Józef TOMASZKO[1]

Opiekun naukowy: Stanisław ZAWIŚLAK[2]

PORÓWNANIE ORAZ WIZUALIZACJA ALGORYTMÓW WYZNACZANIA ŚCIEŻEK W GRAFACH

Streszczenie: Praca dotyczy porównania algorytmów poszukiwania ścieżek w grafach oraz ich wizualizacji. W napisanej aplikacji generowane są grafy spójne, przedstawione w oknie graficznym programu. Rozkład wierzchołków na panelu wyświetlania jest równomierny. Rozważano algorytmy tzw. SPP: Dijkista's and Bellman-Ford's. Posiadając dane położeń wierzchołków grafu – zaimplementowano także algorytm heurystyczny A*. Dodatkowo rozważano algorytmy przeszukania, które także wyznaczają ścieżki. Przestawiono wizualnie uzyskane trasy. W tabelach zestawiono analizy czasu pracy programu. Podsumowano dokonane porównania.

Słowa kluczowe: algorytmy, generowanie, graf spójny

COMPARISON AND VISUALIZATION OF ALGORITHMS OF PATHS SEARCHING IN GRAPHS

Summary: The present paper presents the comparison of the algorithms which search for paths in connected graphs as well as visualization of these paths. The prepared application allows for generation of connected graphs, of moderate density - to present them in the graphic panel of the computer program. The distribution of vertices is even. The following algorithms were considered: so called performing SPP: Dijkista's and Bellman-Ford's ones. Based on the data related to the positions of graph vertices on the screen panel – it was possible to implement also the heuristic algorithm A*. Moreover, search algorithms were considered which also determine paths as additional result. All the paths were presented visually in the graphic panel. In tables, results of comparisons were listed. Some final conclusions were added.

Keywords: algorithms, generation, connected graph

1. Introduction

The basic task of graph theory is to study the properties of structures, called graphs. These structures are complex mathematical objects consisting of two sets (V, E). Elements of the set V are called vertices, and elements of the E set are called edges. These objects are related to each other, because set E is defined as a subset of $V \times V$, i.e. a set of pairs (the Cartesian product of sets) [5,7]. Simplifying the subject, it can

[1] Mgr inż., graduated from the University of Bielsko-Biala, Poland, email
[2] Dr hab. inż., prof. ATH, University of Bielsko-Biala, Poland, email szawislak@ath.bielsko.pl

be concluded that the vertices are connected to each other by means of edges. In contrast, in the sense of discrete mathematics, a graph is equivalent to: (a) incidence matrix and (b) relation. The edges may be undirected or directed, which in turn corresponds to a symmetrical or asymmetrical relationship. There are many ways to describe graphs. The graphical representation is the most transparent from the human point of view. It is customary approach that the vertices are represented as circles, dots or other pictograms connected to each other by straight lines or via curves of any shape (sometimes they are arcs or Bezier curves). Graph drawing is special branch of knowledge having its own annual conference. The rules for planar graphs are given in book [9]. Moreover, aesthetic graph drawing was analyzed by Professor Helen Purchase in many papers and conferences [10, 11]. Her tips were partly incorporated in the written software e.g. vertices should be evenly distributed in the visualization panel - what was done. General rules of creation algorithms for discrete problems can be found in [1]. Some papers were dedicated to finding paths in graphs [2,4]. Topic is still under investigations e.g. instead of fixed weight, random values are assigned. The comparison of different algorithms was discussed in [6] – however in our solutions visualisation and user friendly interface are additional indictor. Start and destination points are marked by means of mouse, changing vertices' colors.

Graphs are a very universal tool for solving problems related to searching. Usually we are looking for the shortest route, the fastest route or information about whether there is any route connecting two points. Knowledge, which flow from this research is used in a lot of areas of live. For example GPS application must have implementation of algorithms for search the best road to travel. Another use of the research is routing, i.e. searching not only the shortest but the fastest route for sending packets between computers in the network. In this case, the vertices are routers and the edges of the connection between them. Thanks to such solutions, the efficiency of computer networks has increased many times. Path-searching algorithms find their application also in entertainment. They are widely used in various games in which characters have to go to a designated place.

