Vector vs. Raster-based Algorithms for Cross Country Movement Planning

Joost van Bemmelen, Wilko Quak,
Marcel van Hekken, and Peter van Oosterom

TNO Physics and Electronics Laboratory,
P.O. Box 96864, 2509 JG The Hague, The Netherlands.
Phone: +31 70 3264221, Fax: +31 70 3280961,
Email: oosterom@fel.tno.nl

Abstract

The problem of cross country movement planning is to find the least cost path between a source and a destination given a polygonal partitioning of the plane. This paper presents the comparison between a vector-based algorithm and several raster-based algorithms for solving the cross country movement planning problem. Besides three known raster methods (standard, more-connected, and quadtree), a new method, *extended raster* approach, has been designed and implemented. The latter finds better solutions because of the higher degree of freedom when moving through a rasterized terrain, that is, more directions can be followed. The vector method of Mitchell and Papadimitriou has also been implemented for comparison. Assuming a reasonable sized terrain with a reasonable accuracy all methods have performance problems. Therefore, a *hierarchical approach* is presented which can be applied in both the vector and raster domain. This approach will find the optimal path in contrast to other hierarchical approaches, which do not guarantee to find the optimal path, but often return a good path. The performance improvement should make the hierarchical approach suitable for interactive movement planning tasks.

1 Introduction

The cross country movement (CCM) problem is also known as the Weighted Region (Least Cost Path) Problem [15, 22]. The unit cost of traversing a given region (polygon) is uniform within the region, but varies between regions and is based on soil, vegetation, and so on. Finding an optimal path, that is, locating a corridor from a given source location to a given destination location [3, 6, 10] is not only used for travelling. It can also be applied in other planning situations, such as building highways, railways, pipelines and other transport systems. The cost function is based on optimization criteria such as time, safety, fuel usage, impact, length, etc. Note that the CCM-problem is very different from the more common linear route planning in a road network, because the cost of using an element (polygon) is not fixed.

Section 2 contains the description of four different raster-based algorithms for the CCM planning problem. The vector-based approach of Mitchell and Papadimitriou

[15] follows in Section 3. Section 4 contains details with respect to the implementation, some test results, and a comparison of the different methods. All presented solutions require a lot of time and space when using realistic data sets. Therefore, an exact solution hierarchical approach is given in Section 5. Finally, conclusions and future work are described in the last section.

2 Raster CCM Algorithms

The first type of solution to the CCM-problem begins with superimposing a regular rectangular grid on the plane [1]. The sampling theories of Shannon, Whittaker and Nyquist [11, 17] state that the gridsize should be at least $2\sqrt{2}$ times smaller than the smallest detail to be kept. After the rasterization a suitability score (weight or cost value) is assigned to each grid-cell. This value represents the "cost per unit distance" travelling inside a cell. Several techniques, such as the Delphi Method and the Nominal Group Technique [9], have been developed to assign the scores. These scores are determined by several geographic variables such as soil type, obstacles, vegetation and overgrowth. Other factors that influence the score are weather conditions and means of transportation: foot, bike, car, tank, etc.

Several raster-based algorithms for finding the shortest path[2] have been developed. All algorithms are based on making a graph with nodes (e.g. raster cells) and implicit edges, the possible movements between the nodes. Then the Dijkstra [1, 4] graph search algorithm is used to find the optimal path. Other search techniques could be used instead, e.g. based on heuristics: A*-algorithm [8]. The main difference between the algorithms is the assignment of nodes and edges:

- standard 4-connected rasters: only rook's moves on a chess board are allowed; see Subsection 2.1;
- more-connected rasters [3, 6, 10]: 8-connected rasters (queen's move are also allowed), 16-connected rasters (queen's+knight's moves), 32-, 64- and 128-connected; see Subsection 2.2;
- extended raster: this new method extends the number of possible angles to move from one individual grid cell to one of its 8-connected neighbors; see Subsection 2.3;
- quadtree [20] based rasters in order to save memory space; see Subsection 2.4.

2.1 Standard Raster Distortions

A number of geometric distortions can be distinguished in the paths produced on a standard raster grid: *elongation*, *deviation* and *proximity* distortion [6, 10]. In Fig. 1 a raster of a uniform terrain, i.e. each cell having the same suitability score, is visualized. A route is planned from raster cell s to raster cell t. The path which is as close as possible to the continuous-space path (see Fig. 3), is a stair-stepped route. The *elongation* error ϵ is defined as the ratio of the cost of the calculated shortest path to the cost of the continuous-space path; e.g. in Fig. 1, $\epsilon = 2/\sqrt{2} = \sqrt{2}$, that is, the calculated shortest path on a 4-connected raster is approximately 41 per cent longer than the continuous-space path.

[1]An alternative grid is a honeycomb grid in which each hexagon has six neighbors [12].

[2]The term "shortest path" has the same meaning as "optimal path" in this paper.

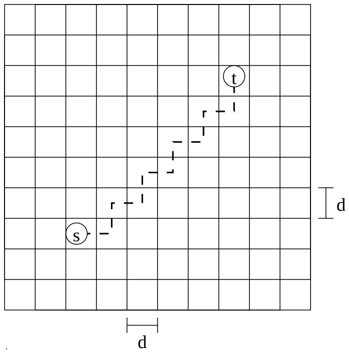

Fig. 1: Stairs effect.

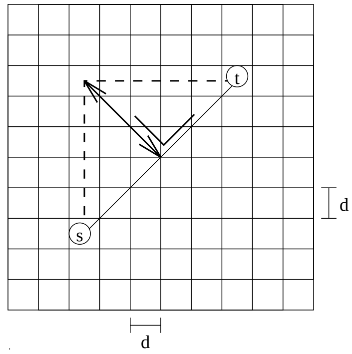

Fig. 2: Max deviation.

