we can list some basic requirements and describe our own implementation briefly.

Requirements:

- The pathfinder must be able to find the least-cost paths from a start vertex to one or more goal vertices, where the cost is defined by a combination of travel time and risk areas as described earlier.
- To each found least-cost path, the pathfinder must attach two attributes that give its total travel time and the travel time spent in risky areas. This is because neither the AIPP algorithm nor a human operator is interested in the weighted cost, which is an artificial measure just used to steer the pathfinding.
- The pathfinder should handle multiple goals efficiently.

Our implementation:

We had developed a pathfinding component earlier [12], but it did not handle multiple goals well, since it would restart a new search for each goal. To handle multiple goals, we have replaced our old A* algorithm that assumed a single goal by the similar Dijkstra's algorithm that finds the least-cost path from a start vertex to all other vertices.

Both A* and Dijkstra's algorithm needs to use a priority queue for good performance, but there is

some disagreement in the literature about whether the priority queue should support the *Decrease-Key* operation which is cumbersome to implement. Our first implementation used a binary heap for the basic priority queue and an auxiliary data structure to support *Decrease-Key*, but like Chen et al. [13], we eventually found it more efficient to refrain from *Decrease-Key* and instead allow multiple incarnations of the same vertex in the queue.

Although we have found Dijkstra's algorithm to work reasonably well, it is limited by being sequential. The alternative Bellman-Ford algorithm can be parallelized but has the drawback that some threads will often redo work that has already been done on another thread. There are researchers who are developing alternative algorithms that can be massively parallelized on a GPU, for example Wang et al. [14], but the performance can depend on statistical properties of the search graph. Fjellborg [15] has developed such GPU algorithms specifically for search graphs that arise in terrain routing, but the statistical connectivity of the graph still matters. His GPU algorithms can be up to 8 times faster than Dijkstra's algorithm for terrain that is easy to traverse, but are not faster when the terrain is hard to traverse.

5. RESULTS

5.1 Algorithm Walkthrough

We have used our pathfinder to try out the AIPP algorithm on a specific example problem. Our study area is a piece of hilly and mostly forested terrain just south of Little Norway in California, 13 km south of Lake Tahoe, where we have access to relevant geodata in three forms:

- elevation rasters at 20 m resolution,
- thematic terrain type rasters at 30 m resolution,
- road line geometries.

Our pathfinding component can accept the elevation rasters as they are, but it requires two other input rasters containing estimates of soil strength and local roughness on a scale from 1 (best) to 5 (worst). The available terrain type rasters use the twenty-one classes of NLCD (National Land Cover Data) which are not enough for highly accurate trafficability analysis, but we have reclassified them into soil strength and roughness in an approximate way, and resampled the results to the same resolution as the elevations. On the other hand, the twenty-one NLCD classes are too many to get individual colors in a background map, so in our screenshots we are just using four terrain type colors for water, open terrain, shrubs and forests. (The difference in hue between open terrain and shrubs can be hard to see but is not important.)

