SCIENTIFIC OASIS

Decision Making Advances

Journal homepage: www.dma-journal.org
ISSN: 2956-2384

Development of an Algorithm and Software for Optimal Route Selection

Irakli Basheleishvili[1,*], Manana Chumburidze[1], Elza Bitsadze[1], Lia Janadze[1]

[1] Department of Computer Technology. Faculty of Exact and Natural Sciences, Akaki Tsereteli State University, Kutaisi, Georgia

ARTICLE INFO

Article history:
Received 5 September 2024
Received in revised form 27 October 2024
Accepted 26 November 2024
Available online 6 December 2024

Keywords:
Route; Optimal; Algorithm; Multi-criteria decision.

ABSTRACT

The paper deals with developing an optimal route selection algorithm and software, which, taking into account various criteria, ensures the adoption of efficient and economical route selection decisions. The algorithm is based on permutations for defining routes. It utilizes multi-criteria analysis methods such as TOPSIS and CRITIC, which enable the selection of optimal routes based on various factors, including cost optimization, time savings, safety measures, etc. The paper also presents the proposed algorithm's software, which helps to solve real-world problems related to route selection.

1. Introduction

The task of choosing the optimal route remains an important challenge in the modern world, as it serves to solve diverse and dynamic tasks. Many industries, including logistics, transportation, travel and commercial services, emergency management, and more, rely on this process to maximize operational efficiency and reduce financial costs [1-3].

Optimal route selection is associated with complex factors that may change according to constantly variable circumstances. For instance, optimal route selection in logistics may involve finding a path that reduces fuel costs while ensuring timely delivery. In the transportation sector, the goal might be to discover the fastest possible route. In emergency management, a quick and optimal route can become a defining factor, as it depends on how quickly an ambulance can reach its destination to save lives.

Route optimization requires consideration of many factors, including topological and geographic factors, traffic density and road conditions, weather conditions, cost-effectiveness, and more [4]. Therefore, the problem of selecting an optimal route remains relevant today as it is closely related to the effective functioning of many industries.

* *Corresponding author.*
E-mail address: basheleishvili.iraklil@gmail.com

https://doi.org/10.31181/dma31202564

The rapidly growing demand for comfortable and fast service, as well as adherence to the principles of green transportation and low energy consumption, forces companies to constantly seek more efficient and optimal routes. Thus, the task of choosing the optimal route is not only about economic efficiency but also about ecological and social responsibility [5-6]. This adds relevance to the research of the problem.

Given the relevance of the topic, our goal is to create an algorithm for selecting optimal routes that take multiple criteria into account. The contribution of the authors in the paper lies in the development of an optimal route selection algorithm and software that takes into account a variety of industrial needs. The novelty of the paper lies in the consideration of multiple criteria, which makes the algorithm more tailored to specific sectors such as logistics and emergencies. The proposed algorithm can help companies reduce costs, increase safety, and improve customer satisfaction, significantly enhancing the relevance of the research and the practicality of its results.

The remainder of the article is organized as follows: Section 2 defines the problem. Section 3 presents the route selection algorithm, Section 4 discusses the software implementation of the algorithm, and Section 5 contains conclusions and future research directions.

2. Problem Definition

The problem of selecting an optimal route involves the processes of analysis and decision-making aimed at identifying the best route based on specific criteria. The route selection problem is defined under the following conditions: we have a single starting location from which the route begins and must also end. Additionally, there are several additional locations that the route must include. The goal of the route selection process is to return to the starting location most economically and efficiently while visiting all necessary locations.

The task involves the following aspects:

i. *Starting and ending location* – The route begins and ends at a specific starting location.
ii. *Additional locations* –Several locations must be visited, increasing the complexity of the route. Each additional location must be visited once before the route returns to the starting location.
iii. *Optimization objectives* – It is important to consider several key objectives during the route optimization process, such as:
    - *Cost minimization* – Fuel consumption and transportation costs, which significantly affect the outcome.
    - *Time reduction* – The length of the route and transportation duration, which are directly related to customer satisfaction.
    - *Safety assurance* – Selecting a route that maximizes the safety of passengers and transported goods.