2. Algorithms for graphs without weight

2.1 Breadth-first search algorithm

The breadth-first search algorithm is one of the simplest methods of graph scanning. The search for the path starts from the selected starting point. Then go to the first vertex next to it. If it is not the destination, it has not been searched, and it is not in the queue to be searched, we put it in queue. Sequentially visit all the neighbors. If none of them is the end point, we continue to search for the element that is the first in the queue. The steps of the algorithm are repeated until the desired vertex is found. The result of the algorithm's operation is a root-length search tree at the starting point. However, we are not sure if the solution is the best in terms of path optimization.

2.2. Depth-first search algorithm

Depth-first searching is the second of the simplest algorithms for looking for a way in the graph. As in any algorithm, the search starts from the starting point. In the next step, go to the first vertex next to it. If it is not, then the target vertex has not been

searched, and it is not in stack of the vertices to be searched, we put it on the stack. Then we go to all neighbors of the starting point. If none of them is a destination, we continue to search for the item last added to the stack. The steps of the algorithm are repeated until the desired vertex is found. The result of the algorithm is the search tree deep into the root at the starting point. As with the Breadth-first search, there is no certainty as to the quality of the solution received. If the graph is connected, then each crawl vertex is visited only once as a result of searching through the above methods. In this sense, the algorithms are fully effective. If further searches continue, we can detect all the connectivity components of the analyzed graph.

2.3. Random step of depth-first or breadth-first algorithm

Searching randomly depth-first or breadth-first is a combination of the two mentioned algorithms. Also in this algorithm, the search starts at the starting point. Just like in previous algorithms, first go to the first adjacent vertex. Again, if this is not the desired vertex, it is no longer in the list of searched vertices, and it is not in the list of vertices to be searched, we put it in the list. Then we visit all the neighbors of the starting point. If none of the neighbors is the final destination, the next step is choose by lottery. The steps of the algorithm are repeated until the desired vertex is found. Again, we have no information about the quality of the solution found.

3. Algorithms for weighted graphs

3.1 A* algorithm

The algorithm A * is a heuristic algorithm for searching for the shortest path in a weighted graph. It is widely used in the field of artificial intelligence, and in computer games. The algorithm works by selecting vertices that have the best transition cost factor to bring us to the point you are looking for. To do this, visit each of the neighbors of the currently searched item in sequence and check their distance to the destination point. Then, the top is chosen, which has the best transition cost factor to get closer to the target, and the rest is added to the list of reserve tops. The distance of neighbors to the end point is checked again. Taking into consideration the new points and the reserve list, the best of the vertices is chosen. The steps are repeated until the final vertex is reached. The algorithm is complete and optimal. This means that it finds the shortest path connecting both vertices.

3.2 Dijkstra algorithm

The algorithm was developed by the Dutch computer scientist Edsger Dijkstra. It is used to find the shortest path between the start point and all other vertices in the graph with edges having non-negative weights. It is possible to perform such modification so that the graph stops working when it reaches the destination point. For this purpose, the algorithm places all the vertices adjacent to the starting point to the list, remembering which point it is possible to go to. Then go to the neighbor, to whom the cost of the transition is the smallest. Again, he places the vertices connected with him that have not been visited yet. Again, it selects the transition through the

edges with the lowest weights. This algorithm is greedy. This means that at every step, he is looking for a locally best transition to the next, yet unreached destination.

3.3 Bellman-Ford algorithm

An algorithm used to determine the shortest paths in a graph weighted from the selected vertex to all other vertices of the graph. The operation of the algorithm does not differ significantly from the Dijkstra method. The only difference is to reconsider the paths that lead through the vertices that are better than during the greedy approach. Therefore there may be edges with negative weights in the graph. The only limitation is the absence of cycles with a total negative weight.

4. Results of analyzes

The analyzes were carried out for five graphs with different parameters. Each of them has a different number of vertices and edges. In order to draw the right conclusions, it is worth collecting the results obtained, grouping them according to selected parameters and presenting them in a clear form. In the next step, the received data should be collated, looking for the dependencies resulting from the tables obtained in this way. The result of such action should be an answer to the question, which algorithm should be used depending on the current needs of the consumer.

4.1 Exemplary results of analysis: time [in milliseconds]

The exemplary graphs are shown in Figs. 1 and 2. The found paths are marked in other color than the graph vertices and edges. The performance of program is registered and stored. The reports were generated and the collected results are depicted in tables.