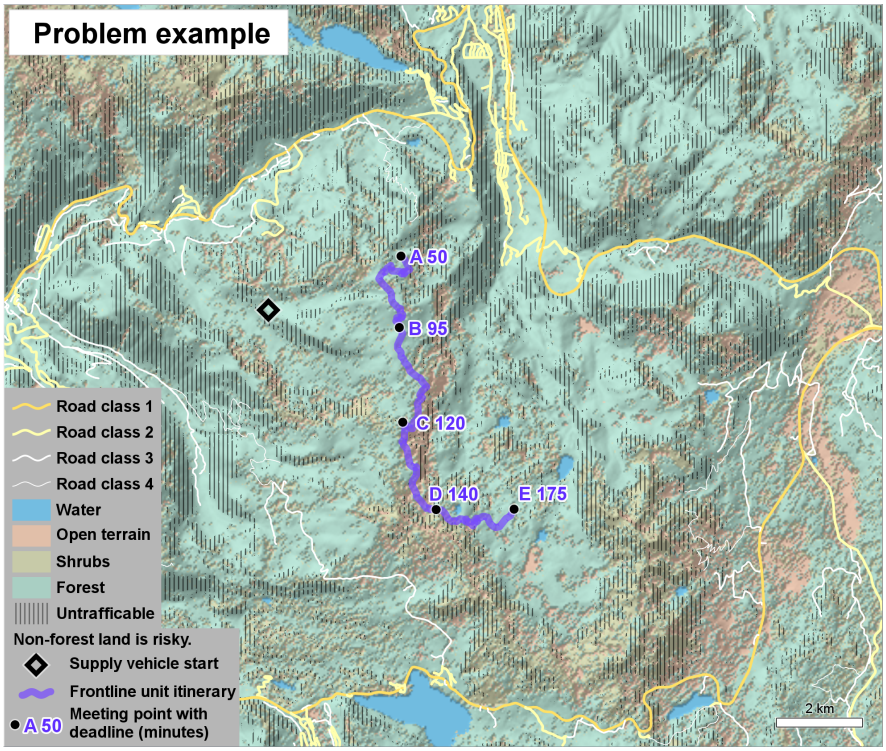

Figure 6: *Problem example.*

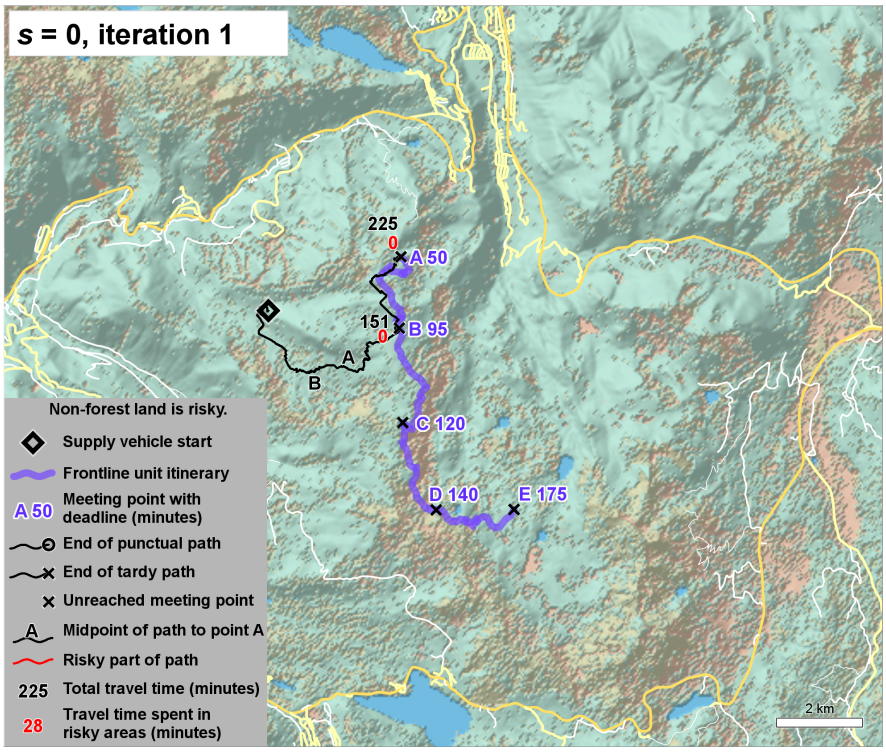

Figure 7: *AIPP results using safety factor $s = 0$, iteration one.*

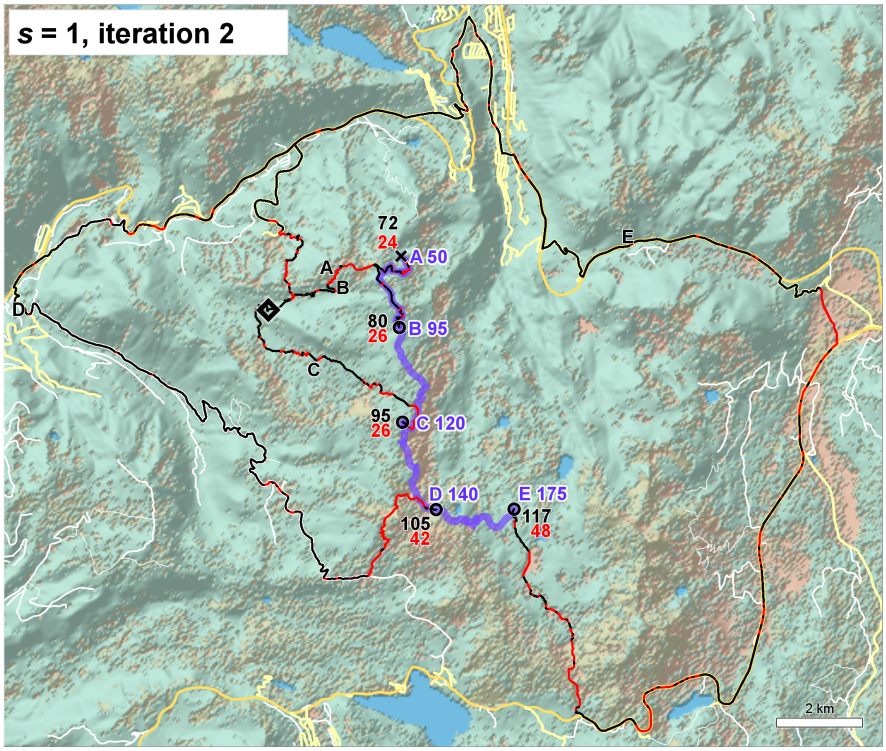

Figure 8: *AIPP results using safety factor* $s = 1$*, iteration two.*

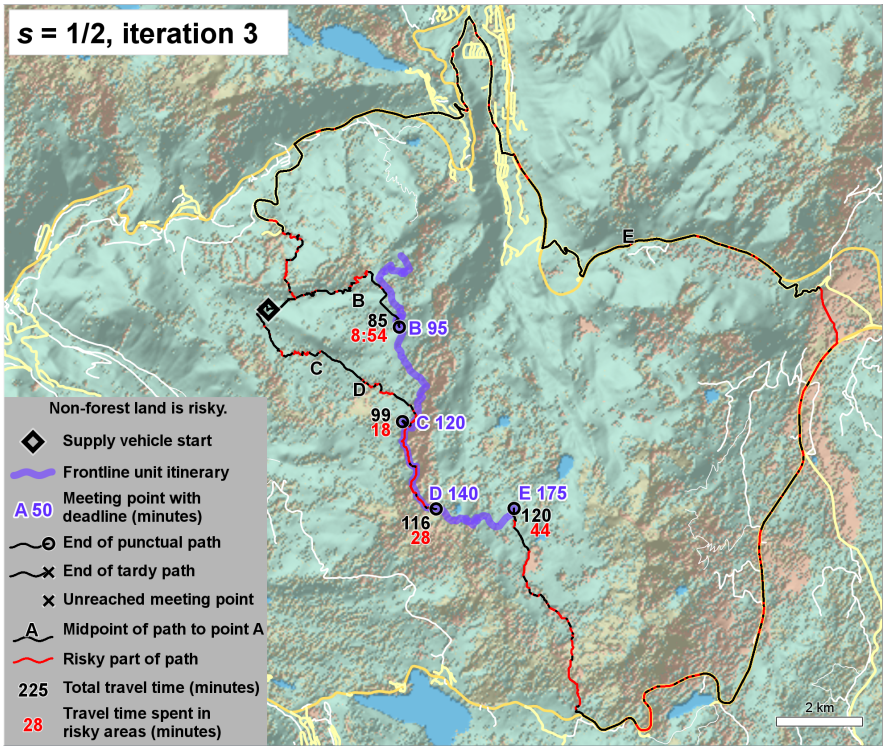

Figure 9: *AIPP results using safety factor* $s = 1/2$*, iteration three.*

In the study area, we have generated a southward-going itinerary for the frontline unit by using our pathfinding component in a preliminary pass, and for simplicity we used the same vehicle model as for the supply vehicle. Then we chose an initial start position for the supply vehicle. See Figure 6. The deadlines of the rendezvous points A to E in the screenshots are expressed in minutes since the operation start, so that they can be compared directly to the travel times along the supply vehicle's paths.

