International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 11 (2020), pp. 3546-3550
 https://dx.doi.org/10.37624/IJERT/13.11.2020.3546-3550

Method for Optimal Route

Aleksey Nikolaevich Iliukhin[1], Lenar Ajratovich Galiullin[2]

[1]*PhD, Associate Professor, Kazan Federal University, Russia.*

ID Scopus: 56622584900, ORCID: 0000-0002-3333-5566,

[2]*PhD, Associate Professor, Kazan Federal University, Russia.*
ID Scopus: 39361435200, ORCID: 0000-0001-8640-1734,

Abstract

Describes the work on modern scientific achievements in graph theory, studying the developed algorithms, comparing their advantages and disadvantages, the reason for choosing the Ford-Bellman algorithm as a tool for solving the problem. The principle of the Ford-Bellman Algorithm and its structure were also described. The works on designing the architecture of Ford-Bellman algorithms in C # language, designing a class hierarchy for geometric objects modeling graphs, designing a model that implements three-dimensional geometric space were described. The architecture of the graphical interface and the hierarchy of its classes were also described. The paper describes the implementation of the model, which searches for the shortest route between any two points of the graph, the implementation of the graphical interface through which the algorithm is controlled. As a result of the work, the design and development of the application for finding the shortest route in a weighted connected pseudograph was carried out. The algorithm has been tested. The algorithm works and successfully performs the task. The algorithm was encapsulated and allocated to a separate class library, which makes it possible to use it in many other projects of different industries. It can be used for engineering calculations in which the problem can be represented as a graph; in tasks where it is necessary to find the shortest path from one point to another; in robotics and systems with artificial intelligence.

Keywords: route; method; automation; map; programming.

I. INTRODUCTION

The most visually obvious example of solving the optimization problem is to find the shortest road route between two settlements [1]. This problem is solved by minimizing a certain parameter. This parameter may vary [2]. It can be the travel time, the long travel, and the gas mileage, cost, etc. This task is relevant in itself, but with the development of computing power and artificial intelligence, it becomes possible to use graph theory to automatically determine the optimal route.

This problem can be described in terms of graph theory, where a graph models all possible paths from one settlement to another. In such a graph, the peaks will model points that can be reached, and the edges - the paths between any two points. Each edge has a weight; this is a numerical value that describes any path parameter.

Such a graph can be analyzed and, using combinatorial operations, select a list of edges that correspond to the set condition, that is, it will be a route that is minimal in weight [3]. Based on this information, a person and another automatic algorithm or neural network can make a decision.

II. METHODS

At present, Graph Theory is a dynamically developing branch of science. Algorithms in graph theory are well understood [4]. Studied such graphs as:

-oriented graphs;

non-oriented graphs;

-trees;

bipartite graphs;

Hypergraphs

-hypergraphs;

-flat graphs;

-planar graphs;

Open algorithms such as:

Ford-Bellman algorithm;

-prima algorithm;

Kraskal algorithm;

Magu-Weisman algorithm;

Hungarian algorithm;

-heuristic algorithm;

gamma algorithm.

Route search algorithms are used to find paths between geographic features on online cartographic services [5].

Examples of popular cartographic web services:

- Google Maps;

- Yandex maps;

- Yahoo! Maps;

- Bing Maps;

- Apple Maps.

Depending on the problem being solved, different shortest path search algorithms are used [6]. The main algorithms for finding the shortest path:

-Dijkstra algorithm - works with a weighted graph. In graphs with negative weights, it is not applicable, since it creates an infinite loop;

-wave algorithm - based on a breadth-first search, that is, by structure it is a recursive algorithm;

Johnson's algorithm - finds the shortest paths between all pairs of vertices of a weighted directed graph;

-Warshell-Floyd algorithm - finds the shortest paths between all the verins of a weighted oriented graph;

-search algorithm A*;

-Ford-Bellman algorithm - works with a weighted graph. Applicable to graphs with negative weights.

III. RESULTS AND DISCUSSION

The solution to the problem of finding the shortest route was decided to be carried out on a graph, which is a model of geographical space. As a model, a weighted connected pseudograph was chosen as the most suitable for modeling [7]. The vertices of the graph model all possible destination points, the edges - the existing paths between two points. And the weights of the edges are the distance between two points [8]. The graph includes only edges and does not include arcs, since it is assumed that for any path from point A to point B, there is a return path from point B to point A [9]. It is also assumed that the return path requires the same costs as the direct one [10]. As the weights of the ribs, the path length and the optimal route were used, this is such a route whose total path length is minimal.

The tool for solving the problem, it was decided to choose the Ford-Bellman algorithm for the following reasons:

-universality - the Ford-Bellman algorithm works in both a weighted and an unweighted graph. If all the edges are the same weight, then the algorithm finds the shortest path. If the weights of the edges are different, then the algorithm finds the least weight path;

-algorithm is simple to implement and does not require special programming techniques (for example, a recursive algorithm or additional sortings), which saves resources and improves productivity;

-reliability - the algorithm does not give closed endless cycles like the Taryan algorithm, which allows you not to overload the algorithm with preconditions for error checking;

- simplicity - the algorithm does not require additional data processing and preliminary operations such as Dijkstra's algorithm;

-algorithm can be encapsulated in a separate class library, which allows you to use it in other projects.