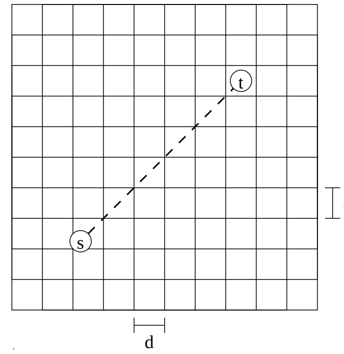

Fig. 3: Real distance.

In general, on a uniform grid, the shortest path will make moves in at most two directions. The maximum *deviation* error δ occurs when all moves in one direction are executed first; e.g. in Fig. 2, $\delta = 0.5$ times the continuous-space path. Note that paths in Fig. 1 and 2 have the same elongation error.

Proximity distortion occurs from the fact that suitability scores are usually calculated for each cell independently of its neighbors. In this way paths can be found which are calculated optimal, but the influence of their neighborhood is ignored. Therefore, paths trough a cheap narrow strip will get the same cost value as paths trough a large uniform terrain. The problem of proximity distortion can be solved by performing a smoothing operation on the suitability score matrix. Note that only the score matrix changes and that the actual problem of route planning does not change.

2.2 More-Connected Rasters

An approach to reduce the distortions in the standard raster is to extend the number of directions a route can follow from each cell. If new implicit edges (queen's moves) are also inserted between every two already existing edges, then a *8-connected* raster is created; see Fig. 4. This process can be repeated to obtain a *16-connected* raster by adding the knight's moves (Fig. 5) and repeated again to obtain a *32-connected* raster; see Fig. 6. In the same manner *64- and 128-connected* rasters can be created.

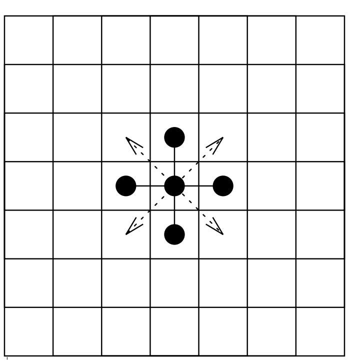

Fig. 4: 8-connected.

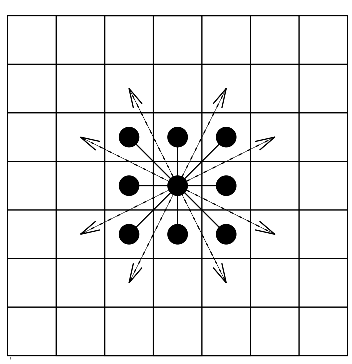

Fig. 5: 16-connected.

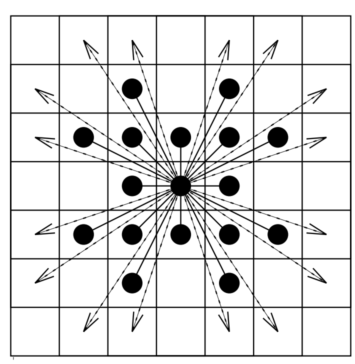

Fig. 6: 32-connected.

In a uniform terrain, the increment in directional possibilities leads to better shortest paths, not only intuitively (Fig. 7, 8, and 9), but also mathematically [10]; see Fig. 10. Fig. 11 visualizes the difference in shortest paths found with different connected raster approaches in a simple terrain. The more-connected raster approach has also several drawbacks. The computation of the cost of an edge is more complicated, because an edge may pass several cells with line segments with different lengths [24]. Furthermore, more-connected rasters imply more dense graphs, which result in longer computation times.

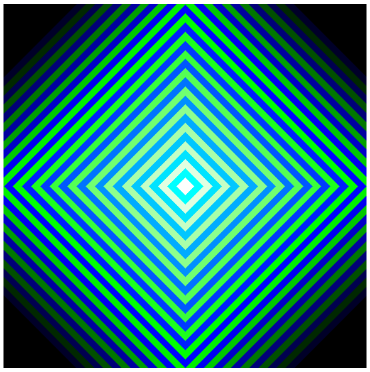

Fig. 7: 4-connected.

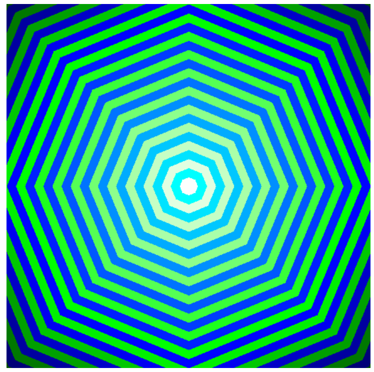

Fig. 8: 8-connected.

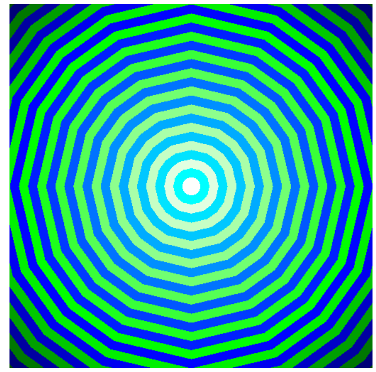

Fig. 9: 16-connected.

Another drawback is that in a non-uniform terrain certain desired angles may be expensive, because the longer edges may also pass very expensive cells. Perhaps the most surprising drawback of the 16- and more-connected raster cell approach is that intersecting paths are possible. One expects that when two shortest paths meet that they will converge and continue as one path to the source. In Fig. 12.a an example of two intersecting paths is given. In this figure the dark colored raster cells have a higher suitability score than the light colored raster cells.

connect	Maximum Elongation ϵ	Maximum Deviation δ
4	1.41421	0.50000
8	1.08239	0.20710
16	1.02749	0.11803
32	1.01308	0.08114
64	1.00755	0.06155
128	1.00489	0.04951

Fig. 10: Maximum elongation and maximum deviation errors.