Table 1. Results of the analysis time in milliseconds for row of graphs with increased number of vertices n = 200 up to 2000

Algorithm \Vertexes	200	400	800	1200	2000
DFS	2,4033	6,0384	17,9056	82,6583	247,2479
BFS	2,0421	9,636	30,7057	189,988	766,7776
Random DFS or BFS	3,5708	7,4699	20,1265	81,0026	410,3915
A*	10,0309	9,1452	31,5553	54,7474	151,4847
Bellman-Ford	5,7837	26,5291	92,9959	242,6377	1402,9104
Dijkstra	5,7579	17,8989	55,3088	207,6891	1088,6863

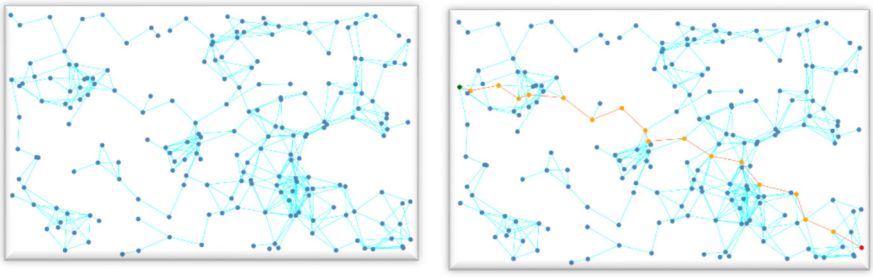

Figure 1. Exemplary graph of n=200 vertices (left) and found paths by means of A algorithm (right); starting and destination point are pointed by user by means of mouse*

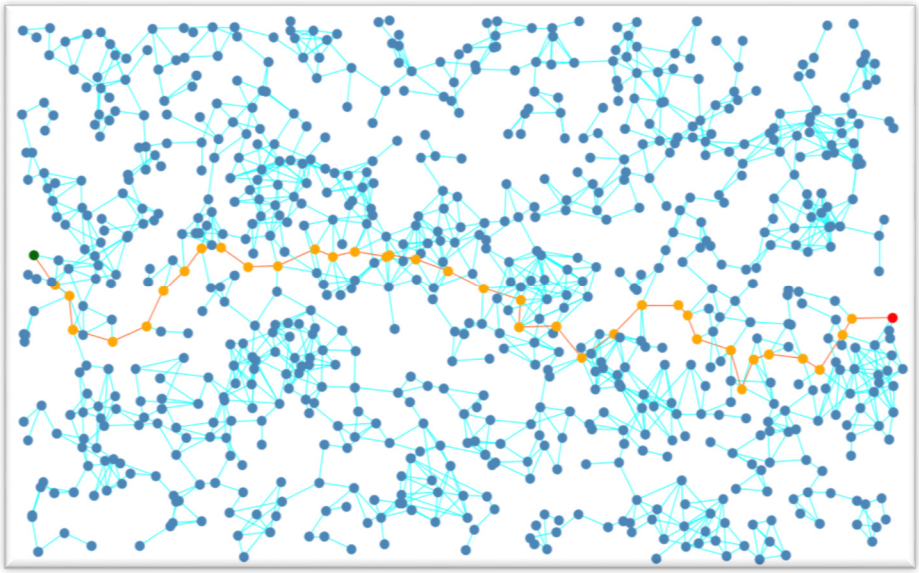

Figure 2. Found paths by means of A algorithm (right); starting and destination point are pointed by user by means of mouse, in the case n=800.*

The resolution is not sufficient for presentation of every vertex separately. However, due to the fact that the route is marked in different color as well as the start and destination points. The performance of consecutive

4.2 Exemplary results of analysis: number of completed steps

Table 2 Results of the number of completed steps

Algorithm \Vertexes	200	400	800	1200	2000
DFS	107	250	510	781	910
BFS	191	394	769	1199	1994
Random DFS or BFS	191	362	633	918	1669
A*	54	259	524	634	504
Bellman-Ford	262	676	1538	2223	3754
Dijkstra	199	399	799	1199	1999

4.3 Exemplary results of analysis: path length measured as number of edges

Table 3 Results of the path length

Algorithm \Vertexes	200	400	800	1200	2000
DFS	2370	2929	4322	6596	7306
BFS	1268	1551	1401	1521	1302
Random DFS or BFS	1454	2304	1853	2155	2212
A*	1222	1428	1351	1442	1247
Bellman-Ford	1222	1428	1351	1442	1247
Dijkstra	1222	1428	1351	1442	1247