So, what should we use as the risky areas? The basic risk is that the supply vehicle can be detected by the enemy, but viewshed analysis or visibility indexes would be complicated for this example. Instead, we simply assume that the supply vehicle will be concealed in a forest but exposed to risk elsewhere. Since our vehicle model is assumed to go slower and to handle slopes worse in forests than in shrubs or open terrain, and since the forested areas have complex fractured shapes, there will be opportunities to trade safety for speed or vice versa. On the other hand, there are also steep areas in difficult terrain types that are simply untrafficable, so the pathfinder has limited options. The areas that the pathfinder regards as untrafficable are indicated by vertical stripes in Figure 6, since the precise slope values are difficult to estimate just from the hillshading. But the stripes are removed from the subsequent screenshots to reduce the visual clutter.

5.2 AIPP Iterations

Let us now walk through the AIPP algorithm.

In the first iteration we use the safety factor $s = 0$ that makes the pathfinder maximally cautious: only the forest can be traversed. With this setting, the pathfinder finds two tardy paths to rendezvous points A and B and none to the other points; see Figure 7. Each rendezvous point has been placed inside a forested and trafficable area, but points C, D and E are probably surrounded by rings of terrain that is non-forested or untrafficable or both.

Since we have not found a punctual and fully safe path, we must do a second iteration with $s = 1$ which makes the pathfinder maximally bold: it will leave forests without qualms. Now, the pathfinder finds a tardy path to A and punctual paths to B, C, D and E. See Figure 8. Since A cannot be reached in time even with maximal boldness, we can drop it from further consideration; the paths to the other points are punctual but they probably spend more time than necessary in risky areas. So we bisect to make a third iteration with safety factor $s = 1/2$, see Figure 9. The screenshots for iterations No. 4 to 7 can be found in the Appendix, and the results from all iterations are summarized in Table 1, where the safety factor values are listed in increasing order, with a row above indicating in which iteration they were tried. Times are written either as whole minutes or as minutes and seconds separated by a colon.

As in the screenshots, purple text denotes rendezvous points with deadlines, black numbers indicate total travel times for paths, and red numbers next to them indicate the travel time spent in risky areas. In addition, the total travel times that are punctual have got a green background and the winning alternative has a gray background. A label "No path" means that the pathfinder failed, while a dash means that the pathfinder was not invoked.

In the main algorithm description, we did not formulate a definite rule when the bisections should stop. But in this example, one can reason that an 8th iteration with s halfway between 3/16 and 7/32 would not find a punctual path to C, because the path to C had become tardy already in iteration 7 with s = 7/32, and a smaller value of s would make the pathfinder more cautious, so the total travel time cannot become shorter. That leaves only B, but in the last two iterations, the travel time spent in risk areas differs only by 2 seconds and the total travel time differs only by 10 seconds, so an 8th iteration will not be able to improve things much.

Iteration	1	5	6	7	4	3	2
s	0	1/8	3/16	7/32	1/4	1/2	1
A 50	225 (0)	-	-	-	-	-	72 (24)
B 95	151 (0)	112 (2:26)	95:08 (5:08)	94:58 (5:10)	91 (6:19)	85 (8:54)	80 (26)
C 120	No path	143 (2:26)	130 (4:37)	122 (6:44)	117 (8:04)	99 (18)	95 (26)
D 140	No path	-	-	-	143 (14)	116 (28)	105 (43)
E 175	No path	-	-	-	179 (16)	120 (44)	117 (49)

Table 1: Summary of algorithm walkthrough. See section 5.2 for details.

6. PERFORMANCE

The computational speed of our implementation is highly dependent on the size of the area to be searched. Since we use terrain rasters of 20 meters resolution, we do not want to retrieve geodata from an unnecessarily large search area. On the other hand, if the search area has been too cropped, the pathfinder could fail to find some useful paths: in our examples, it could fail to find the two paths to D and E in Figure 8 that take long detours on roads.

Choosing a suitable area of interest is difficult, but a simple approach is to put the responsibility on the human user. For example, one can define the search area as based on the bounding rectangle of the start position and all goal positions, but extended with a user-defined margin. The user could choose the margin to be a constant distance or use some rule of thumb like "twice the longest distance from start to any goal plus 15 km".

