5th INTERNATIONAL CONFERENCE
TRANSPORT SYSTEMS TELEMATICS TST'05

ZESZYTY NAUKOWE
POLITECHNIKI ŚLĄSKIEJ 2005
TRANSPORT z.59, nr kol. 1691

optimal route, turn penalties, Lee's algorithm, wave propagation algorithm

Rafał SZŁAPCZYŃSKI [1]

A NEW ALGORITHM FOR SEARCHING OPTIMAL PATH ON A RASTER GRID INCLUDING TURN PENALTIES – USE CASES

The paper introduces a new algorithm for finding optimal routes on raster planes. The algorithm takes advantage of its innovative data structure and results in minimizing the number of direction changes within a route. Presented use cases include finding optimal trajectories for sea going vessels, determining routes for vehicles on city plans and planning transport routes on varied terrain.

NOWY ALGORYTM ZNAJDUJACY OPTYMALNĄ TRASĘ NA SIATCE RASTROWEJ Z UWZGLĘDNIENIEM KOSZTÓW ZMIAN KIERUNKU – PRZYPADKI UŻYCIA

W referacie przedstawiono nowy algorytm znajdujący optymalne trasy na siatkach rastrowych. Dzięki zastosowanej strukturze danych jest on w stanie uwzględniać koszty zmian kierunku w ramach trasy. Możliwe zastosowania obejmują znajdowanie optymalnych trajektorii dla statków morskich, wyznaczanie tras pojazdów na planach miast oraz planowanie tras transportu na obszarach o zróżnicowanym terenie.

1. INTRODUCTION

Raster grids are a digital representation of planar data that is currently in use in a number of fields. One of the most common operations to be performed on a raster grid is to determine an optimal route between a start cell and a destination cell, which does not cross any obstacles. In reality only algorithms of linear time and space complexities are useful as only such may be accepted by a real-time system processing large numbers of cells. The big *O* notation will be further used for complexities, with *O(n)* indicating linear complexity.

The first solution to meet the condition mentioned above was the maze routing algorithm presented by Lee [3], often described as wave propagation process. To date Lee's algorithm and its variations are among the most widely used routing methods, with applications in maze games, VLSI design and road map routing problems. However, the original algorithm proposed by Lee has one serious drawback: it works only for the

[1] Faculty of Ocean Engineering, Gdańsk University of Technology, Gabriela Narutowicza 11/12, 80-952 Gdańsk, 347-14-24, rafal@pg.gda.pl

2-geometry grid plane (also known as the Manhattan geometry). Only recently has it been upgraded to higher geometries while sustaining the linear time and space complexities. This new solution has been proposed by Chang, Jan and Parberry [1]. Despite the major progress, the potential use of the improved Lee's algorithm has still some limitations. Both the original algorithm and the upgraded version tend to find the shortest path, which is not always identical to the optimal one. In presence of many obstacles the algorithms determine routes containing so many turning points and direction changes that they are unusable in practice. Also, the number of turns contributes to the total time spent traversing a route. Thus minimizing the number of turning points is a desirable objective. Although Chang, Jan and Parberry algorithm may brings significant improvements, its data structure based on that of the original Lee's method makes it unable to include the cost of a turn in path length.

The article proposes the solution to this problem. A new data structure has been designed so as to reflect the cost of all turning points in each cell's arrival time. An algorithm utilizing this structure has been implemented. The algorithm takes input parameters: user specified values of the turn costs (penalties) and returns the determined path.

2. THE MAZE ROUTING: LEE'S ALGORITHM

The popularity of this method lies in its simplicity and the guarantee to find the shortest path if one exists. Lee's algorithm is often described as a wave propagation process. Below its data structure is briefly described.

The cell map static data is stored in an array containing three fields for every cell:
PI (Passable / Impassable) – Integer number field, its value indicates whether a cell is passable or impassable (an obstacle). Its value is 1 for passable, infinity for impassable.
AT (Arrival Time) – Integer number field, its value is this cell arrival time – time needed to travel from the start cell to this cell.

The dynamic data is stored in two lists: L_1 and L_2 containing cell pointers or indexes in the cell array.

A cell array is initialised with appropriate *PI* values and *AT* values set to infinity. Two lists L_1 and L_2 are defined to keep track of the cells on the front of the wave (frontier cells) and their equal-distance step neighborhood cells respectively. Putting the source cell in the list L_1 initializes the search. After all neighboring cells in L_1 are included in L_2, the list L_2 is processed, so that an expanded wave front is found. Then every cell in L_1 is deleted if all of its neighboring cells have been processed (updated) and L_1 is updated by this new front of the wave. The search is terminated when the destination cell is found or (in case of the barrier of obstacles) when there are no more new cells to process (the last front of the wave has not generated any neighboring, obstacle-free cells).