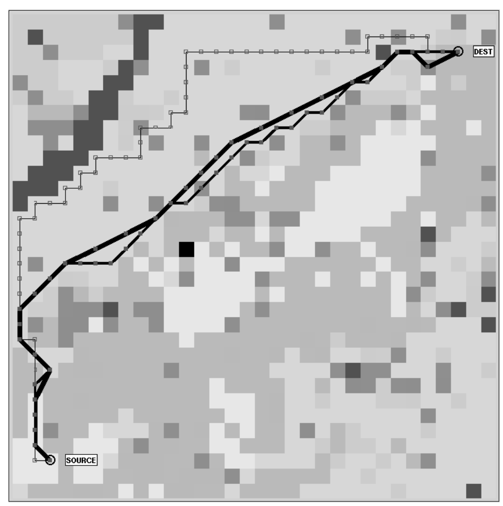

Fig. 11: 4-, 8-, and 16-connected paths.

2.3 Extended Raster Approach

The extended raster approach solves the problem of intersecting paths and possibly too expensive angles because of long edges by defining graph nodes at the sides of the raster cells instead of at the centers of the raster cells; see Fig. 12. By varying the number of graph nodes on the sides of the raster cells the number of allowed directions can be varied. Nodes in the center of raster cells (or anywhere *inside* a raster cell) are not necessary, because according to *Snell's law of refraction of light*, paths within uniform areas are always straight lines; see Subsection 3.2. Note that in the extended raster approach there are two kind of nodes: *data nodes,* which correspond to raster cells and *search nodes,* which are located on the boundaries of the raster cells.

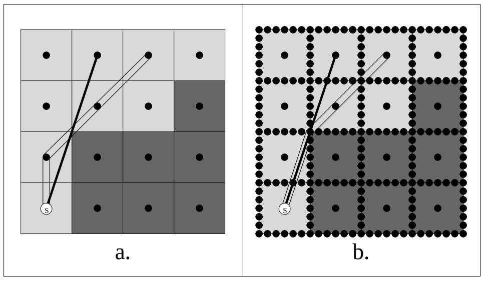

Fig. 12: Intersection and fusion.

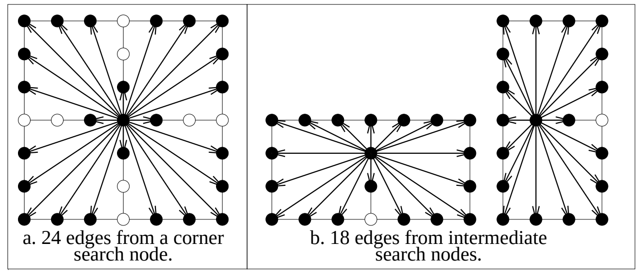

Fig. 13: 2 intermediate nodes.

The total number of search nodes n_s on a square raster of $n \times n$ raster cells (data nodes) with i intermediate search nodes on each side of a raster cell is $n_s = (2i+1)n^2 + (2i+2)n + 1$. In the graph, the total number of edges is $|E| = 2(3i^2+5i+2)n^2 + 2(i+1)n \approx 2(3i+2)(i+1)n^2$. Note that the number of directions which can be followed from each corner search node differs from the number of directions which can be followed from each intermediate search node; see Figs. 13.a and 13.b. The search graph of the extended raster approach without intermediate search nodes can be compared to the search graph of the 8-connected raster approach. The higher degree of freedom with the increasing number of intermediate search nodes i is visualized in a uniform terrain in Fig. 14, 15, and 16. However, the same improvement is also obtained in non-uniform terrains in contrast to the more-connected approach.

Another advantage of the extended raster method is that each edge is contained in exactly one raster cell. The movement cost of such an edge is calculated depending on the length and on the suitability score of only one raster cell. Horizontal or vertical moves along the raster edges use the suitability score of the cheapest cell adjacent to the edge. A drawback of the extended raster method is that the search graph gets quite dense when the number of intermediate nodes i increases. However, the search graph can be made less dense by applying Snell's law of reflection, which states that shortest paths bend towards the normal when entering a cell with higher cost; see Subsection 3.2.

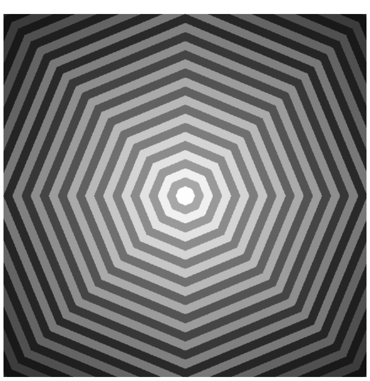

Fig. 14: $i = 0$ nodes.

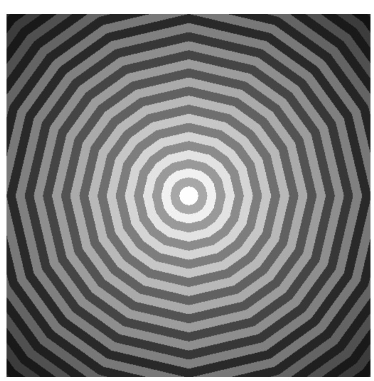

Fig. 15: $i = 1$ node.

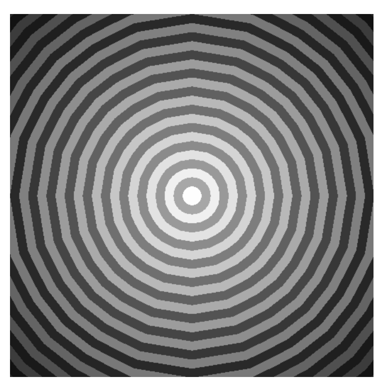

Fig. 16: $i = 2$ nodes.