As it appears, this problem is similar to the traveling salesman problem (TSP) [7,8], which is a classic combinatorial optimization task that seeks to find a route through a set of cities, visiting each city exactly once and returning to the starting city while minimizing the total route length. Considering that our idea is based on route selection while taking into account multiple different criteria, this problem will be derived from the TSP into a multi-criteria decision-making problem. Solving the problem requires the use of optimization algorithms that will allow us to find the most efficient route while considering various criteria.

3. Route Selection Algorithm

The proposed algorithm for route selection is based on the permutation [9,10] of route combinations and the determination of corresponding distances, as well as multi-criteria decision analysis methods technique for order preference by similarity to ideal solution (TOPSIS) and criteria importance through intercriteria correlation (CRITIC). The TOPSIS method is used for ranking alternatives, while the CRITIC method is employed for calculating the weights of the evaluation criteria. The CRITIC and TOPSIS methods were chosen for their capabilities, which assist in optimizing the route selection process within the context of multi-criteria decision analysis.

The use of the CRITIC method, in this case, is justified by its ability to objectively determine the weights of criteria based on their variation and correlations. The CRITIC method considers the variational range of the data for the criteria and their correlations, indicating the informational significance of each criterion [11-13]. Consequently, this method allows significant criteria to receive greater weight, while less important criteria receive less, thus reducing subjectivity and ensuring a more objective evaluation.

The selection of the TOPSIS [14] method is based on its ability to evaluate alternatives based on their proximity to positive ideal and negative ideal solutions. The principle of TOPSIS is founded on the idea that the best alternative should be close to the ideal (best) solution and as far as possible from the anti-ideal (worst) solution [15,16]. This unique approach, which is based on the distance between positive and negative ideals, occupies a significant place in multi-criteria assessment. It greatly simplifies the process of ranking alternatives [17,18].

Combining these two methods enables the optimal selection of routes to be maximally accuracy-oriented and scientifically justified. Thus, the combined use of the CRITIC and TOPSIS methods reduces the risk of subjectivity and allows for a more structured assessment of alternatives, which overall provides a high level of objectivity and reliability in the decision-making process [19,20].

The optimal route selection algorithm we propose includes the following general stages.

Step 1. *Definition of locations and creation of their distance matrix*. In the distance matrix (Table 1), $d_{ij}(i, j = 1, \ldots, n)$ defines the distance from location *i* to location *j*. Naturally, the values of the elements located on the main diagonal of the matrix will be 0. This matrix is symmetric since $d_{ij} = d_{ji}$, meaning that the distance from *i* to *j* is the same as from *j* to *i*.

Table 1
Distance matrix

	L_1	L_2	…	L_n
L_1	d_{11}	d_{12}	…	d_{1n}
L_2	d_{21}	d_{22}	…	d_{2n}
…	…	…	…	…
L_n	d_{n1}	d_{n2}	…	d_{nn}

Stage 2. *Formation of possible routes*. It involves the creation of all possible routes and the calculation of their corresponding distances. The search for all possible routes and the calculation of their respective distances are based on permutations. The pseudocode for the search is provided below:

Algorithm DetermineRoutesDistances (Locations, distance_matrix)
1. Initialize distance_list to an empty list
2. For each permutation of Locations (permutations(Locations)) do:
3. Set current_distance to 0

```
    For i from 0 to length(route) - 1 do:
        If i == length(route) - 1 then:
            current_distance += distance_matrix[route[i]][route[0]]
        Else:
            current_distance += distance_matrix[route[i]][route[i + 1]]
    Append (route, current_distance) to distance_list
Return distance_list
Function permutations(array)
If length(array) == 0 then:
    return [[]]
If length(array) == 1 then:
    return [array]
result = []
For each element in the array do:
    remaining_elements = array without element
    For each perm in permutations(remaining_elements) do:
        result.append([element] + perm)
return result
```

Stage 3. *Definition of the decision matrix*. The decision matrix (Table 2) consists of alternatives and evaluation criteria, where x_{ij} (i=1,…,k, j=1,...,m) represents the evaluation of the route R_i according to the criterion C_j. The alternatives are the routes defined in *Stage 2* as $\{R_1, R_2, \dots, R_k\}$. The evaluation criteria are $\{C_1, C_2, \dots, C_m\}$. In this case, the first criterion C_1 represents the total distance of the route, which is calculated during the determination of all possible routes (*Stage 2*). The other evaluation criteria depend on the user's preferences regarding which criteria they wish to consider for optimal route selection.