To get a well-defined context for performance results from our example, we have chosen the search area to be exactly the area shown in Figure 6, which is large enough to contain the roads the surround the central terrain. This area is 20500 × 17260 meters, so at 20-meter resolution we get terrain rasters of 1025 × 863 cells, corresponding to a search graph of 1025 × 863 = 884575 vertices. If we consider the edges between two adjacent raster cells to be directed, we get about 8 times as many edges of that kind. The search area also contains 844 road line features that together contain 12638 line segments that our implementation treats as extra graph edges. On the other hand, the search area contains many small terrain areas that are untrafficable by our vehicle model, so these areas will not be searched by the pathfinder algorithm.

Using this search area, each iteration of our algorithm example that used a positive s took roughly the same time to calculate, in the range from 215 to 275 ms. The first iteration with $s = 0$ was much faster at 75 ms, which makes sense since it always stays in the forest and explores less of the search area. The processor used is an Intel® Core™ i7-10700K.

For each iteration, our implementation begins by using Dijkstra's algorithm to generate the least-cost path tree to all reachable vertices in the search area; this tree is a space-efficient representation of all the least-cost paths. After that, the algorithm must use the tree to retrace the path to each reachable rendezvous point, to be able to calculate the total travel time and the time spent in risk areas, but this retracing can be done quickly, requiring only 5 to 8 ms in our example. Altogether, the seven iterations of our example took 1490 ms.

Our pathfinder component is still a prototype and can be improved further: for example, it is obvious that Dijkstra's algorithm does not need to run until it has found all reachable vertices in the search area, only until it has reached all the reachable rendezvous vertices.

Since the total computation time depends almost entirely on the size of the search area and only

slightly on the number of rendezvous points, one can afford to use many rendezvous points. Instead of manually selecting around half a dozen points, one can let the algorithm automatically extract rendezvous points at, say, every 100$^{\text{th}}$ meter along the itinerary. In our example the itinerary is 11926 meters long, so that approach would extract 120 rendezvous points. Based on the measured performance for 5 rendezvous points, we estimate that using 120 points instead would increase the computation by at most 700 ms.

While the number of rendezvous points does not matter much, a geographically larger search area will be slower. If the pathfinder is based on Dijkstra's algorithm, the complexity will be $\mathcal{O}(n \log n)$ where n is the number of terrain raster cells in the search area, since the number of edges from the raster is proportional to the number of vertices and the edges from road line segments should be few in comparison.

7 FUTURE WORK

7.1 Handling different kinds of risk

So far, our algorithm description and our example scenario has assumed that there is only one kind of risk area that can be assigned a uniform safety factor. But in the example scenario, it could make sense to claim that while forests are safe and open terrain is risky, shrubland is somewhere in between, offering some concealment but not as good as forests.

In our basic implementation of safety factors in our pathfinding component, it is already possible to assign different values of the safety factor to different kinds of risk areas. If this is done, the generated paths will contain several risk attributes: instead of only one attribute giving the travel time spent in a single kind of risk area, the pathfinder will produce one attribute for each kind of risk. For example, there could be one attribute for the travel time through shrubland and another for the travel time through open terrain.

But if one wants to exploit this in the AIPP algorithm, some generalizations must be made. We propose that a human expert should first assign numeric weights to the different kinds of risks on a numeric scale that is calibrated to the safest kind of risk. In our example, shrubland is risky but safer than open terrain, and the safety factor s used in the AIPP algorithm description should then be used for shrubland. Now, if the human expert has decided that open terrain is only 60% as safe as shrubland, the generalized AIPP algorithm can communicate that information to the pathfinding component in each iteration. That is, whenever AIPP asks the pathfinder to treat shrubland as having the safety factor s, it will also ask it to treat open terrain as having the safety factor $0.6s$.

In this way, the AIPP algorithm will be able to find the punctual path that has the least amount of weighted risk, but the result will of course depend on how the risk weights were chosen. We believe this numeric amount of weighted risk will not be easily interpreted by a human operator. And that is why it is important that the results should be displayed with the explicit travel times through each kind of risk: to show numbers that are understandable. The weights assigned to the different kinds of risk will be subjective, but a human operator can experiment with different settings to get more suggestions.

7.2 Early Maneuver

The AIPP algorithm generates a solution that re-balances travel time and risk to account for a new deadline to reach the meeting between the supply vehicle and the frontline unit. The link up is done using a set of possible rendezvous points, where the rendezvous point that accumulates the least amount of risk is chosen. Although, an interesting topic at hand is the idea of an early maneuver.