2.4 Quadtree Approach

One of the drawbacks of the previously described approaches is that even in uniform areas every sample-point is stored as a separate data-object. In the *quadtree* [20] data structure, these uniform areas are grouped together. This does not only result in more efficient memory usage, but also in more efficient algorithms, because of the smaller number of nodes and edges in the graph.

The *quadtree* is a hierarchical data structure, and is based on the successive subdivision of a raster into four equal-sized quadrants or regions. If a quadrant does not consist entirely of raster cells with the same suitability score, then it is again subdivided into quadrants, and so on. In this way regions are obtained with a uniform suitability score.

As an example of the quadtree, consider the $2^3 \times 2^3$ raster shown in Fig. 17. In Fig. 19, it is shown how this raster is subdivided into square regions. The layout of the quadtree data structure is shown in Fig. 18. All white nodes in this figure correspond with regions in the quadtree and are called *leaf* nodes. The black nodes are called *internal* nodes. The topmost node is called the *root* node. The search graph is created by connecting all adjacent (along an edge or point) cells; see Fig. 19. In order to be able to search the graph efficiently all the edges are stored explicitly.

1	1	1	1	1	1	9	9
1	1	1	1	1	1	9	9
1	1	1	1	9	9	9	4
1	1	1	1	9	9	4	4
1	1	1	4	4	4	4	4
1	1	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4

Fig. 17: Raster.

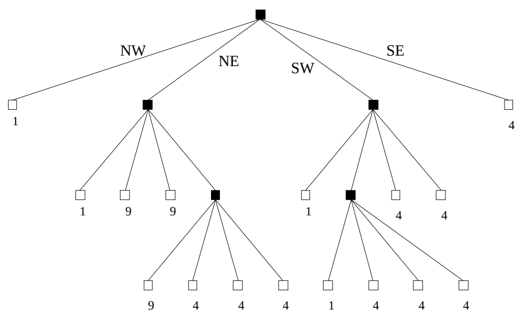

Fig. 18: Quadtree.

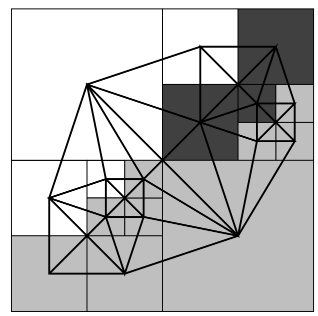

Fig. 19: Graph

Although in the worst case the number of regions in the quadtree is equal to the number of sample points (e.g. a chess board), in most cases the number of regions is much smaller. When there is not much reduction, then it is possible that the gridsize has been chosen too large in the sampling phase. One drawback of the quadtree method is that within a large quadtree node every “position” has the same distance from the source. This is visualized in an iso-time diagram as a large blob with only one color. A related problem is caused by the fact that the path has to go from center to center, which may result in non-optimal solutions in the case of large nodes; see Fig. 21. A solution for this might be a combination of the quadtree and the extended raster approach.

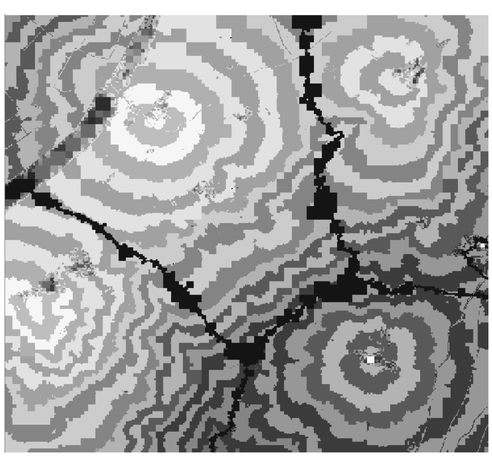

Fig. 20: Weighted Region Voronoi Diagram.

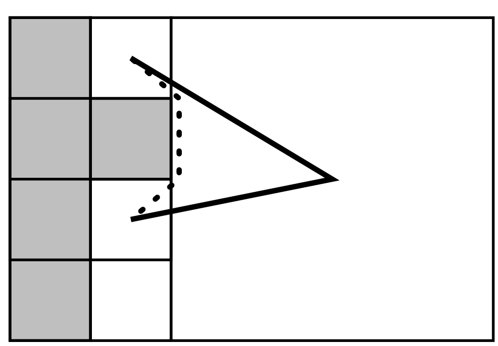

Fig. 21: Deviation error caused by large quadtree cells.

3 Vector CCM algorithm

The precision of the raster-based solutions, i.e. the quality of the returned shortest paths, depends on the fineness of the grid, and on the number of move directions permitted. The ideal solution is only found when both tend to infinity. Clearly, this is a very unpractical solution. Therefore, the vector approach, which produces an exact solution, is examined. In the vector-based movement planning problem it is now assumed that the plane is subdivided into polygonal regions. Each region (face) is associated with a positive weight $\alpha_f\,(> 0)$. However, each edge is also associated with a weight $\alpha_e\,(> 0)$ in contrast to the raster approach. These weight factors specify the cost per traversed distance in a region or along an edge. By giving the edges their own weight factor, on-road movement planning and CCM-planning can be nicely integrated. Obstacles can be seen as regions with an infinite weight. We now want to find the path with the least cost across this map given a source and a destination. The vector-based algorithm to solve the CCM planning problem (also called *weighted region problem*) is based on the article of Mitchell and Papadimitriou [15].

The first big step in the vector algorithm is to triangulate the polygonal partitioning and is described in Subsection 3.1. Some observations about shortest paths in a vector map are given in Subsection 3.2. Subsection 3.3 gives an outline of the vector CCM algorithm.

