Konuralp Journal of Mathematics, 8 (2) (2020) 279-283

Konuralp Journal of Mathematics

Journal Homepage: www.dergipark.gov.tr/konuralpjournalmath
e-ISSN: 2147-625X

Classical and Heuristic Algorithms Used in Solving the Shortest Path Problem and Time Complexity

Öznur İşçi Güneri[1] **and Burcu Durmuş**[1*]

[1]*Department of Statistics, Faculty of Science, Muğla Sıtkı Koçman University, Muğla, Turkey*
[*]*Corresponding author*

Abstract

Optimization is the process of obtaining the most appropriate solution for a specific purpose and within the constraints given. In mathematical sense, it can be expressed as minimizing or maximizing a function. In this study, one of the optimization problems, the shortest path problem, is discussed. Classical and heuristic algorithms developed for solving shortest path problems are widely used. In this study, from the classical algorithms, Dijkstra, Bellman Ford, Johnson algorithms and from heuristic algorithms, Genetic, Scaling and Dinitz algorithms are examined. In this context, the complexities of the algorithms were investigated and comparisons were made. The results obtained from the examinations are presented with tables and graphs.

Keywords: *Optimization, Shortest path problems, Classic algorithm, Heuristic algorithms, Complexity.*
2010 Mathematics Subject Classification: *90C35, 90C59.*

1. Introduction

There are many optimization models within the scope of operations research. Optimization model is a mathematical expression of a real life problem. The best known of these are linear programming, integer programming, assignment problems, shortest path problems, knapsack problems, capital budgeting problems and travelling salesman problems.
In an optimization problem, f_i is the function, x_i is the decision variables and b_i is the source vector,

$$Max(or\ Min) f_0(x_1, x_2, ..., x_n) \tag{1.1}$$

Constraints:
$f_1(x_1, x_2, ..., x_n) \leq b_1$

$f_k(x_1, x_2, ..., x_n) \leq b_k$

$f_m(x_1, x_2, ..., x_n) \leq b_m$

are defined as. If all functions in the problem are linear, they are expressed as linear optimization problems. In addition, if some or all of the decision variables are to be integers, the model is converted to integer programming.
Network problems in optimization problems can also be formulated and solved as linear programming problems. Shortest path problems are one of the most known algorithms in network optimization. Many methods have been developed to solve these problems, both classical and heuristic.
In the shortest path problem, if the distance between any two points is d (x, y), the source x is the end of y. The aim of this problem is to find a series of branches that provide the minimum total distance from the desired source to the end [16]. The shortest path problem aims to find the shortest path between nodes, as in the case of a route or travelling salesman problem. However, it differs from them in two respects. First, there is no obligation to visit all nodes on the way to the destination. The second is that there is no return to the beginning. Therefore, the shortest path problem does not include a loop. In a problem with k nodes, the shortest path has a maximum of (k-1) paths. An exemplary shortest path algorithm can be graphically illustrated as in Figure 1.

Email addresses: **oznur.isci@mu.edu.tr (Öznur İşçi Güneri), burcudurmus@mu.edu.tr (Burcu Durmuş)**

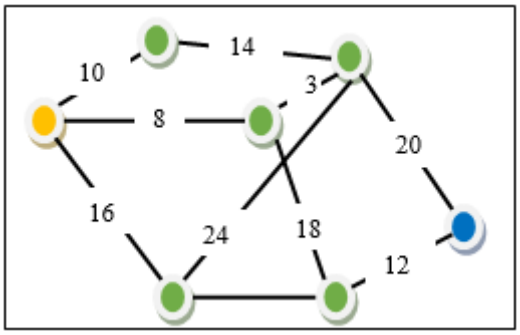

Figure 1. A sample shortest path problem with 7 nodes

Here the yellow node is the starting point and the blue node is the target point. The green nodes represent the routes to be taken. The flow is from the starting point to the destination and is non-iterative.
This problem is encountered in different ways in different fields. In computer networks, especially in wireless sensor networks with energy constraints, data packets need to be transported to the destination using at least sensors to ensure energy conservation. This is possible by finding the shortest distance route. Depending on the task given to the robot to reach the target with the shortest path is possible with the solution of the shortest path problem [20]. Shortest Path Problems are seen in many fields such as;

- Vehicle routing and route calculation [3],
- Navigation devices [15],
- Network systems [2],
- Robot movement system [8],
- Rubik's Cube solution,
- Touristic trip plans,
- Water quality monitoring networks [5]

Shortest path problems are finding the shortest path from one given point to the other. However, this path may not always be optimal. Let A-C-B be the shortest path between points A and B. The most cost-effective route can be A-D-B. In order to get rid of this situation, an accurate and complete analysis of the problem should be made and the aim of the problem should be determined exactly. Thus, the problem is solved as the shortest path or the most cost effective way.