3. CHANG AND JAN UPGRADED VERSION OF THE LEE'S ALGORITHM

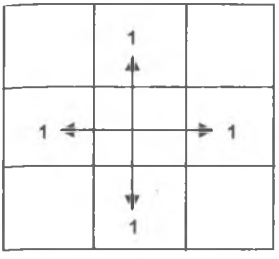

Fig.1. A 2-geometry grid neighbourhood

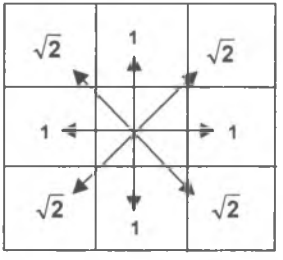

Fig.2. A 4-geometry grid neighbourhood

This method presented here improves 2-geometry grid routing to any higher geometry grid. Possible moves from a cell on the 2- and 4-geometry grid planes are shown in Fig.1 and 2 respectively.

The static data is stored in an array containing three fields for every cell:

PI (Passable / Impassable) – Integer number field, its value indicates whether a cell is passable or impassable (an obstacle). Its value is 1 for passable, infinity for impassable.

AT (Arrival Time) – Floating point number field, its value is this cell arrival time – time needed to travel from the start cell to this cell.

VIS (Visited) – Boolean field, its value indicated whether a cell has already been visited (inserted to *temp-list*) or not. True for a visited cell, false otherwise.

The dynamic data is stored in three circularly used lists: L_1, L_2, L_2 and an extra *temp-list*. The index of a list currently in use is stored in a *bucket_index* variable. All lists contain cell pointers or indexes in the cell array.

There are two major differences between the 2- and 4-geometry algorithms. First – additional lists are introduced to control the propagation speed in different directions (the additional two diagonal directions have single-step distance $\sqrt{2}$ instead of 1). Second, the *VIS* cell field is introduced to make sure each cell is inserted into temporary list exactly once. These two aspects make the algorithm keep the original linear complexities.

4. ROUTING ALGORITHM WITH TURN PENALTIES

It is assumed here, that whenever a sea-going vessel or a land vehicle changes its direction it results in a significantly longer passage time, than it would take to cover the same distance without altering the direction. In case of a ship the reason is mostly the dynamics of this maneuver (the fall in speed in particular). In case of a land vehicle, changing a road may produce this extra time. The additional time difference called the delay in the paper, should therefore be taken into account when determining a route. The fact of the direction changes being time consuming is the main reason for introducing a concept of the turn penalty. The other one is that the less complex routes (consisting of the lesser number of straight lines)

may be preferred for safety reasons. Hence the idea of the turn penalty parameters in a routing algorithm. The turn penalty parameter value might be set separately for each turn angle by the system operator. The exact value of the parameter might be equal to the delay time or larger – to enforce determining less complex routes.

4.1. DATA STRUCTURE

The static data is stored in an array containing three fields for every cell:
PI (Passable / Impassable) – Integer number field, its value indicates whether a cell is passable or impassable (an obstacle). Its value is 1 for passable, infinity for impassable.
GAT (Gate Arrival Time) – A sub-array of the floating point numbers. The size of the array is equal to the maximum number of the neighboring cells and is 8 for 4-geometry. Each field of this sub-array contains the gate arrival times for different incoming gates of the current cell. Gate arrival time is the time it takes to travel from the source cell to the current cell via certain neighbour of the current cell.
VIS (Visited) – Boolean field, its value indicates whether a cell has already been visited (inserted to *temp-list*) or not. True for a visited cell, false otherwise.
The dynamic data is stored in circularly used lists: $L_1 \ldots L_n$ and one extra temporary list *temp-list*. The number n of the lists that are used depends strictly on the maximum direction change cost specified and thus is configured indirectly via algorithm input parameters.

n=ceiling (maximum {single-step distances}+maximum {specified direction change costs}+1), where *maximum {single-step distances}* is $\sqrt{2}$ for 4-geometry.