3.1 The Constrained Delaunay Triangulation

The algorithm we developed to build a *constrained Delaunay triangulation* (CDT) over a planar set of n points together with a set of non-intersecting edges has the properties that all specified points and edges can also be found in the output and it is as close as possible to the unconstrained Delaunay triangulation [18]. It runs in $O(n \log n)$ time, which is asymptotically optimal [21]. This algorithm is based on the concept of two other algorithms. The first algorithm is the unconstrained Delaunay triangulation (UDT) algorithm of Lee and Schachter [14] and the second algorithm is the CDT algorithm of Chew [2].

The input of our algorithm is a graph $G = (V, E)$ in which V is a set of n vertices and E is a set of edges, the so called *G-edges*. Two different kinds of edges appear in a CDT: G-edges, already present in the graph, and *D-edges*, created by the CDT algorithm. If the graph has no G-edges then the CDT and the UDT are the same. The graph can be thought of to be contained in an enclosing rectangle. This rectangle is subdivided into n separate vertical strips in such a way that each strip contains exactly one region (a part of the strip) which in turn contains exactly one vertex. After dividing the graph into n initial strips, adjacent strips are pasted together in pairs to form new strips. During this pasting new regions are formed of existing regions for which the combined CDTs are calculated; see Figs. 22, 23, and 24. This pasting of adjacent strips is repeated following the divide-and-conquer paradigm until eventually exactly one big strip, consisting of exactly one big region, is left for which the CDT is calculated. More details about the algorithm and its implementation can be found in [24].

3.2 Principle Properties of Shortest Paths

The following properties form the basis of the vector-based CCM algorithm and are valid for the shortest path between two points:

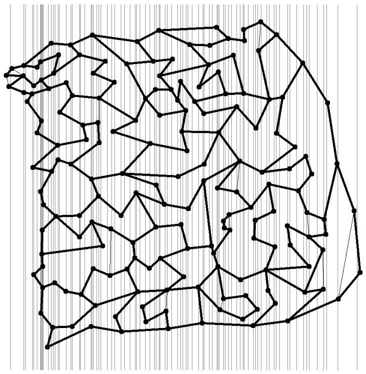

Fig. 22: Step 1, 91 strips.

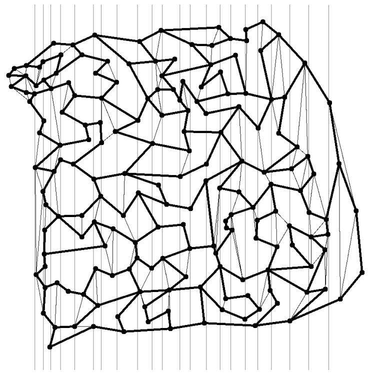

Fig. 23: Step 3, 23 strips.

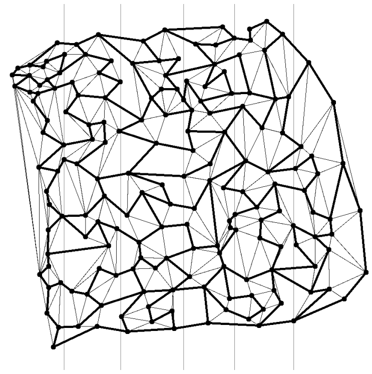

Fig. 24: Step 5, 5 strips.

1. *Within a uniform region, the shortest path does not bend.* Note that this does not mean that the shortest path between two points within the same area always is a straight line; see Fig. 25.
2. *Shortest paths do not intersect* unless they have the same distance to the source and can therefore be interchanged.
3. *On entering a new region the shortest path obeys Snell's Law of Refraction:* $\alpha_f \sin(\theta) = \alpha_{f'} \sin(\theta')$ where θ and θ' are the angles of entry and exit; see Fig. 27[3].
4. *The path entering edge e at the critical angle $\theta_c = \arcsin(\alpha_{f'}/\alpha_f)$ will continue along the edge* when going from a expensive to cheap region. Critical reflection occurs when the path re-enters the same face after travelling along the edge; see Fig. 25.
5. *Critical use occurs when an edge is cheaper than its surrounding faces.* It is possible that the path enters a low cost edge from face f at critical angle $\theta_c = \arcsin(\alpha_e/\alpha_f)$, then follows the edge and finally leaves the edge at critical angle $\theta'_c = \arcsin(\alpha_e/\alpha_{f'})$ through face f'; see Fig. 26.
6. *When the shortest path hits a vertex it can continue in any direction,* except that the path will not go directly back to the face where it came from.

3.3 Outline of the Algorithm

Because shortest paths so much resemble rays of light, the algorithm more or less simulates the propagation of a wavefront of light through the triangulation [15]. It is assumed that the source point is located on a vertex. If this is not the case, the triangulation can be easily modified in such a way that the source point is on a vertex. The wavefront is originated at the source point, and will initially spread in an circular way. It is propagated through the weighted triangulated map where Snell's law of refraction of light is applied each time a region boundary is crossed.

---

[3] To minimize the cost function $F(m_x) = \alpha_f \sqrt{(}m_x^2 + s_y^2) + \alpha_{f'} \sqrt{(}(t_x - m_x)^2 + t_y^2)$ of the path from s to t trough m, the equation $F'(m_x) = 0$ has to be solved. By applying some basic goniometric calculation this can be rewritten to Snell's Law. Note that in order to find m_x the following polynom in q of degree 4 has to be solved ($q = m_x/t_x$ and $\alpha = \alpha_f/\alpha_{f'}$):

$$q^4(\alpha^2 t_x^2 - t_x^2) + q^3(2t_x^2 - 2\alpha^2 t_x^2) + q^2(\alpha^2 t_x^2 + \alpha^2 t_y^2 - s_y^2 - t_x^2) + q(2s_y^2) + (-s_y^2) = 0$$

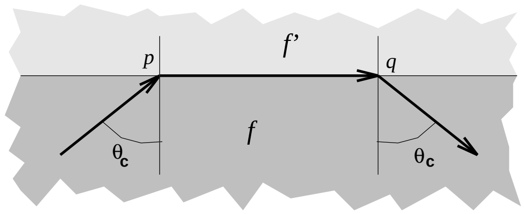

Fig. 25: Critical reflection.

