USE OF ALGORITHMS FOR PEDESTRIAN SAFETY AGAINST HARASSMENT IN MEDELLÍN

Esteban Vergara Giraldo	Miguel A. Cock Cano	Andrea Serna	Mauricio Toro	J. Sebastián Camacho
Universidad Eafit	Universidad Eafit	Universidad Eafit	Universidad Eafit	Universidad Eafit
Colombia	Colombia	Colombia	Colombia	Colombia
evergarag@eafit.edu.co	macockc@eafit.edu.co	asernac1@eafit.edu.co	mtorobe@eafit.edu.co	macockc@eafit.edu.co

https://github.com/QuitoTactico/ST0245-001

ABSTRACT

We need to find the shortest path to move from point A to point B, but without exceeding a harassment index limit. We are talking about a constrained shortest path problem, which has to be solved using a pathfinder algorithm. Is important to do this route in the most optimized way, the fastest to minimize the time it takes, but also this route must be secure for the users.

Like these, there are some problems that have to be solved by the constrained pathfinding, like finding the optimum route for an Uber travel, the fastest, but without using the streets with most traffic.

Keywords

Constrainted shortest path, street sexual harassment, secure-path identification, crime prevention.

1. INTRODUCTION

It is from common knowledge that everyone should take care of themselves in the streets, everyone can be harassed at any time while walking down the street, and apps like google maps only show you the optimal path to a set destination, but not the must secure one.

So with this project we are planning on helping people arrive safely at any place they want to go, and they can choose between being the fastest without an insecurity index, or being the safest, without exceeding a distance-to-travel limit.

1.1. Problem

We have to find the fastest path to travel from one point to another, but that path must not exceed a harassment risk.

Also we need to find a second path between these points, but prioritizing the security over the speed. Being the safest, but the path cannot surpass a maximum total distance limit.

1.2. Solution

We decided to use Dijkstra to search the shortest path, and the path with the least risk of harassment between two points. We choose this algorithm because it is way faster than a lot of other options we had, and the dijkstra algorithm is specialized in finding the shortest way in a graph, while the other options we had(dfs and ¿bfs?) were just algorithms that goes through the graph

1.3 Article structure

In what follows, in Section 2, we present related work to the problem. Later, in Section 3, we present the data sets and methods used in this research. In Section 4, we present the algorithm design. After, in Section 5, we present the results. Finally, in Section 6, we discuss the results and we propose some future work directions.

2. RELATED WORK

In what follows, we explain four related works to path finding to prevent street sexual harassment and crime in general.

2.1 Pathfinding algorithms for UAV traffic management

The unmanned aerial vehicles provide a wide range of services, but for their mobilization they need some digital guide, a route-to-travel generator. Unfortunately, the algorithms used in those UAVs usually don't consider the other UAVs' paths, and they can crash each other making their owners lose millions of dollars.

By the use of cooperative A*, with the UAVs communicating their situation to each other while they travel to reach their goals, this problem was solved by a Multi-Agent Pathfinding Algorithm, where the process considers the routes planned by the other individuals to avoid them. [1]

2.2 A* Pathfinding being used in modern games

Some of the most famous games use the commonly named "bots", or "NPCs" to make the game feel real, and natural. The bots are characters that move by their own, and act by themselves, without being handled by the gamer.

To reach and maintain that real sensation in games, the way in which the bots move is too important, it needs to feel natural, and normal, reaching goals and places in the most optimized form. For that, games like Age of Empires uses A* in their bot characters path-generating. When an enemy attacks you, he reaches you using the fastest possible path, not just walking around randomly until catch you, the enemy

uses a natural one. Things like this have made the game so popular among the people. [2]

2.3 Pathfinding algorithm for multi-objective route planning (SIGPA)

When we use Uber, or some transport app device, we may need to reach more than one single place, need to pass through at least some specific places, but the pathfinding just works if we use just two points. How to solve this?

Easy, use pathfinding between each pair of goals, in combination of all, and compare. But it's a slow method due to the big number of code iterations. So this A* based algorithm was created: SIGPA.
SIGPA, tries many cases at the same time, is a swarm intelligence, combinating the results in a Breath-First way instead of trying just the same case until the end, to later prove the others, a Depth-First way. Comparing using the A*combinatorial (Depth one) method, and SIGPA, SIGPA was three times faster. [3]

2.4 Wheelchair users routes planning using pathfinding

The wheelchair users also need pathfinding to know which are the best routes to travel in their conditions: With not too much traffic, ideally with low pendant, and optionally near to hospitals if something happens. But using A* or Dijkstra pathfinding is time-consuming, and in real map situations the space to analyze increases dramatically, also the time necessary to process, and the many restrictions used, the common solutions aren't appropriate.