An early maneuver is defined as the idea of cutting corners *early* relatively speaking along the path in order to catch up on the original path. The attempt is to create a new solution that re-uses the

old solution and rendezvous point, however, the supply vehicle will try to gain speed by cutting into risk-filled areas. These scenarios assume that the supply vehicle is losing travel time by avoiding a set of risk filled zones.

The positive aspect of an early maneuver is to tackle the possibility of additional unforeseen loss of traverse speed along said path. Let's say that the supply vehicle is cutting corner early, re-routes back again on the original path and stays on time. Assume that a bit further down the path, the supply vehicle slows down again. Since the maneuver to cut corners was done at an early stage, the vehicle can now more robustly handle this scenario with more leeway to cut corners again to reach the link up. If the AIPP algorithm proposes cutting corners in the endgame of the path, due to less accumulated risk, the supply vehicle might still lose additional time along the new path. This could result in a scenario in where the interception no longer is feasible.

To do this, one could try and route from current position to an existing position following the original path. The new position should be far away enough such that the time gain is enough to cover the time loss. In example, the supply vehicle has lost 2 minutes of time due to unforeseen delays, a re-route is done such that cutting corners through a risk area is done almost immediately to then fall back to the original planned path.

7.3 Handling uncertainty

We have assumed that the frontline unit itinerary is predetermined and that the supply vehicle paths can be calculated precisely. In reality, the arrival times to the rendezvous points will of course be uncertain. For the frontline unit, it seems likely that the uncertainty accumulates along the itinerary, so that later rendezvous points have more uncertain deadlines.

For each problem instance, we think one could define a more conservative variant where the frontline unit is assumed to go 5 percent faster and the supply vehicle 5 percent slower, say. When the algorithm is applied to the conservative variant of the problem, the deadlines will be stricter and the supply vehicle may need to take more exposure risks to satisfy them. In this way, one would get alternative paths that reduce the risk of a missed deadline.

7.4 Cooperating Frontline Unit

In this case study, the frontline unit follows a predetermined itinerary and the supply vehicle has the full responsibility to adapt. But if the frontline unit is willing to cooperate by adapting its itinerary, it may be possible to find a better rendezvous point. This could require the frontline unit to take more risks and be more delayed, but the overall risk for the mission could be decreased. We are looking into possible modifications to the AIPP algorithm to achieve this additional flexibility.

8. CONCLUSIONS

Resupply missions in modern combat is of upmost importance for mission critical success. Both in the aspect of manufacturing large quantities of supplies, such as ammunition, but also the logistics of getting it out on the frontline. The AIPP algorithm has shown to be capable of generating punctual paths that enables the link up between supply vehicle and mobile frontline unit, using a small number of iterations. For example, after twelve iterations the final bijected safety factor intervals will have become 1/1024, which should suffice in practice.

A well-defined final result can be chosen automatically by the algorithm as the punctual path to some rendezvous point that spends the least time in a risky area. This ensures that the operational user does not receive an overwhelming amount of computer-generated paths to choose from, as can happen with multi-objective path planning algorithms without deadlines. However, it can also be useful to present some alternative punctual paths to other rendezvous points, alternatives that were found in intermediate iterations.

We believe that the ability to handle different kinds of risk areas, as discussed in section 7.1, will be important in practice. The operator would need to assign weights to the kind of risks presented, and changing the weights will result in different outcomes, adding complexity to the problem. But this complexity is useful: since the AIPP algorithm accepts some level of risk, the operator should be able to pick between the relevant kinds.