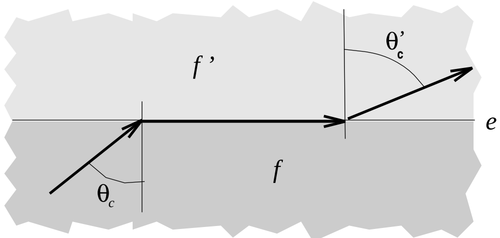

Fig. 26: Critical use.

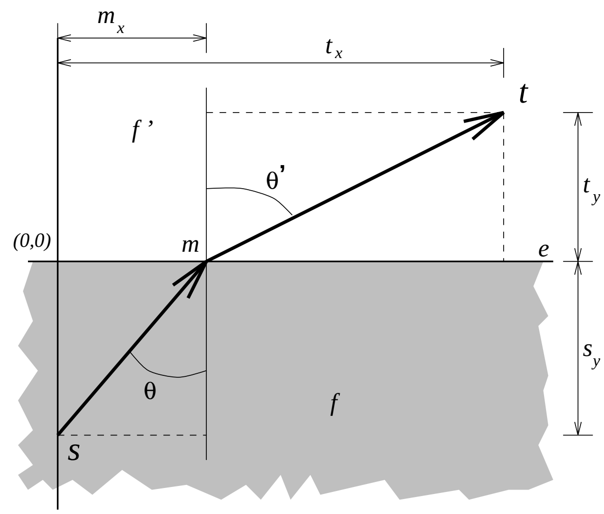

Fig. 27: Snell's Law of Refraction.

Instead of keeping track of the exact location of the wavefront, only the passing of the wavefront over certain locations is tracked. These locations are called *events*. Events are located at positions where the wavefront hits a vertex and at certain other important locations on the edges. All events which have been created, but which have not yet been overrun by the wavefront are kept in a priority queue sorted on their distance to the source. These events can be seen as the output of the algorithm because they can be used to efficiently trace back the shortest path from any point in the triangulation to the source.

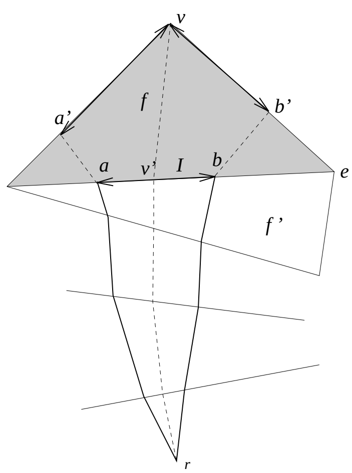

Fig. 28: Projection of interval.

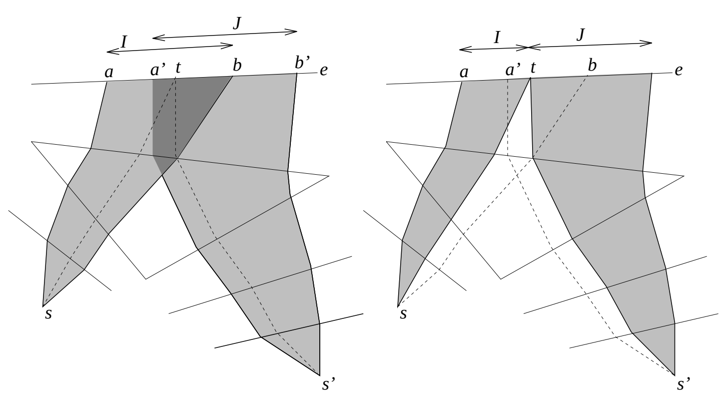

Fig. 29: Find the tie-point.

As can be seen in Fig. 28, it may be possible that a bundle of rays hits a vertex after propagation and that it need to be splitted. Another interesting situation occurs when there are two possible and efficient bundles of rays (or intervals) which reach, through a different route, a certain edge from the source, then it has to be established which point in the overlap of two intervals is equally well reached via both intervals. Between any two intervals there can only be one such point. This point, which is called the *tie-point*, can be found by performing a binary search on the overlap of the two intervals; see Fig. 29. The latter operation may be quite expensive as it involves solving a polynom of degree 4 (see Subsection 3.2) again and again for each iteration.

4 Implementation and test results

The implementation has been done in C++ with the aid of two C++ library packages: LEDA [16] for geometric data-structures and libg++ from GNU [24] for some other ADTs, e.g. the splay tree priority queue and complex numbers. The CCM-planning algorithms are connected to the Postgres/GEO++ GIS [23, 25] using a "close coupling" [7]. Full integration is planned in the near future. In Fig. 30, the large window shows the terrain and some optimal paths from three sources, the upper right window shows the iso-time diagram[4], and the lower right window shows the user interface of the analyses modules in which the parameters can be set.

Fig. 30: User interface of CCM analysis programs.

For the tests two maps have been used; a vector data set of 181 vertices, 524 edges and 344 faces, and a raster data set of 512×512 raster cells representing the same area. Two sources and two destinations were chosen and after each test the average of the four source-destination paths was computed. In Figs. 31 and 32 examples are shown of paths found with a raster (32-connected) and the vector approach. The measured cost of the paths are: *more-connected raster approach:* 27.25, 22.36, 21.76, 21.58, 21.54, 21.51 (for the connectivity values 4, 8, 16, 32, 64, 128), *extended raster approach:* 22.34, 21.75, 21.62, 21.57 (for $i = 0..3$), *quadtree raster approach:* 22.47, and the *vector approach:* 21.89[5].