In this case, they used Hierarchical Shortest Path Algorithm (HSP), that provides abstract plans of future routings, because the code executes itself since the mid of the midline between a and b too. Also exists an hybrid between HSP and A* called HSPA*, superior than the classic solutions in efficiency so far.

[4]

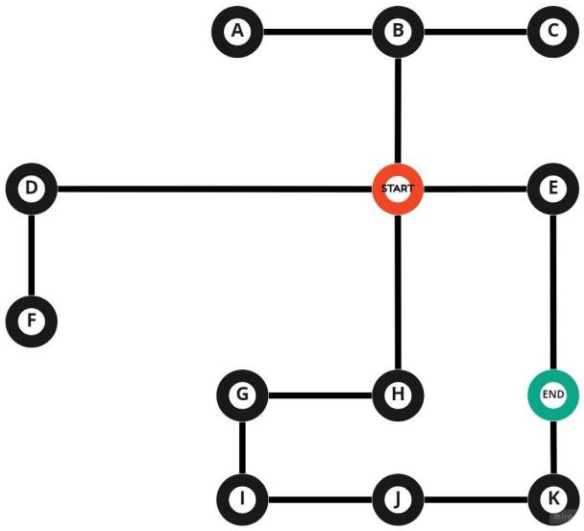
3. MATERIALS AND METHODS

In this section, we explain how data was collected and processed and, after, different constrained shortest-path algorithm alternatives to tackle street sexual-harassment.

3.1 Data Collection and Processing

The map of Medellín was obtained from Open Street Maps (OSM)[1] and downloaded using Python OSMnx API[2]. The (i) length of each segment, in meters; (2) indication whether the segment is one way or not, and (3) well-known binary representation of geometries were obtained from metadata provided by OSM.

For this project, we calculated the linear combination that captures the maximum variance between (i) the fraction of households that feel insecure and (ii) the fraction of households with income below one minimum wage. These data were obtained from the quality of life survey, Medellín, 2017. The linear combination was normalized, using the maximum and minimum, to obtain values between 0 to 1. The linear combination was obtained using principal components analysis. The risk of harassment is defined as one minus the normalized linear combination. Figure 1



presents the risk of harassment calculated. Map is available at Github[2].

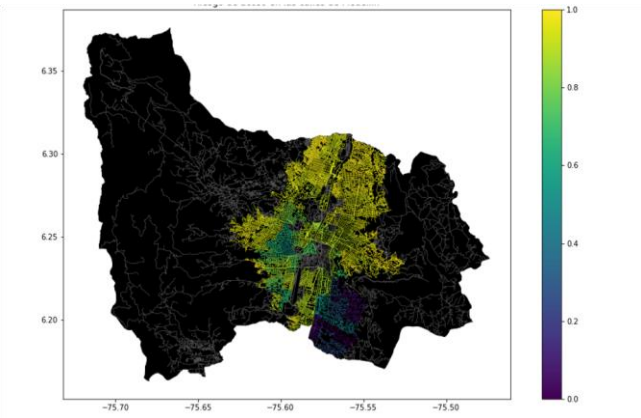


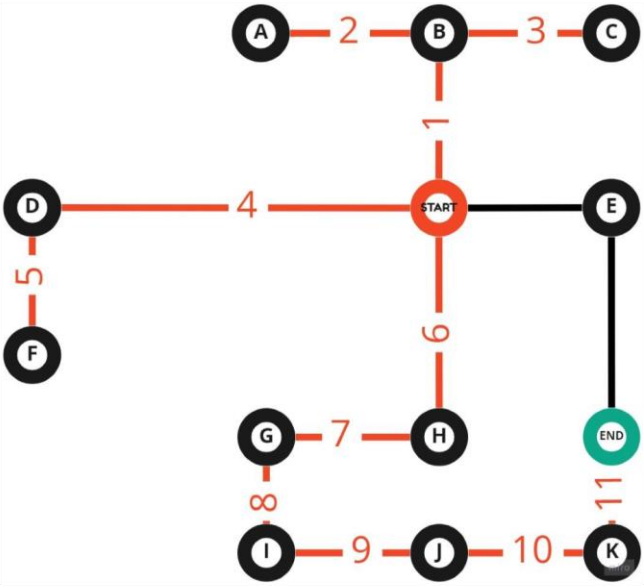
Figure 1. Risk of sexual harassment calculated as a lineal combination of the fraction of households that feel insecure and the fraction of households with income below one minimum wage, obtained from Life Quality Survey of Medellín, in 2017.