Table 2
Decision matrix

	C_1	C_2	…	C_m
R_1	x_{11}	x_{12}	…	x_{1m}
R_2	x_{21}	x_{22}	…	x_{2m}
…	…	…	…	…
R_k	x_{k1}	x_{k2}	…	x_{km}

Stage 4. *Data collection*. After forming the decision matrix, it is necessary to fill it with data. This stage involves evaluating each alternative according to the criteria. In our case, in addition to the first criterion (C_1) (which is the total distance of the route), we manually gather this data input.

Stage 5. *Determining the weights of evaluation criteria*. It should also be noted that the user has the option to manually determine the weights of the evaluation criteria according to their preferences and to not use the algorithm presented above. As mentioned, we use the CRITIC method to calculate the weights of the evaluation criteria. The pseudocode for the algorithm to determine the weights of the evaluation criteria is presented below:

```
Function CalculationWeights (decisionMatrix):
    n = number of alternatives (rows in decisionMatrix)
    m = number of criteria (columns in decisionMatrix)
    stdDevs = array of size m initialized to 0
    for j from 1 to m:
        mean = sum of decisionMatrix[i][j] for i from 1 to n
```

```
    variance = sum of (decisionMatrix[i][j] - mean)^2 for i from 1 to n
    stdDevs[j] = sqrt(variance)
correlations = 2D array of size m x m initialized to 0
for j from 1 to m:
    for k from 1 to m:
        if j != k:
            meanJ = sum of decisionMatrix[i][j] for i from 1 to n
            meanK = sum of decisionMatrix[i][k] for i from 1 to n
            covariance = sum of (decisionMatrix[i][j] - meanJ) * (decisionMatrix[i][k] - meanK) for i from 1 to n
            corr = covariance / (stdDevs[j] * stdDevs[k])
            correlations[j][k] = corr
criticWeights = array of size m initialized to 0
for i from 1 to m:
    corrSum = sum of abs(correlations[i][k]) for k from 1 to m, k != i
    criticWeights[i] = stdDevs[i] * (1 - corrSum / (m - 1))
totalWeight = sum of criticWeights[i] for i from 1 to m
normalizedWeights = array of size m
for i from 1 to m:
    normalizedWeights[i] = criticWeights[i] / totalWeight
return normalizedWeights
```

Stage 6. *Ranking of the alternatives*. Finally, the ranking of alternatives should be carried out, which ensures that the optimal route is selected. The pseudocode of the alternative ranking algorithm is given below:

Algorithm RankingAlternatives (decision_matrix, weights)

```
Set number_of_alternatives = number of rows in decision_matrix
Set number_of_criteria = number of columns in decision_matrix
Normalize the decision matrix
    For each criterion j from 1 to number_of_criteria do:
        For each alternative i from 1 to number_of_alternatives do:
        normalized_matrix[i][j] = decision_matrix[i][j] / sqrt(Σ(decision_matrix[k][j]^2))
Determine the weighted normalized matrix
    For each criterion j from 1 to number_of_criteria do:
        For each alternative i from 1 to number_of_alternatives do:
        weighted_normalized_matrix[i][j] = normalized_matrix[i][j] * weights[j]
Identify the ideal (best) and anti-ideal (worst) solutions
    For each criterion j from 1 to number_of_criteria do:
        ideal_solution[j] = max(weighted_normalized_matrix[i][j])  // if the criterion is benefit
        anti_ideal_solution[j] = min(weighted_normalized_matrix[i][j]) // if the criterion is cost
Calculate the distance from the ideal and anti-ideal solutions
    For each alternative i from 1 to number_of_alternatives do:
        distance_to_ideal[i] = sqrt(Σ((weighted_normalized_matrix[i][j] - ideal_solution[j])^2))
        distance_to_anti_ideal[i] = sqrt(Σ((weighted_normalized_matrix[i][j] - anti_ideal_solution[j])^2))
Calculate the relative closeness to the ideal solution
    For each alternative i from 1 to number_of_alternatives do:
        relative_closeness[i] = distance_to_anti_ideal[i] / (distance_to_ideal[i] + distance_to_anti_ideal[i])
Rank the alternatives based on relative closeness
Sort alternatives by relative_closeness in descending order
Return the ranked list of alternatives
```