[4] Note that the iso-time diagram of a scene with multiple sources results in a kind of Voronoi diagram.

[5] The coordinates of the raster and vector map are slightly different. Therefore, it is dangerous to compare the length of the vector path with the raster paths.

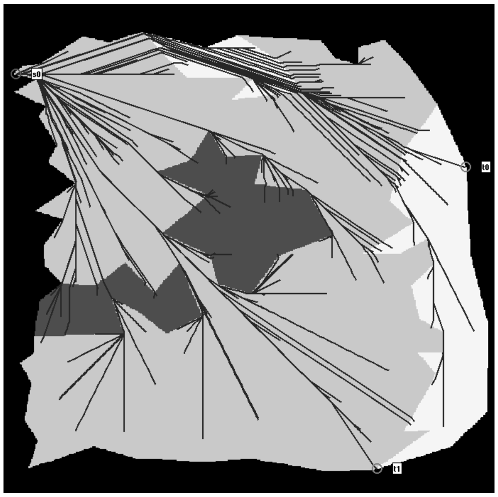

Fig. 31: Raster solution.

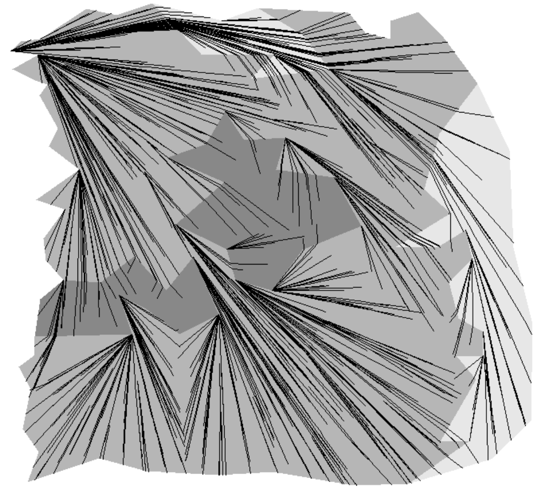

Fig. 32: Vector solution.

More test results, with different data sets and different resolutions, can be found in [24]. In these tests, CPU times, memory usage, and the quality of the paths are measured. The results clearly follow the theoretical worst case time bound of a Dijkstra search $O(e \log v)$ in a graph with v vertices and e edges. With the more-connected raster approach a connectivity rate of 16 provides the most satisfactory trade-off between the quality of paths and the CPU time. With the extended raster approach a number of intermediate search nodes of 1 provides the most satisfactory trade-off. The vector implementation turned out to be heavy resource user. However, this is not a surprise as the theoretical worst case time bound is $O(n^8)$ [15].

5 Hierarchical planning

All presented solutions require a lot of time and space to solve the problem when using realistic data sets, e.g. a 50K $\times$ 50K raster. A possible generic improvement to this is hierarchical planning: the search starts at a coarse level and is refined when required. This approach would not only reduce the required main memory, but also reduce computation time. A major drawback of most known hierarchical approaches is that they do not guarantee to find a global optimal solution. We present an algorithm that guarantees to find an optimal path. It is not an approximation or good path algorithm. A description for a raster implementation based on pyramid tree [19, 20] is given, but the same method can be applied in the vector case (based on an hierarchical constrained triangulation [26]).

The hierarchical approach is based on a hierarchical set of maps, see Fig. 33. Note that both the minimum and maximum cost values are stored in the course maps. This set starts with a coarse representation of the terrain. The next map is a more refined (detailed) version of the previous map, etc. In this way a whole list of maps exist where the last map has enough details for the application it is needed for. Assume that the shortest path from s to t has to be found.

In a hierarchical approach any cell $\mathcal{C}$ of a coarse version of the map will cover several different cells on a finer level. For every cell $\mathcal{C}$ we store the highest and the lowest resistance value of any underlying cells of the map. The following values are now calculated using a coarse version of the map:

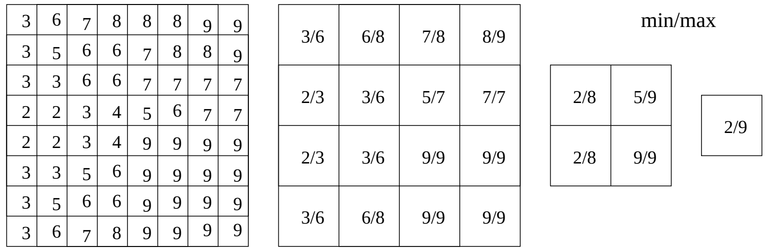

Fig. 33: Pyramid data structure with 4 levels.

1. The shortest path from s to t using the worst-case (highest) values for every part of the map. This value is an upper-bound for the real length of the shortest path from s to t.
2. For every cell $\mathcal{C}$, the shortest path from s to $\mathcal{C}$ and the shortest path from $\mathcal{C}$ to t are calculated using the best-case (lowest) values. The sum of these two values is a lower-bound for the shortest path from s to t via cell $\mathcal{C}$.

If the best path ever attainable via $\mathcal{C}$ (as described in item 2) is longer than the shortest path that was guaranteed to exist (as described in item 1), then it is sure that the shortest path will not go via cell $\mathcal{C}$. In the search for the shortest path, cell $\mathcal{C}$, and all its subcells, need not be considered anymore and can be discarded from memory. Next, all cells of the map which have not been discarded are replaced by a more detailed version and the whole process is repeated.

It is interesting to note that in normal cases this method discards all nodes that are outside an ellipse shaped cell around s and t. When s and t are close to each other then almost the whole map can be discarded in an early phase of the procedure, because the ellipse around these points will be small. If there is an obstacle between s and t which would force the shortest path outside this ellipse shaped cell then the cell is automatically adapted so that all possible shortest paths are included. An important issue is that only one version of the map needs to be in main memory at any time. When a part of the map is refined the refined version can be loaded from disk and replace the coarse version from memory.