[1] https://www.openstreetmap.org/ [2] https://osmnx.readthedocs.io/

[2] https://github.com/mauriciotoro/ST0245Eafit/tree/master/proyecto/Datasets/

3.2.1 Depth-first search (DFS)

The depth-first search works when you set a starting point and a goal then you select a path to walk until you get to the finish or you find an ending point, if the first path did not work and you found other paths in your previous walk you go through them repeating the initial process or you get to the start and select another initial path and repeat again the initial process. The problem with depth-first is that it may not find the shortest path because if it finds one that works it will return it even if it is really long.



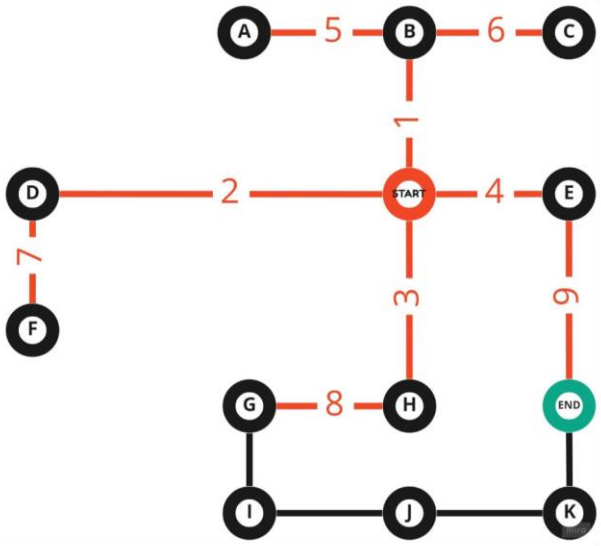
3.2 Constrained Shortest-Path Alternatives

In what follows, we present different algorithms used for constrained shortest path. [5]

Watching this example we can see how dfs works, selecting paths iteration by iteration until one works being in this case in the eleventh iteration and finding a really long path, but it may work if we wanted to know if there are other important variables in other ways.

3.2.2 Breadth-first search (BFS)

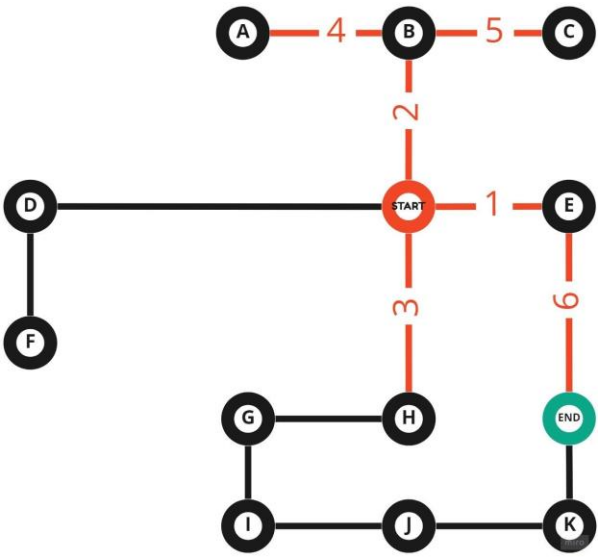
The Breadth-first search works as the following instead of taking a path and going through it until finding the end, you are going to take all the possible paths until and “number them by the amount of recursions that you have to make to end in there, until you find the end, this one not like depth-first will find the shortest path, the downside is that



Comparing this example of bfs with the dfs one, we can see a small improvement in the amount of iterations but one big step in the distance of the path found to the end.