4.4 Selection of the analysis the number of vertexes in path

Table 4 Results of the number of vertexes in path

Algorithm \Vertexes	200	400	800	1200	2000
DFS	33	52	124	180	219
BFS	17	26	36	39	32
Random DFS or BFS	22	42	50	60	63
A*	18	28	38	42	36
Bellman-Ford	18	28	38	42	36
Dijkstra	18	28	38	42	36

As can be seen from the tables above, the smallest increase in time in relation to the number of vertices in the graph is shown by the algorithm A*. It is worth noting that it has the most complex path estimation mechanics. This is reflected by comparing the ratio of times and quantities of iterations performed to those of other algorithms. Nevertheless, thanks to the promotion of transitions that best approximate

the search to the target vertex and the ignoring of non-decaying elements, the time invested in calculations is lower many times. Its 'competitors' in the form of Bellman-Ford and Dijkstra algorithms - achieve many times worse results. Therefore, it is the best way to look for the way, taking into account the time needed to find a solution. Another advantage of the algorithm is the fact that it always finds an ideal solution, which is not without significance for the optimization of travel time between sites (in case of real journeys). This makes the A * algorithm to be used in every situation, as long as it is possible to use.

In the case when a given graph has no edge weights, it is not so obvious to decide which way to look for the best path. The algorithm of graph passing deeply gives the best time result, but the results of the path length and the number of vertices are even several times worse than the other algorithms. Correlation of these results strongly limits the legitimacy of using this method of searching for a path. Its use only makes sense if the consumer needs information on the existence of any route between selected points.

The results obtained thanks to the breadth-first searching are very much resemble those obtained by the A * algorithm. This is a very good result for graphs without edge weights. This makes the algorithm useful when it is necessary to determine the path closest to the optimal one, and we cannot use weighted algorithms for this purpose. The third algorithm for graphs without weights cannot be unambiguously assessed. The randomness of its steps plays a large role in it, but in most cases he achieved results worse than the DFS algorithm and better than the BSF algorithm.

5. Conclusions

Own computer application was written for generation of simple graphs and searching the paths throughout the graph. The visualization is on an acceptable lever. The target was a possibility to compare the considered algorithms which is performed via generation of versatile protocols. These data were converted into the above placed tables which enables detailed comparisons described in the chapter 4.

REFERENCES

1. AHO A. V., HOPCROFT J. E., ULLMAN J. D.: Data structures and algorithms, Addison-Wesley, London, 1983.
2. AHUJA R. K., MEHLHORN, K., ORLIN J. B., and TARJAN R. E.: Faster algorithms for the Shortest Path Problem, Journal of Association of Computing Machinery, Volume 37, 1990, pp. 213- 223.
3. FRIEZE A.M., GRIMMETT G.R.: The shortest-path problem for graphs with random arc-lengths, Discrete Applied Mathematics, North-Holland, Volume 10, 1985, pp. 57-77.
4. GOLDBERG A. V., HARRELSON Ch.: Computing the shortest path: A search meets graph theory, Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, 2005, pp. 156-16.

5. KULIKOWSKI J.L.: Zarys teorii grafów, Państwowe Wydawnictwo Naukowe PWN, Warszawa, 1986.
6. TARAPATA Z.: Nieklasyczne modele i metody planowania tras w systemach wspomagania planowania ruchu: analiza złożoności, efektywności i zastosowań, Prace Naukowe Politechniki Warszawskiej, seria Transport, Zeszyt 60, 2007.
7. WILSON R.: Wprowadzenie do teorii grafów, Wydawnictwo Naukowe PWN, Warszawa, 2000.
8. ZOCHOWSKA R., Modelowanie wyboru trasy w gęstych sieciach miejskich. Zeszyty Naukowe Politechniki Śląskiej Nr kol. 1836, seria Transport, Zeszyt 71, Gliwice 2011.
9. TAKAO NISHIZEKI, SAIDUR RAHMAN: Planar graph drawing, World Scientific, 2004, 295p.
10. PURCHASE, Helen. Which aesthetic has the greatest effect on human understanding?. In: International Symposium on Graph Drawing. Springer, Berlin, Heidelberg, 1997. pp. 248-261.
11. PURCHASE H.C., COHEN R.F., MURRAY J.: Validating graph drawing aesthetics. In: International Symposium on Graph Drawing. Springer, Berlin, Heidelberg, 1995. pp. 435-446.