-algorithm is easy to test and verify its reliability.

The algorithm was divided into functional elements that work separately and are connected with the main object using composition.

FordBellmansAlgorithmBody is the main class containing the basic structural elements of the algorithm. It contains links to:

- ConverterOfGraphToMatrix - the object responsible for converting the graph into a matrix of weights;

- MakerOfLambdaTable - the object responsible for compiling the lambda table from the resulting weight matrix;

-ConstructorOfTrace - the object responsible for compiling the shortest route and issuing it as a collection of coordinates;

Also FordBellmansAlgorithmBody contains methods:

-DoMainWork - a method that accepts a collection of graph edges and the number of vertices that delegates commands to other objects, coordinates and exchanges data;

-SetBeginAndEndOfTrace - a method that sets from which point to start and at what point to end the calculation of the shortest route;

-GetShortestWay - getting the shortest route in the form of collections of Id points in it.

Detailed analysis of objects contained in the FordBellmansAlgorithmBody class:

-ConverterOfGraphToMatrix - takes a collection of graph edges and converts it into a two-dimensional weight matrix, where the first index is the rows of the matrix and the second index is the columns of the matrix. The row indexes of the matrix are the Id number of the point of origin of the edges, and the column indices of the matrix are the Id number of the point of end of edges. Since only edges are used and no arcs are used, the matrix is diagonally symmetric. Main fields:

-DefaultValue - The default value for the edge weight. By default, a sufficiently large number is used, the most disadvantageous for building a route.

-Matrix - matrix of weights in which weights of all edges are stored

The class also contains the Convert method, which receives a collection of edges in the graph and the number of vertices in the graph.

The method performs the following algorithm:

- an array of the required size is created and initialized;

- in the array, default values are set;

- all the weights of the edges from the resulting graph are read;

- all weights of the edges are transferred to the array.

MakerOfLambdaTable - takes in a matrix of weights and on its basis returns a lambda matrix. Where rows are vertex indices and columns are lambda indices. Main fields:

WeightToChooseMinimalN - a list of edges from which an edge with a minimum weight is selected.

RefToAdjacrncysTable - reference to the matrix of weights.

DefaultValue - the default value for the matrix cell is the same as in the weight matrix.

Matrix - the lambda matrix itself.

The class also contains the Calculate method, which receives a two-dimensional array of edge weights and converts it into a two-dimensional lambda array.

The method performs the following algorithm:

-obtaining a matrix of weights;

-initialization of the lambda matrix;

-setting the values of the cells of the lambda matrix by default;

-nulling the first element of the matrix;

-calculation of all cells of the lambda table using the Ford-Bellman algorithm;

ConstructorOfTrace - accepts a weight matrix, lambda matrix, default value. And on the basis of this data, it computes a collection of indices of points that make up the shortest route between two selected points.

Contains fields:

-WeighToChooseMininmalWeight - a collection containing the weight of the ribs from which the edge with the minimum weight is selected

-ShortestTrace - a collection of Id points that enter the shortest route between two selected points.

-RefToAdjacencyMatrix - link to the edge weight matrix.

-RefToLambdaTable - link to the lambda table.

-DefaultValue - default matrix cell weight value. Same as in other classes.

The class contains methods:

-SetParameters - sets the weight matrix and lambda matrix, as well as the default value of the matrix cell.

-Construct - creates a collection of points, which are the shortest route between two selected points.

The method performs the following algorithm:

-installation of selected points as the beginning and end of the route;

-search of the shortest path by computing the matrix of weights of edges and lambda matrix;

-creation of collections of points and adding to this collection of points belonging to the shortest route;

Interaction with the library is carried out using the Adapter pattern, which adapts the graph class to the format that the library accepts. To do this, the adapter implements the IRibzContainer interface, which must contain methods:

-GetVertex1IdOfRibN - returns the Id of the first vertex at the edge with the specified index;

-GetVertex2IdOfRibN - returns the Id of the second vertex at the edge with the specified index;

-GetWeightOfRibN - returns the weight of the edge with the specified index;

-GetCount - returns the length of the collection of edges;

This makes the class library encapsulated and independent of other classes, which makes it possible to use it in other, unrelated projects, regardless of their architecture.