3.2.3 Dijkstra

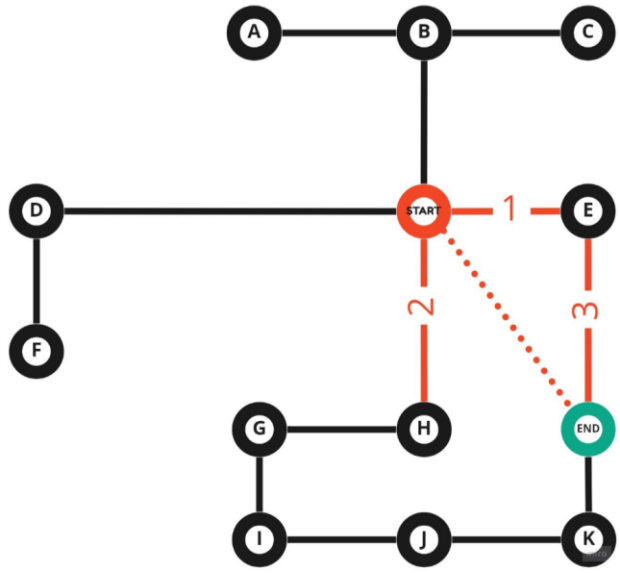
Dijkstra will work a little bit like breath-first by going through all the paths at the same time, the improved part is that it will select the path in order of which sums the least distance traveled, this one's improves the biggest problem of breath-first, in a real life example breath-first may take in count a road between cities if the person lives near one, but you wanted to go to the supermarket 5 blocks away so it is depending on the scale of the problem it may have a lot of iterations, so most of the times it should take a lot of time. illogical to think about it and that what dijkstra will skip making the code faster.



With dijkstra we improve in anything we do without too many iterations, we have the shortest path and we can still search for other paths if needed.

3.2.4 A*

A*(star) is the evolution of dijkstra. it combines everything listed before and more, the thing that it sums to the equations is that it will take into account not only the distance of the path but if it takes you further away or closely to the end.



With A star we optimized it to the top for the search of the shortest path which is really useful but there is the problem and it is that we don't have the opportunity to see other ways, and if we try to apply it we won't have the most secure path.

4. ALGORITHM DESIGN AND IMPLEMENTATION

In what follows, we explain the data structures and the algorithms used in this work. The implementations of the data structures and algorithms are available at Github[3].

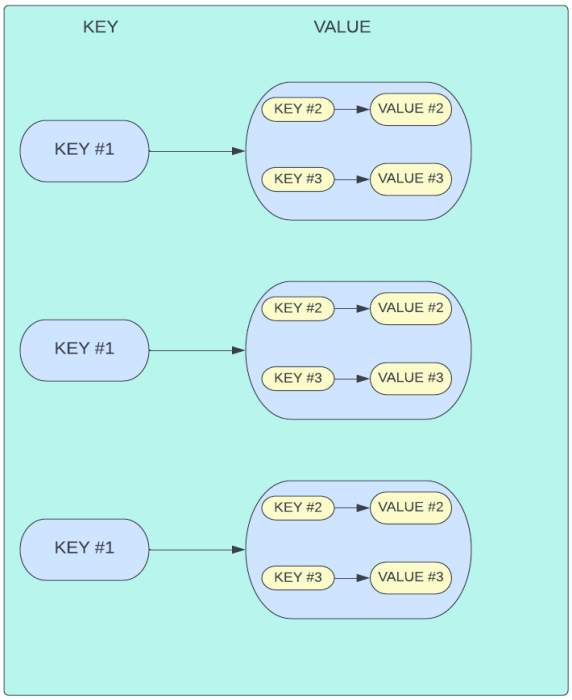
4.1 Data Structures

In this case we are using a dictionary as our data structure, specifically a pandas dictionary, but it works in the same way as a normal dictionary. A dictionary is an unordered collection of data values, used to store data values like a map, which, unlike other Data Types that hold only a single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized.

The key in this dictionary would be all the data that we have about a direction(length, origin, destination, one way, harassment risk and geometry) and the value would be the name of the direction.

The reason we used a pandas dictionary instead of a normal dictionary is that the one of pandas can be read by geopandas and be illustrated as a picture, which makes it way easier to see all the data we have.

Data structure is presented in Figure 2.



[3] https://github.com/QuitoTactico/ST0245-001

Figure 2: An example of a Pandas dictionary. each dictionary has a value which is another dictionary. it's a dictionary of dictionaries.