4.2. ALGORITHM OVERVIEW

The complete formal description of the algorithm is provided in [4]. The key difference between the new solution and that proposed by Chang, Jan and Parberry is a replacement of each cell's *AT* (Arrival Time) field with *GAT* array. The idea of incoming gates arrival times makes it possible to take into account direction change costs without sacrificing the linear complexities that characterized the previous versions. Every time a cell data is updated, the arrival time of the appropriate gate is modified depending on the direction of the neighboring cell that has initiated the update operation. The new candidate value of the gate arrival time field is determined according to the following formula:

$$GAT_{new,\ j,\ gate\_number} = \text{minimum}\ \{GAT_{i,1} + distance_{i,j} + delay_{gate\_number,1}, \ldots, GAT_{i,8} + distance_{i,j} + delay_{gate\_number,8}\},$$

where: i and j are indexes of the neighboring cells, *gate_number* is the current gate of the c_j cell and numbers from 1 to 8 denote all gates of the c_i cell. Delay values (penalties) are equal to zero for two gates of the same direction and have appropriate parameter values d_1, d_2 or d_3 for two gates whose direction difference is 45, 90 or 135 degrees respectively. The present *GAT* value is replaced with the candidate value if the new value is lesser than the current one. The diagrams below illustrate the way the *GAT* array values of the wave front cells are updated.

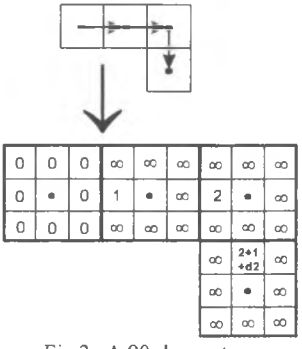

Fig.3. A 90-degree turn

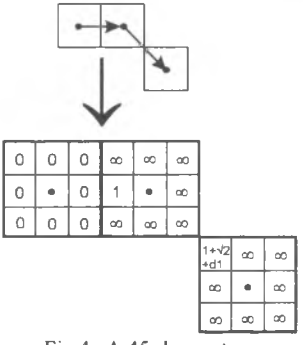

Fig.4. A 45-degree turn

4.3. COMPUTATIONAL COMPLEXITY

For each cell c_i, whose distance from the source cell is equal or lesser than that of the destination cell, the following actions are performed:

- each of its neighbours c_j is checked and possibly updated,
- for each of its neighbours c_j, each of the gates of the cell c_i is checked

This gives a total of $(2*\lambda)*(2*\lambda)*n$ steps for the worst case, where λ is a constant denoting geometry level (eight neighbours and eight gates for 4-geometry) and n *is* the number of cells, whose distance from the source cell is equal or lesser than that of the destination cell. Thus the computational complexity of the proposed solution is $O(n)$.

4.4. RESULTS FOR PROPOSED ALGORITHM VS. CHANG, JAN AND PARBERRY METHOD

Below example results for the new routing method and Chang, Jan and Parberry algorithm are presented. It has been assumed that costs of direction changes are: 1, 2 and 3 for 45, 90 and 135-degree turns respectively.

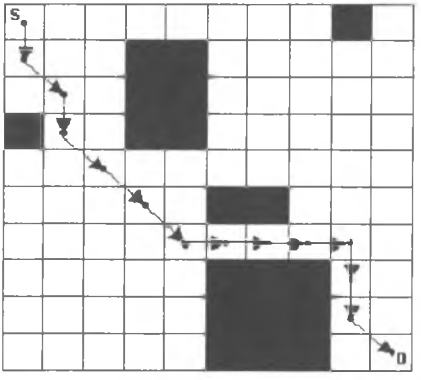

Fig.5. Route determined by the Chang, Jan and Parberry algorithm

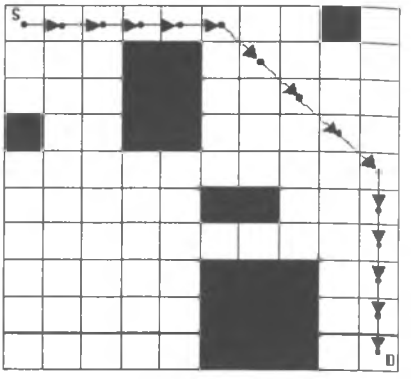

Fig.6. Route determined by the proposed algorithm with direction change costs (penalties) 1, 2 and 3 for 45, 90 and 135-degree turns respectively

In the figures above two different solutions of the same task are visualized. Proposed algorithm finds a route with two turning points as opposed to Chang, Jan and Parberry method, which determines a route with six turning points. Thus the total path lengths and penalties for both methods are:

For the Chang, Jan and Parberry route:
Basic path length = $8 + 5*\sqrt{2} \approx 15{,}07$
Total penalties = $5*d_1 + 1*d_2 = 7$
Total path cost = basic path length + total penalties $\approx 22{,}07$

For the proposed route:
Basic path length = $10 + 4*\sqrt{2} \approx 15{,}65$
Total penalties = $2*d_1 = 2$
Total path cost = basic path length + total penalties $\approx 17{,}65$

As illustrated above, taking a seemingly longer (3.8%) route results in a much lower (22%) overall path cost.