IV. SUMMARY

To implement the algorithm, the Windows platform and the C # programming language were chosen. This choice was made for the following reasons:

- C # language is focused on work with stationary computers with high performance;

- C # language supports object-oriented programming and static typing;

- C # language contains libraries that implement default design patterns;

-Support for inheritance and interfaces;

- C # language has a developed development environment and tools for testing and debugging code;

- C # language has standard libraries for creating a graphical interface, which allows focusing on business logic of the program on distracting tasks;

-in C # language it is easier to implement the Model Visual Controls pattern;

-the presence of a garbage collector;

At the same time, the C # programming language has several disadvantages:

-the presence of the garbage collector leads to inevitable performance losses, which prevents the use of C # programs in real-time devices in which an immediate reaction to changes is needed;

- undeveloped elements of functional programming can serve as an obstacle to further work when introducing artificial intelligence systems into the program;

- incompatibility with other systems (for example, with Linux) may create problems in the future when it becomes necessary to use the program on other platforms;

All these shortcomings were recognized as insignificant, and they do not have much impact in the framework of this work.

The application architecture was designed based on the following patterns:

-singleton - used as a single class responsible for the storage and implementation of the work of all other classes;

- mediator - provides interaction between classes that implement interface elements and classes that implement models;

-command - provide encapsulation of individual commands thereby relieving the code of unnecessary intricacies;

-bridge - provides independence of classes that implement model from classes that implement visual;

-observer - provides notification of classes that implement model about events that occur in PaintArea in real time.

-template method - provides the versatility of the classes of the graphical interface.

factory method - provides a single way to create interface elements.

-adapter - provides the connection of different class libraries with each other.

V. CONCLUSIONS

The algorithm can easily be transferred to any programming language and any technological platform. It is universal and works with any weighted graph of arbitrary size. The algorithm is applicable in completely different optimization problems, not necessarily related to geolocation and has no restriction in application.

The study of this algorithm can be continued further in several directions:

- complication of the structure of the algorithm. The use and calculation of not one optimal solution, but several. Their sorting, enumeration, comparison and storage in memory;

-binding the algorithm with neural networks and training artificial neurons to recognize and remember the optimal solutions found by this algorithm.

ACKNOWLEDGEMENTS

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

REFERENCES

[1] Iliukhin AN, Galiullin LA. *Clusters in the production of ice.* Journal of Advanced Research in Dynamical and Control Systems. 2019;11(12):1371-1375.

[2] Valiev RA, Galiullin LA. *Information support of an automated personnel management system based on web technologies*. Journal of Advanced Research in Dynamical and Control Systems, 2019;11(8):2834-2837.

[3] Tazmeev AK, Galiullin LA. *Database of precedents for technological processes of internal combustion engines parts.* Journal of Advanced Research in Dynamical and Control Systems, 2019;11(8): 2795-2797.

[4] Khamadeev SA, Galiullin LA. *Database structure of the technological route for the ice production*. Journal of Advanced Research in Dynamical and Control Systems, 2019;11(8):2798-2800.

[5] Tazmeev AK, Galiullin LA. *Decision support system for the production of internal combustion engines.* Journal of Advanced Research in Dynamical and Control Systems. 2019;11(8):2801-2804.

[6] Khafizov AA, Shakirov YI, Valiev RA, Valiev RI. *Determination of regression materials mircohardness, processed by low-temperature plasma dependence on process conditions.* Journal of Physics: Conference Series. 2017;789(1): 10, article №012024,.

[7] Valiev RI, Shakirov YI, Khafizov AA, Valiev RA, Nuriev IM. Generalized current-voltage characteristics of electric discharge liquid cathode. InJournal of Physics: Conference Series 2017 Jan (Vol. 789, No. 1, p. 012067). IOP Publishing.

[8] Shakirova GY, Shakirov YI, Ilyin VI, Valiev RA, Drogaylova LN. Determination of steel bar dispersed mass in electric discharge with alternative electrode. InJournal of Physics: Conference Series 2017 Jan (Vol. 789, No. 1, p. 012050). IOP Publishing.

[9] Khafizov AA, Shakirov YI, Valiev RA, Valiev RI, Khafizova GM. Study of thermal and electrical parameters of workpieces during spray coating by electrolytic plasma jet. InJ. of Phys.: Conf. Ser 2016 Jan (Vol. 669, p. 012030).

[10] Shakirov YI, Valiev RI, Khafizov AA, Valiev RIO, Khakimov RG. Erosion of electrode metal in the electric discharge under the exposure of the electrolyte stream. InJournal of Physics: Conference Series 2016 (Vol. 669, No. 1, p. 012064). IOP Publishing.

BIOGRAPHIES OF AUTHORS

Aleksey Nikolaevich Iliukhin – Associate Professor, Naberezhnye Chelny Institute (branch) KFU/Higher Engineering School/Department of Information Technology and Energy Systems/Department of Information Systems, NI. Academic degrees: Candidate (technical sciences), specialty 05.13.06 – Automation and control of technological processes and production (by industry). Knowledge of languages: German (Basic Speaker), English (Basic Speaker).

Lenar Ajratovich Galiullin – Associate Professor, Naberezhnye Chelny Institute (branch) KFU/Higher Engineering School/Department of Information Technology and Energy Systems/Department of Information Systems, NI. Academic degrees: Candidate (technical sciences), specialty 05.13.06 – Automation and control of technological processes and production (by industry), the title of the dissertation "Automation of the technological process of diesel testing based on the fuzzy neural network method".