Although there are similarities between quadtree approach and hierarchical approach, they are fundamentally different. The quadtree is very useful in case there are uniform parts in the terrain: its saves space and computation time. However, it can not avoid visiting a part which is unrelevant in finding the optimal path. However, the quadtree may be used to store the data pyramid efficiently.

6 Conclusion and Future work

We have implemented several raster-based and one vector-based cross country movement (CCM) planning algorithms. The raster-based algorithms have several advantages. They are relatively easy to implement and perform reasonably well. In theory however, an exact solution can only be found when both the *move-set* and the number of *search nodes* (related to the sampling resolution) in the area under consideration tend to infinity. This means that in practice only an approximation of the solution is found. The vector-based method we implemented has the advantage that it returns exact solutions. However, the performance of the vector-based method is not satisfactory.

The current raster- and vector-based approaches have major performance problems and are too slow to be used in an interactive GIS. New methods to improve the performance must be found. An extended raster approach which uses quadtree-based techniques to reduce the number of search nodes looks very promising. Further, we will implement and test the new hierarchical approach. Other types of solutions are based on using genetic algorithms [5, 13] and parallel algorithms [22].

The discrete vector approach, which avoids expensive trace-back computations to find tie-points, can be considered a combination of the extended raster approach and the vector approach. Each edge, side of a face, has a fixed number of search points which are used to create a connectivity graph. The shortest path can be found by applying the Dijkstra algorithm. This path can be enhanced locally by shifting points in such a way that on every crossing of an edge Snell's Law is obeyed [22]. Additional future work might include finding solutions for the following possible extensions to the CCM-problem:

- Find path from source to destination which passes a "variable" go-through location for generating alternative paths [3];
- Source or destination with different shapes: e.g. line, area, or multiple points;
- Path with non-zero width (e.g. a shape of 1 km width);
- Take crowdedness (multiple walkers) into account;
- Add the third dimension to the problem: the cost to travel uphill is higher than the cost to travel downhill, i.e. costs are direction sensitive (anisotropic).

References

[1] S. Baase. *Computer Algorithms Introduction to Design and Analysis*, chapter 4 and 6. Addison Wesley Publishing Company, 1988.

[2] L. Paul Chew. Constrained delaunay triangulations. *ACM*, pages 215–222, 1987.

[3] Richard L. Church, Scott R. Loban, and Kristi Lombard. An interface for exploring spatial alternatives for a corridor location problem. *Computers and Geosciences*, 18(8):1095–1105, 1992.

[4] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik 1*, 1:269–271, 1959.

[5] David E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

[6] M. F. Goodchild. An evaluation of lattice solutions to the problem of corridor location. *Environment and Planning A*, 9:727–738, 1977.

[7] Michael Goodchild, Robert Haining, and Stephen Wise. Integrating gis and spatial data analysis: problems and possibilities. *International Journal of Geographical Information Systems*, 6(5):407–423, 1992.

[8] P.E. Hart, N.J. Nilsson, and R. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science and Cybernetics*, SSC-4(2):100–107, 1968.

[9] B. F. Hobbs and A. H. Voelker. Analytical multiobjective decision-making techniques and power plant siting: A survey and critique. Technical Report ORNL-5288, Oak Ridge Nat. Lab., Oak Ridge Tennesee, February 1978.

[10] Dennis L. Huber and Richard L. Church. Transmission corridor location modeling. *Environmental Engineering ASCE*, 111(2):114–130, 1985.

[11] D. P. Huijsmans and A. M. Vossepoel. *Informatie in Gedigitaliseerde Beelden*, volume One: Introduction. RUL, January 1989.

[12] Scott T. Jones. Solving problems involving variable terrain. *BYTE Publications Inc*, pages 74–82, March 1980.

[13] Ir. D.W. Jonker. Genetic algorithms in planning. Technical report, FEL-TNO Divisie 2.1, June 1993.

[14] D. T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer and Information Sciences*, 9(3):219–242, 1980.

[15] J. S. B. Mitchell and C. H. Papadimitrou. The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the Association for Computing Machinery*, 38(1):18–73, January 1991.

[16] S. Näher. Leda user manual version 3.0. Technical report, Max-Planck-Institut für Informatik, Im Stadtwald, D-6600 Saarbrücken, 1992.

[17] Alan V. Oppenheim, Alan S. Willsky, and Ian T. Young. *Signals and Systems*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

[18] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry*. Springer-Verlag, New York, 1985.

[19] Hanan Samet. *Applications of Spatial Data Structures*. Addison-Wesley, Reading, Mass., 1989.

[20] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, Mass., 1989.

[21] M. I. Shamos and D. Hoey. Closest-point problems. In *Proceedings of the 16th Annual Symposium on the Foundation of Computer Science*, pages 151–162, October 1975.

[22] Terence R. Smith, Cao Peng, and Pascal Gahinet. Asynchronous, iterative, and parallel procedures for solving the weighted-region least cost path problem. *Geographical Analysis*, 21(2):147–166, 1989.

[23] Michael Stonebraker, Lawrence A. Rowe, and Michael Hirohama. The implementation of Postgres. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):125–142, March 1990.

[24] Joost van Bemmelen and Wilko Quak. Master's thesis: Cross country movement planning. Technical Report FEL-93-S205, TNO Physics and Electronics Laboratory, August 1993.

[25] Tom Vijlbrief and Peter van Oosterom. The GEO system: An extensible GIS. In *Proceedings of the 5th International Symposium on Spatial Data Handling, Charleston, South Carolina*, pages 40–50, Columbus, OH, August 1992. International Geographical Union IGU.

[26] J. M. Ware and C. B. Jones. A multiresolution topographic surface database. *International Journal of Geographical Information Systems*, 6(6):479–496, 1992.