5. ALGORITHM' USE CASES

5.1. SEA ROUTING

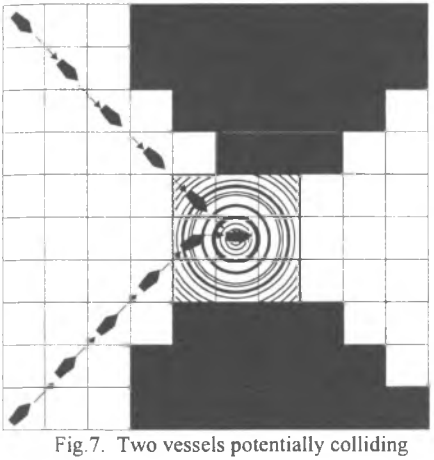

Fig.7. Two vessels potentially colliding

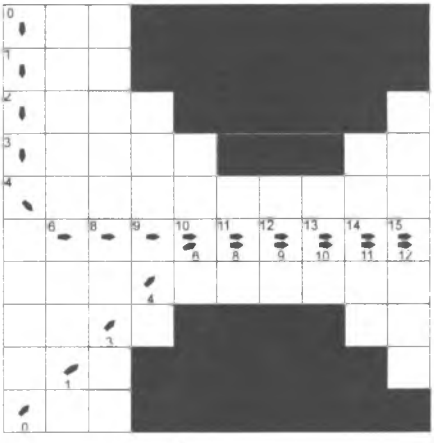

Fig.8. A route including course alteration determined for the second vessel by the proposed algorithm

The algorithm in the version described in Section 4. is a general-purpose routing tool. When ship routing is considered however, collision avoidance rules must be applied. Whenever there are two potentially colliding ships, the international regulations for preventing collisions at sea determine which one is to give way to the other. This decision is made, depending on the positions and courses of both ships. The resulting collision avoidance maneuver must fulfill the following conditions:

- the alteration of course must be large enough to be apparent to another ship – at least 15 degrees,
- the alteration of speed should only be applied if necessary, that is, if the alteration of course alone does not guarantee safe passage or course alteration necessary is too large to be accepted (economical reasons),
- ships should keep a safe distance, while collision avoidance action is taken.

In the situation presented in the Fig.7 there are two potentially colliding ships. According to the regulations, the ship in the upper part of the figure is obliged to alter its course, so as to give way to the other one. A route which includes such course alteration is shown in the Fig.8. The route has been determined by the modified version of the proposed algorithm. This version additionally checks, whether a cell arrival time might be updated with a new value, that is, whether a cell is not occupied by the other vessel's domain [2] in this particular time unit The numbers in the top and bottom part of the cells indicate cell arrival times - time units when the cells will be occupied by the give-way and stand-on vessels respectively.

5.2. ROAD ROUTING

Another possible application of the proposed algorithm is road routing for vehicles, when a start point and destination are specified. It is assumed here that changing a road is an action that may consume an additional amount of time as well as generate some stress. Therefore it is penalized with an additional time cost. This penalty is implemented in the same way as the turn penalty in the generic algorithm (Section 4.). For this, the cell map structure requires an additional data field within each cell, containing the number of the road that the cell is a part of. In the Fig.9., a route determined by an algorithm without the feature of road change penalty is shown. It is contrasted with the route, which takes into account the road change penalty (set to an equivalent of traversing two cell units) that is presented in the Fig.10. While the length of the first route is considerably smaller (14 cell units vs. 17 cell units), the second route one would be both traversed in a shorter time (19 time units vs. 20 time units) and easier to follow.