2. Algorithm Complexity

The complexity of the algorithm, also called the run time, is a concept related to the run time of the algorithm. This time depends on both the structure and the size of the problem. As the problem space and the size of the problem increases, so does the time required. In complexity, the aim is to determine the maximum time required. Thus, complexity can be defined as a function that measures the solution time required as the size of the problem increases [1].
The time complexity of algorithms is covered by the terminology given in Table 1. The *O symbol* (Landau or Big-O notation) represents an estimate of complexity.

Table 1. Basic Complexity Terminologies

Complexity	**Terminology**	**Sample**
O(1)	Constant Complexity	Finding whether a number is odd or even
O(n)	Linear Complexity	Finding a value from the list
O(n^b)	Polynomial Complexity	Necessary problems in binary nested loop
O(b^n), b>1	Exponential Complexity	Calculation of Fibonacci numbers
O(logn)	Logarithmic Complexity	Searching for a value in an ordered array
O(nlogn)	nlogn	Combinational sorting algorithms
O(n!)	Factorial Complexity	Factorial calculus problems

Landau notation is used to calculate the worst-case analysis of the algorithm. The best algorithm is the fastest and least memory consuming algorithm. In this case, the worst-case analysis can carry out analyses to address this requirement.
In this study, we examined how to calculate the solution complexity of algorithms developed for the shortest path problems by Landau (Big-O) notation. We compared the results by considering 6 different methods, classical and heuristic.

3. Classical and Heuristic Algorithms

Several algorithms have been developed to determine the shortest path. In this study, Dijkstra, Bellman-Ford, Johnson classical methods and Genetic, Scaling, Dinitz heuristic methods are used. Time complexities for the solution of algorithms are calculated and presented in tables.

3.1. Classical Algorithms

3.1.1. Dijkstra's Algorithm

This algorithm calculates the shortest path from one point to all other points [4]. It is a simple but effective solution. Steps of Dijkstra's Algorithm:

Step 1. Set the starting node.
Step 2. Look at the neighbours of the starting node and go to the shortest distance point.
Step 3. Go to the nearest neighbour from the current point until all points are circulated.
Step 4. When all points are circulated, the algorithm is completed and the solution is found.
There are issues to be considered during the solution phase. Attention should be paid to the aspects of the problem structure. It can only be moved in the direction of the arrow and no return is possible. If there is a double direction, the two-way arrow is used. Graf values should be greater than '0'. Otherwise, another algorithm such as Bellman-Ford should be used.
Dijkstra's algorithm is based on the Greedy approach. So it chooses the best way of moving from one point to another. It does a local search with another phrase.

Time complexity of the algorithm:
n nodes including:

- There is 1 operation in the first step and (n-1) operation in the second step of the algorithm.
- Reviewing all points requires (n-1) operation.
- There are three operations in the calculations on the nodes: Comparison with other nodes, summing and assigning the best one [21]. Thus, the solution requires a total of $[1+n-1+(n-1)[n-2+1+4(n-1)] = 5n^2-8n-5]$.
- Total complexity is determined as $O(n^2)$.

In other words, the algorithm comprises 2 nested loops, each having an O(n) complexity. It can be said that the solution complexity is $O(n^2)$.

3.1.2. Bellman-Ford's Algorithm

Bellman-Ford's algorithm is one of the shortest path finding algorithms. The logic of the algorithm is similar to Dijkstra. However, it is simpler than that and is also suitable for graphs with negative edges. However, the cost of calculation is higher than Dijkstra. The steps of the Bellman-Ford's algorithm are:
Step 1. Set the starting node.
Step 2. The distance of each node to the other nodes is calculated.
Step 3. According to the results in step 2, the nearest neighbour values are found for each node and the shortest path is calculated.

Time complexity of the algorithm:
The Bellman-Ford algorithm includes (n-1) iteration, (n-1) node and (n-1) comparison per node for the control of each iteration [14]. Therefore, the time complexity of the algorithm is recorded as $O(n^3)$.