4.2 Algorithms

In this work, we propose algorithms for the constrained shortest-path problem. The first algorithm calculates the shortest path without exceeding a weighted-average risk of harassment *r*. The second algorithm calculates the path with the lowest weighted-average risk of harassment without exceeding a distance *d*.

4.2.1 First algorithm

In this project, with all the options of algorithms, we chose the dijkstra algorithm, the way this algorithm works is that given a graph and a source vertex in the graph the algorithm will find the shortest paths from the source to all vertices in the given graph.

Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a SPT (shortest path tree) with a given source as a root. We maintain two sets, one set contains vertices included in the shortest-path tree, another set includes vertices not yet included in the shortest-path tree. At every step of the algorithm, we find a vertex that is in the other set (set of not yet included) and has a minimum distance from the source.

Algorithm is exemplified in Figure 3.

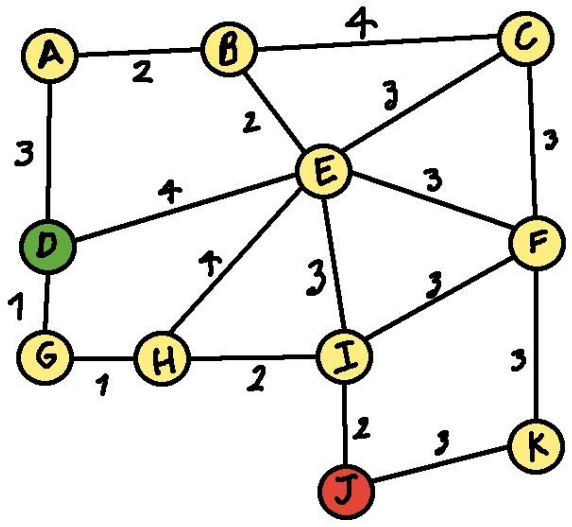


Figure 3: Solving the constrained shortest-path problem with Dijkstra.

4.2.2 Second algorithm

Explain the design of the algorithm to calculate calculates the path with the lowest weighted-average risk of harassment without exceeding a distance *d* and make your own figure. Do not use figures from the Internet, make your own. *(In this semester, the algorithm could be DFS, BFS, a modified version of Dijkstra, a modified version of A*, among others).* Algorithm is exemplified in Figure 4.

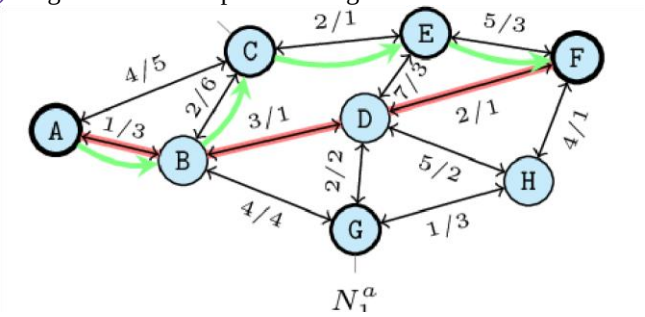


Figure 4: Solving the constrained shortest-path problem with Deep First Search (DFS). (Please, feel free to change this Figure if you use a different algorithm).

4.4 Complexity analysis of the algorithms

Explain, in your own words, the analysis, for the worst case, using O notation. How did you calculate such complexities? Please explain briefly.

Algorithm	Time Complexity
Name of the algorithm	$O(V^2*E^2)$
Name of the second algorithm (in case you tried two)	$O(E^3*V*2^V)$

Table 1: Time Complexity of the name of your algorithm, where V is… E is... *(Please explain what do V and E mean in this problem).*

Data Structure	Memory Complexity
Name of the data structure	$O(V*E*2^E)$
Name of the second data structure (in case you tried two)	$O(2_{E*}2_V)$

Table 2: Memory Complexity of the name of the data structure that your algorithm uses, where V is… E is...
(Please explain what do V and E mean in this problem).

4.5 Design criteria of the algorithm

Explain why the algorithm was designed that way. Use objective criteria. Objective criteria are based on efficiency, which is measured in terms of time and memory. Examples of non-objective criteria are: "I was sick", "it was the first data structure that I found on the Internet", "I did it on the last day before deadline", "it's easier", etc. Remember: This is 40% of the project grading.

5. RESULTS

In this section, we present some quantitative results on the shortest path and the path with lowest risk.