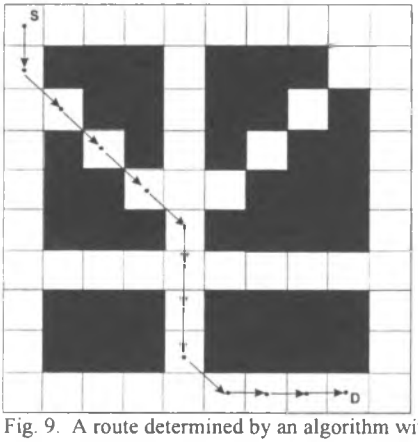

Fig. 9. A route determined by an algorithm without the feature of road change costs

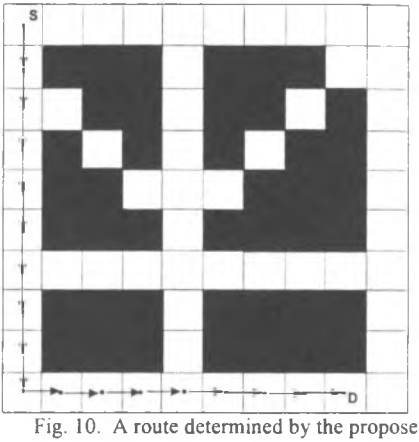

Fig. 10. A route determined by the proposed algorithm with road change cost (penalty) set to an equivalent of traversing two cells

5.3. ROUTING ON A VARIED TERRAIN

The division of all cells into passable and impassable may sometimes be too strong a simplification. Cells might be passable but may have different passage time or may be unfavourable due to some terrain qualities. In case of the sea routing, the heavy traffic in some areas might be such factor. In case of the road routing, the quality of the roads and their speed limits may significantly increase the time spent on traversing a route. In the example given (Fig.11 and 12), it is assumed that the grey cells have the passage time twice as long as the white ones due to the road quality / speed limits factor. Therefore a route determined by an algorithm which takes this into account (Fig.12) is considerably better than the route found by an algorithm, which ignores the aforementioned issue.

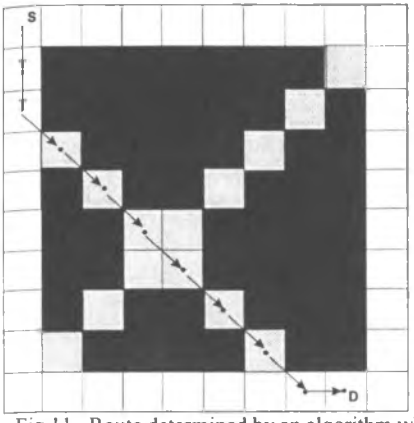

Fig.11. Route determined by an algorithm which does not take into account varied terrain

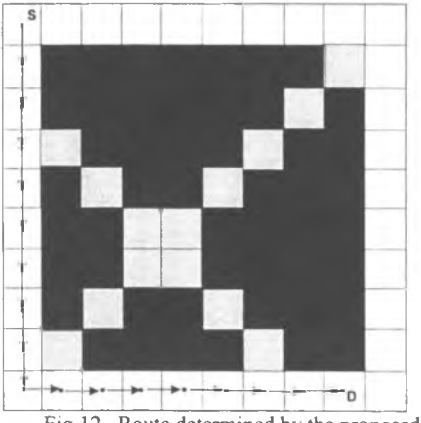

Fig.12. Route determined by the proposed algorithm with a varied terrain feature (grey cells have a double passage time)

The examples presented in subsections 4.1 – 4.3 do not exhaust the potential fields of application of the algorithm. Furthermore, different aspects and features presented in these examples might be combined within one routing tool, depending on the particular needs.

6. CONCLUSIONS

In the paper a general searching method on the raster plane was presented. Owing to its new data structure, the algorithm is capable of including costs of direction changes in the total cost of an optimal path, while keeping the linear computational time and space complexities. Therefore, when costs of the direction changes are significant, the benefits of using the solution proposed here might be considerable and hence - the solution - superior to those previously known. Such cases include sea routing and road routing among others.

BIBLIOGRAPHY

[1] CHANG, K. Y., JAN, G.E, PARBERRY, I. (2003). A Method for Searching Optimal Routes with Collision Avoidance on Raster Charts. The Journal of Navigation, 56, 371-384.
[2] FUJI, Y. AND TANAKA, K. (1971). Traffic capacity. The Journal of Navigation, 24, 543-552.
[3] LEE, C.Y. (1961). An algorithm for path connection and its applications. IEEE Trans. Electron. Comput., EC-10, 346-365.
[4] SZŁAPCZYŃSKI R. (2005) A method for searching optimal path on a raster plane including cost of direction changes. Polish Maritime Research 2005 / 3.

Reviewer: Prof. Bernard Wiśniewski