3.1.3. Johnson's Algorithm

Johnson's algorithm tries to reach the shortest distance by using Dijkstra and Bellman-Ford algorithms. It is known to give better results in small graphs. This is because the time complexity depends on the number of edges in the chart. The biggest advantage of the algorithm is that it generates solutions by updating the graphs with negative edges to positive weights by means of re-weighting. The steps of Johnson's Algorithm is given below [19]:
Step 1. A new node is added to the chart and linked to each other node in the chart.
Step 2. All nodes are updated with the weight re-weighting method.
Step 3. The node added in Step 1 is removed and the shortest path is calculated with the Dijkstra algorithm.

Time complexity of the algorithm:
The complexity of the algorithm using Fibonacci heaps is related to the implementation of the Dijkstra algorithm and Bellman-Ford algorithm for each corner. In this case, complexity is found as $O(nm+n^2 logn)$, where n is the number of nodes and m is the number of connections [13].

3.2. Heuristic Algorithms

3.2.1. Genetic Algorithm

Genetic algorithm is a random search method. Therefore, it works on a solution set instead of a single solution [17]. This method, based on Darwin's theory of evolution, is an evolutionary algorithm that models the biological process and optimizes functions. Stages of the algorithm are [12, 20]:
Step 1. Set the initial population size.
Step 2. The degree of well-being of each chromosome is determined (objective function values are calculated).
Step 3. Select the individuals to be multiplied (Repeat breeding, crossing and mutation operators are applied).
Step 4. New generation is produced by mutation method.
Step 5. The offspring are evaluated (Chromosomes with poor objective function are removed from the population).
Step 6. Individuals with the worst grades are replaced with offspring (steps 3-5 are repeated).
One of the advantages of genetic algorithms is the ability to learn and collect information. They allow working with a large number of parameters. However, it can achieve more than one optimal solution, not a single solution [6].

Time complexity of the algorithm:
Many parameters are required to calculate the time complexity of the genetic algorithm. Basically, the problem size is related to the size of the population and the number of generations [18]. In this case, the formula O(number of intermediate points * population size * number of generations) will give you the complexity of the formula.

3.2.2. Scaling Algorithm

This method introduced by Edmonds and Karp [9] is based on two assumptions. First the given numbers must be positive integers. Otherwise it is rounded to an upper value. Second the numbers of the problem must be limited to polynomials [11]. The steps of the algorithm for problem solving are as follows.
Step 1. Initially, only the top node of each value is calculated.
Step 2. The solution is improved by looking at the two top nodes.
Step 3. The process continues by looking at the higher number of nodes each time until all nodes are examined and the solution is found.

Time complexity of the algorithm:
Each complexity requires O ($\sqrt{}$ n) iteration if the complexity is calculated by considering the algorithm steps. Each iteration takes O(m) time. The total time is given in O($\sqrt{}n$m). With the completion of the operations in the last step, the total time is calculated as O($\sqrt{}n$mlogn).

3.2.3. Dinitz Algorithm

It is a polynomial algorithm created by computer scientist Dinitz [7] to calculate the maximum flow in the flow network. It is similar to the Scaling algorithm proposed by Edmonds and Karp.
Step 1. All nodes are assigned a level. Streams are determined according to these levels. At the end of the step, the new graph is drawn.
Step 2. The changed chart is assigned levels. Flow is determined according to levels. It is also checked whether there is more flow.
Step 3. If no further flow is possible, the process is completed.

Time complexity of the algorithm:
It takes O(m) time to move around each edge on the graph. If the flow is not finished, the increment will be required and it will take O(n) time for each side. In this case, it means 0(nm) time to complete the flow by going around all edges. A new level graph is created for each round and the number of levels increases by one to avoid duplication each round. So the outer loop runs for O(n) time. In this case, the working time will be determined as O(n^2m) [10].

4. Conclusion

The complexity of the problem often impedes the implementation of the appropriate classical algorithm. This is frequently seen especially in large-scale problems. Therefore, heuristic algorithms were needed and studies in this direction gained momentum. The same applies to the shortest path problems. In cases where methods such as Dijkstra and Bellman-Ford cannot provide a solution in a reasonable time, heuristic approaches are taken. Heuristic methods can overcome the limitations of traditional methods.
Some of the most popular methods for shortest path problems have been examined in this study. Table 2 shows the algorithms discussed in the study. The complexity calculation indicated in the table was calculated according to Landau (Big-O) notation. Calculations are based on algorithm steps and iterations. The results were found to be in parallel with the results in the literature.