5.1.1 Shortest-Path Results

In what follows, we present the results obtained for the shortest path without exceeding a weighted-average risk of harassment *r* in Table 3.

Origin	**Destination**	**Shortest Distance**	**Without Exceeding *r***
Universidad EAFIT	Universidad de Medellín	??	0.84
Universidad de Antioquia	Universidad Nacional	???	0.83
Universidad Nacional	Universidad Luis Amigó	??	0.85

Table 3. Shortest distances without exceeding a weighted-average risk of harassment *r*.

5.1.2 Lowest Harassment-Risk Results

In what follows, we present the results obtained for the path with lowest weighted-average harassment risk without exceeding a distance *d* in Table 4.

Origin	**Destination**	**Lowest Harassment**	**Without Exceeding *d***
Universidad EAFIT	Universidad de Medellín	??	5,000
Universidad de Antioquia	Universidad Nacional	???	7,000
Universidad Nacional	Universidad Luis Amigó	??	6,500

Table 3. Lowest weighted-average harassment risk without exceeding a distance *d* (in meters).

5.2 Algorithm Execution-Time

In Table 4, we explain the relation of the average execution times for the queries presented in Table 3.
Compute execution time for the queries presented in Table 3. Report average execution times.

	Average execution times (s)
Universidad EAFIT to Universidad de Medellín	100.2 s
Universidad de Antioquia to Universidad Nacional	800.1 s
Universidad Nacional to Universidad Luis Amigó	845 s

Table 4: Execution times of the algorithm name *(Please write the name of the algorithm, for instance, DFS, BFS, a modified A*)* for the queries presented in Table 3.

6. CONCLUSIONS

Explain the results obtained. Are shortest paths significantly different from paths with lowest harassment-risk? How is this useful for the city? Are execution times reasonable to use this implementation in a real-life situation?

6.1 Future work

Answer, what would you like to improve in the future? How would you like to improve your algorithm and its implementation? Will you continue this projects by further working on Optimization? Statistics? Web development? Machine learning? Virtual Reality? How?

ACKNOWLEDGEMENTS

Identify the kind of acknowledgment you want to write: for a person or for an institution. Consider the following guidelines: 1. Name of teacher is not mentioned because he is an author. 2. You should not mention authors of articles that you have not contacted. 3. You should mention students, teachers from other courses that helped you.

As an example: This research was supported/partially supported by [Name of Foundation, Grant maker, Donor].

We thank you for assistance with [particular technique, methodology] to [Name Surname, position, institution name] for comments that greatly improved this manuscript.

The authors are grateful to Prof. Juan Carlos Duque, from Universidad EAFIT, for providing data from Medellín Life Quality Survey, from 2017, processed into a Shapefile.

REFERENCES

1. Ho, F. Salta, A. Geraldes, R. (AO). Multi-Agent Path Finding for UAV Traffic Management. *AAMAS '19: Proceedings of the 18th International Conference on autonomous Agents and MultiAgent Systems.* ACM Digital Library, 2019, 131-139.

   <https://dl.acm.org/doi/10.5555/3306127.3331684>

2. Xiao, S. Hao, C. *A*-Based Pathfinding in Modern Computer Games*. IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, 2011, 125-130.

   <https://www.researchgate.net/profile/Xiao-Cui-12/publication/267809499_A-based_Pathfinding_in_Modern_Computer_Games/links/54fd73740cf270426d125adc/A-based-Pathfinding-in-Modern-Computer-Games.pdf >

3. Ntakolia, C. and Iakovidis, D. *A swarm intelligence graph-based pathfinding algorithm (SIGPA) for multi-objective route planning*. Computers & Operations Research, VOL 133, 2021

   <https://www.sciencedirect.com/science/article/pii/S0305054821001350>

4. Yang S., Mackworth A.K. *Hierarchical Shortest Pathfinding Applied to Route-Planning for Wheelchair Users*. In: Kobti Z., Wu D. (eds) Advances in Artificial Intelligence. Canadian AI 2007. Lecture Notes in Computer Science, VOL 4509, 2007, 539-550.

   <https://link.springer.com/chapter/10.1007/978-3-540-72665-4_46>

5. Cuevas, D. Análisis de Algoritmos Pathfinding. Pontificia Universidad Católica de Valparaíso. 2013, 4-12.

   <http://opac.pucv.cl/pucv_txt/txt-5000/UCE5372_01.pdf>