9 APPENDIX

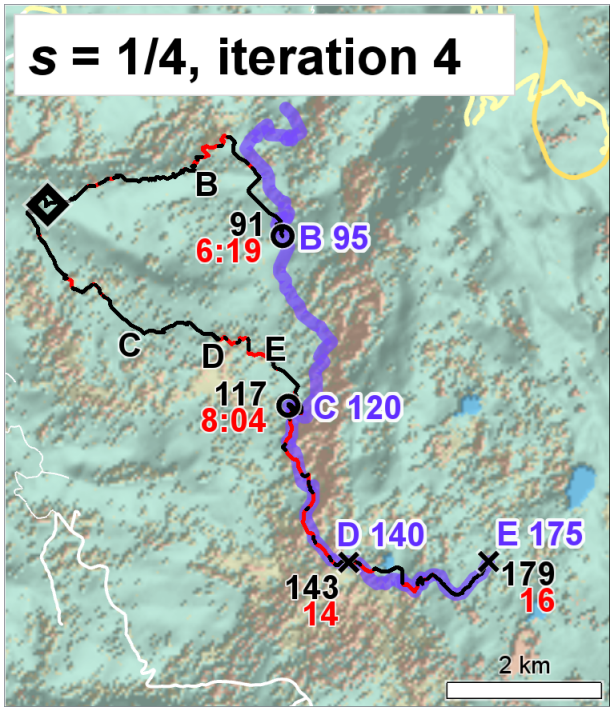

Figure 10: *AIPP iteration four.*

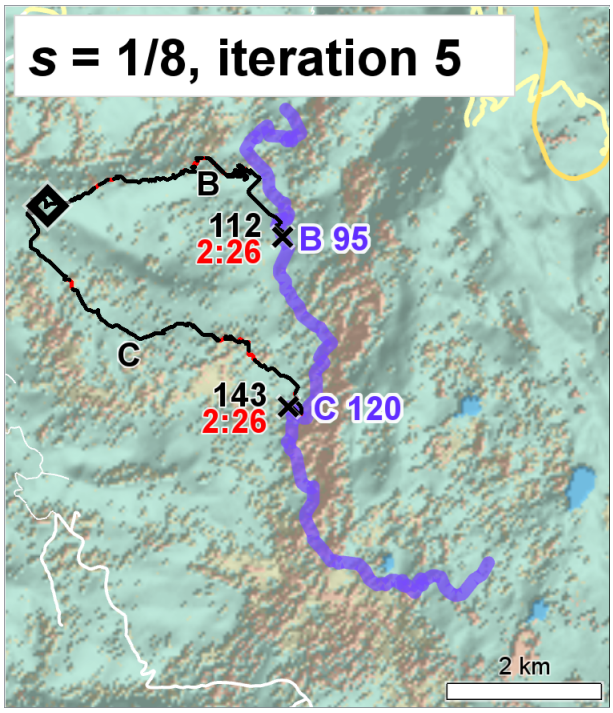

Figure 11: *AIPP iteration five.*

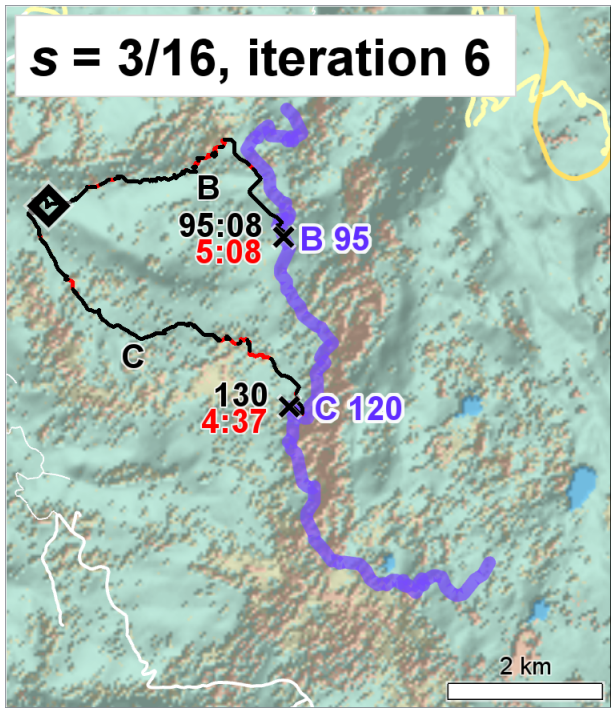

Figure 12: *AIPP iteration six.*

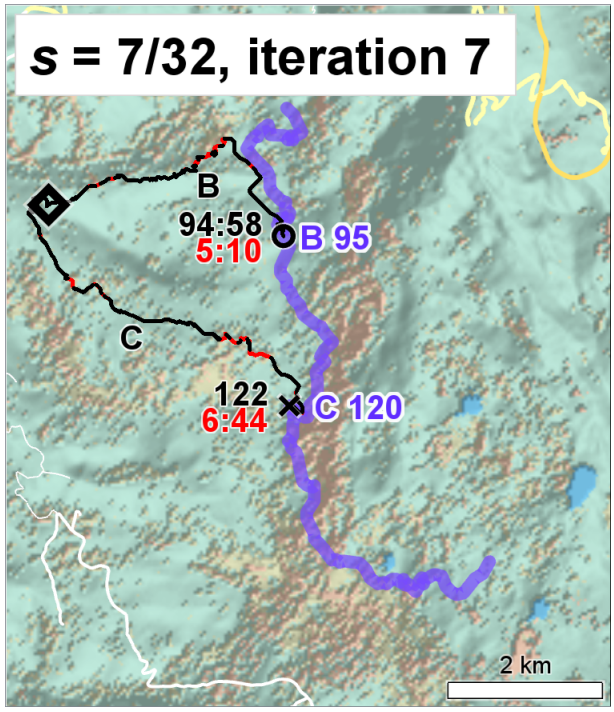

Figure 13: *AIPP iteration seven.*

10. REFERENCES

References

[1] T. Spirlet, "NATO vastly underestimated Russia's ability to resupply its army with troops and ammo, general says," *Business Insider*, Jan. 2024. [Online]. Available: https://www.businessinsider.com/nato-underestimated-russia-ability-to-resupply-army-troops-ammo-general-2024-1

[2] S. Mansoor, "Why U.S. HIMARS rockets are becoming increasingly decisive for Ukraine," *TIME*, Jan. 2023. [Online]. Available: https://time.com/6244479/himars-rockets-ukraine-russia/

[3] S. Cameron-Moore, "Russian ammunition depot hit during Ukrainian drone attack in Crimea, official says," *Reuters*, Jul. 2023. [Online]. Available: https://www.reuters.com/world/europe/ammunition-depot-hit-crimea-moscow-installed-official-2023-07-24/

[4] V. Mittal, "Ukrainian military is targeting Russian fuel supply lines as winter approaches," *Forbes*, Dec. 2022. [Online]. Available: https://www.forbes.com/sites/vikrammittal/2022/12/11/ukrainian-military-is-targeting-russian-fuel-supply-lines-as-winter-approaches/?sh=6af58c73e2dd

[5] W. T. Tutte, *Graph Theory*, ser. Encyclopedia of Mathematics and its Applications. Addison-Wesley Publishing Company, 1984, vol. 21.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100 – 107, Jul. 1968.

