УДК 004.4

THE USE OF DIJKSTRA'S ALGORITHM IN NAVIGATION APPLICATIONS

D.V. Bilkevych, *T.A. Vakaliuk*[2]

[1] *Master's student of the Department of Software Engineering, Zhytomyr Polytechnic State University, Zhytomyr, Ukraine*

[2] *Doctor of Pedagogical Sciences, Professor, Head of the Department of Software Engineering Department of Zhytomyr Polytechnic State University, Zhytomyr, Ukraine*

ipzm241_bdv@student.ztu.edu.ua

Navigation applications are becoming increasingly significant in the modern world as they help users quickly find the optimal routes between various locations. Their applications span various tasks, from route planning for drivers to pedestrian walks and cargo transportation. The Dijkstra algorithm, created by Edsger Dijkstra in 1956, is one of the critical tools for solving the shortest path problem in such systems. Its efficiency in working with graphs, where each edge has a positive weight, has made it an essential component in modern navigation technologies.

The primary objective of this research is to analyze the efficiency and limitations of Dijkstra's algorithm within navigation systems, with a particular focus on its role in modern GPS and public transport applications. By assessing how the algorithm operates in real-world navigation contexts, this study aims to propose potential optimizations and explore alternative methods that could address Dijkstra's limitations in dynamic, real-time applications. This will contribute to developing more practical solutions in navigation systems, especially as they become more integral to daily life and complex logistical operations.

The Dijkstra algorithm [1] belongs to the greedy class. At each step, it selects the nearest vertex to the starting point and gradually expands the set of vertices with known minimum distances. Initially, the algorithm sets the distances to all vertices as infinite, except for the starting vertex, for which the distance is set to zero. Gradually, it calculates the shortest distances to neighbouring vertices until a path to the target vertex or all vertices in the graph is found.

In navigation systems, such as GPS or public transport route planning, road networks are modelled as graphs, with the vertices representing destinations (cities, intersections, stops) and the edges representing roads with certain weights (distance or travel time). The Dijkstra algorithm finds the shortest or fastest path between two points on the map, making it extremely important for everyday navigation.

Dijkstra's algorithm is applied in many modern navigation solutions. One well-known example is GPS navigators, which help millions of drivers daily find optimal routes. When a destination is entered, the navigator uses the algorithm to find the shortest path on the road network graph, considering current traffic conditions and obstacles. Depending on the edge weights (travel time, road length, congestion), the algorithm calculates the best route, minimizing the total travel time or distance.

In modern mobile applications, such as Google Maps [2], Dijkstra's algorithm is employed to determine optimal routes for drivers, pedestrians, and cyclists. These systems often consider additional travel conditions, such as time of day, weather, and potential delays, which enhance the accuracy and relevance of route recommendations. For instance, during peak hours, the system might suggest alternative routes to avoid traffic congestion, making navigation more dynamic and responsive to changing conditions.

In addition to its use in GPS navigators and mobile applications, Dijkstra's algorithm has proven effective in various public transport systems. For instance, it is utilized in applications that plan bus routes, trams, and trains by modelling the transport network as a weighted graph with the edges representing travel time or ticket cost. The algorithm helps commuters determine the most efficient combination of transit options and transfers. Similarly, logistics companies leverage Dijkstra's algorithm to optimize cargo delivery routes, ensuring that goods are transported through the shortest or fastest paths across complex road networks.

Moreover, advancements in the algorithm's application extend beyond just static navigation. For example, Dijkstra's algorithm has been adapted for indoor navigation systems, such as airports, hospitals, and shopping malls, where traditional GPS signals may be weak or non-existent. In these scenarios, the algorithm works with indoor maps and Bluetooth beacon technology to guide users through buildings, enhancing convenience and accessibility for users in large, unfamiliar spaces.

Another critical area where Dijkstra's algorithm demonstrates value is smart city infrastructure. Cities are increasingly adopting intelligent traffic management systems, where Dijkstra's algorithm directs vehicles through the least congested routes in real-time. Combined with data from IoT sensors, which monitor traffic flow, accidents, and road conditions, the algorithm enables city planners to predict and mitigate traffic congestion, enhancing urban mobility dynamically.

Despite its widespread use, Dijkstra's algorithm has certain limitations in real-time navigation. One of the primary drawbacks is that it does not consider live traffic changes, which can result in a theoretically shortest route but not optimal during the journey. Additionally, Dijkstra's algorithm calculates the shortest path to all vertices in the graph, even when the user only needs a route to one specific destination. This inefficiency leads to significant computational costs in large graphs, such as urban or national road networks, where processing time can become critical.

Another limitation is that Dijkstra's traditionally implemented algorithm does not account for real-time factors such as accidents, road closures, or temporary obstacles. These dynamic factors can substantially impact route efficiency. It is necessary to integrate Dijkstra's algorithm with supplementary data sources or real-time optimization techniques to enhance its performance in live navigation scenarios.

Dijkstra's algorithm remains a foundational and highly reliable tool in navigation applications due to its robustness and simplicity. However, its limitations in handling dynamic, real-time changes and its computational inefficiencies in large graphs highlight the need for further optimization. Future advancements in navigation systems may involve enhancing the Dijkstra algorithm with real-time data processing, predictive analytics (e.g., congestion forecasting), and hybrid approaches incorporating other algorithms like A* for specific applications.

As new technologies emerge, such as autonomous transportation systems, the demand for highly accurate and responsive navigation solutions will continue to grow. With its strong foundation, Dijkstra's algorithm will likely remain an essential part of these systems, especially when combined with modern data-processing tools and complementary algorithms. By addressing its limitations and adapting it for real-time applications, Dijkstra's algorithm can continue to play a pivotal role in the future of navigation technology, contributing to more efficient and intelligent route planning across various contexts. Additionally, as navigation systems increasingly integrate with AI and machine learning, Dijkstra's algorithm can be enhanced to make predictive, data-driven route suggestions, further expanding its practical applications in urban and intercity travel.

References:

1. *Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C.* Introduction to Algorithms (3rd ed.). 2009. MIT Press p. 643 – 671.
2. Google Developers Documentation. (n.d.). Google Maps Platform – Directions API. Retrieved from: https://developers.google.com/maps/documentation/directions