Table 2. Complexity of Algorithms

	Algorithms	**Complexity**
Classic Algorithms	Dijkstra	O(n^2)
	Bellman-Ford	O(n^3)
	Johnson	O($nm+n^2logn$)
Heuristic Algorithms	Dinitz	O(n^2m)
	Scaling	O($\sqrt{}$ nmlogn)
	Genetic	O(no of intermediate points * population size * no of generations)

The shortest path problems and its solution methods are used in many areas of operations research. Therefore, solution speeds and therefore run-time complexity have an important place in the subject. In this study, information about classical and heuristic approaches which are commonly used in shortest path problems is given to readers and researchers. The importance of time complexity for classical and heuristic methods is emphasized.

Acknowledgement

We would like to thanks anonymous referees and editor of the journal.

References

[1] Bovet, D.P., Crescenzi, P., *Introduction to the Theory of Complexity*, Creative Commons, California, USA, 2006.
[2] Broumi, S., Bakali, A., Talea, M., Smarandache, F., Vladareanu, L., *Computation of Shortest Path Problem in a Network with SV-Trapezoidal Neutrosophic Numbers*, International Conference on Advanced Mechatronic Systems, Melbourne, Australia, 2016, 417–422.
[3] Chabrier, A., *Vehicle Routing Problem with Elementary Shortest Path Based Column Generation*, Computers and Operations Research, vol:33 (2006), 2972–2990.
[4] Cömert, Z., *En Kısa Yol Problemi*, Bilişim: Paylaşım, 2015, 1–6.
[5] Çetinkaya, C.P., Engin, A., *Su Kalitesi Gözlem Ağlarında Örnekleme için İzlenecek Yol Rotası Optimizasyonuna Bir Yaklaşım ve Gediz Havzasına Uygulanması*, AKU FEMUBID, vol:18 (2018), 265–275.
[6] Çunkaş, M., *Genetik Algoritmalar ve Uygulamaları*, Selçuk Üniversitesi Ders Notları, 2006.
[7] Dinitz, Y., *Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation*, Springer, 2006.
[8] Duchon, F., Babinec, A., Kajan, M., Beno, P., Florek, M., Fico, T., Jurisica, L., *Path Planning with Modified A Star Algorithm for a Mobile Robot*, Procedia Engineering, vol:96 (2014), 59–69.

[9] Edmonds, J., Karp, R.M., *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems*, Journal of the Association for Computing Machinery, vol:19 (1972), 248–264.
[10] Friis, J.M., Olesen, S.B.,*An Experimental Comparison of Max Flow Algorithms*, Aarhus University, Master Thesis, 2014.
[11] Gabow, H.N., *Scaling Algorithms for Network Problems*, Journal of Computer and System Sciences, vol:31(1985), 148–168.
[12] Gonen, B., Louis, S.J., *Genetic Algorithm Finding the Shortest Path in Networks*, International Conference on Genetic and Evolutionary Methods, Nevada, USA, 2011, 1–4.
[13] Gupta, A., *Shortest Paths and Seidel's Algorithm*, CMU, Lecture 4, 2017, 1–10.
[14] Medhi, D., Ramasamy, K., *Network Routing: Algorithms, Protocols, and Architectures*, Morgan Kaufmann Publishers, San Francisco, CA, 2007.
[15] Nazari, S., Meybodi, M.R., Salehi, M.A., Taghipour, S., *An Advanced Algorithm for Finding Shortest Path in Car Navigation System*, First International Conference on Intelligent Networks and Intelligent Systems, Wuhan, China, 2008, 671–674.
[16] Öztürk, A., Yöneylem Araştırması, Ekin Kitabevi, Bursa, Turkey, 2001.
[17] Siyah, B., *Genetik Algoritma Kullanılarak Noktadan Noktaya Yol ve Rota Planlama*, Software Engineer in Deep Learning, WordPress, 2014.
[18] Soltani, A.R., Tawfik, H., Goulermas, Y.J., Fernando, T., *Path Planning in Construction Sites: Performance Evaluation of the Dijkstra, A* and GA Search Algorithms*, Advanced Engineering Informatics, vol:16 (2002), 291–303.
[19] Taştan, O., Temizel, A.,*Accelerating Johnson's All-Pairs Shortest Paths Algorithm on GPU*, Ulusal Yüksek Başarımlı Hesaplama Konferansı, İstanbul, Turkey, 2017, 1–6.
[20] Uçan, F., Kaplan, G.B., Çaputçu, R., Haklıdır, M., Arar, Ö.F., *Taktiksel En Kısa Yol Problemlerinin Genetik Algoritma Yaklaşımı ile Çö z ü mü*, USMOS, Ankara, Turkey, 2007, 326–336.
[21] Yurtay, N., Ayrık İşlemsel Yapılar, Sakarya Üniversitesi Ders Notları, 2008, 14–20.