[7] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, pp. 369–395, 2004.

[8] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research & Management Science. Springer US, 1999. [Online]. Available: https://books.google.se/books?id=ha_zLdNtXSMC

[9] D. Xiang, H. Lin, J. Ouyang, and D. Huang, "Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot." *Nature, Sci Rep 12, 13273 (2022)*, 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-17684-0

[10] Z. Clawson, X. Ding, B. Englot, T. A. Frewen, W. M. Sisson, and A. Vladimirsky, "A bi-criteria path planning algorithm for robotics applications," Preprint. https://arxiv.org/abs/1511.01166, 2017.

[11] L. Li, "Multi-objective A* route optimization for terrain vehicles," Master's thesis, KTH Royal Institute of Technology, Stockholm, 2020.

[12] T. J. Damgaard, M. Rittri, P. Franz, and A. Halota, "Robust path planning in the battlefield," in *In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*. Novi, Michigan: NDIA, 2023.

[13] M. Chen, R. A. Chowdhury, V. Ramachandran, D. L. Roche, and L. Tong, "Priority queues and Dijkstra's algorithm," Computer Science Department, University of Texas at Austin, Tech. Rep. UTCS TR-07-54, 2007.

[14] K. Wang, D. Fussell, and C. Lin, "A fast work-efficient SSSP algorithm for GPUs," in *Proceedings of the 26th ACM SIGPLAN*

Symposium on Principles and Practice of Parallel Programming, 2021.

[15] J. Fjellborg, "An implementation and performance evaluation of parallel algorithms for off-road vehicle routing on the GPU," Master's thesis, KTH Royal Institute of Technology, Stockholm, 2024.