4. Algorithm Implementation

The algorithm proposed in this paper is implemented as a Web application, which allows us to select the optimal route. The application functions as follows: first, we need to define the locations we wish to visit, after which we can establish the distance matrix. Based on the distance matrix, we can identify all possible routes. Along with determining the routes, the total distances for each corresponding route are calculated. Afterward, we can proceed to define the decision matrix, where the alternatives consist of the routes we have defined. Regarding the evaluation criteria, as we already know, one criterion will be the distances we calculated during the stage of finding all possible routes, while additional criteria can be added at the user's discretion.

The calculation of the weights for the evaluation criteria in the application can be performed in two ways: one is to use the weighting determination algorithm proposed in this paper, or the other is for the user to define the weights manually, according to their preferences. After discussing all the aforementioned stages, we can finally rank the alternatives and select the optimal route.

To demonstrate the operation of the application, let's consider a specific example. Suppose a tourist intends to visit four locations (Location 1, Location 2, Location 3, Location 4). They begin their visit from Location 1 and must return to this location at the end. Figure 1 represents the corresponding distance matrix for these locations.

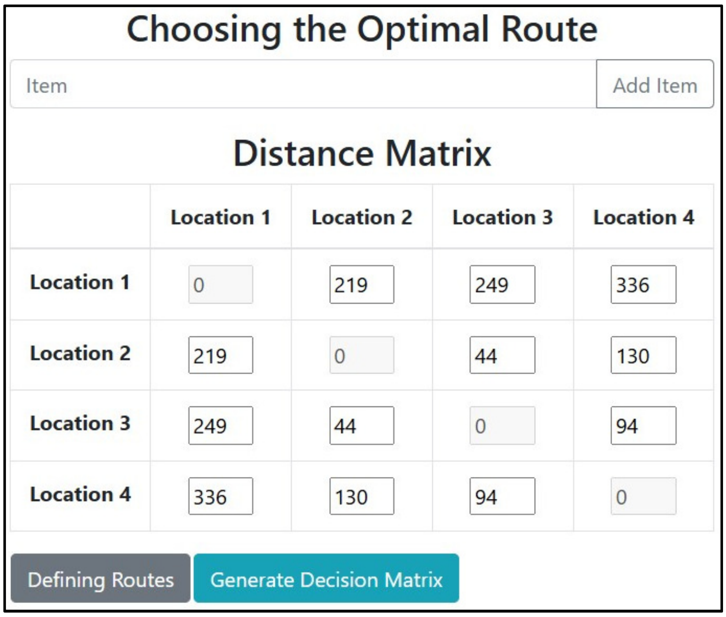


Fig. 1. Distance matrix

The distances matrix describes the distances between the corresponding locations. Our next task is determining all possible routes for visiting these locations and their corresponding total distances. To do this, we need to click the "Defining Routes" button. After this, we can generate the decision matrix by clicking the "Generate Decision Matrix" button, which will provide us with the decision matrix shown in Figure 2.

Decision Matrix:

Enter criterion name | Benefit☐ | Add Criterion

Route	Total Distance (Cost)
Calculate Weight **Weights**	0
Location 1 -> Location 2 -> Location 3 -> Location 4 -> Location 1	693
Location 1 -> Location 2 -> Location 4 -> Location 3 -> Location 1	692
Location 1 -> Location 3 -> Location 2 -> Location 4 -> Location 1	759
Location 1 -> Location 3 -> Location 4 -> Location 2 -> Location 1	692
Location 1 -> Location 4 -> Location 3 -> Location 2 -> Location 1	693
Location 1 -> Location 4 -> Location 2 -> Location 3 -> Location 1	759
Choosing the Best Option	

Fig. 2. The decision matrix including the alternatives (routes) and one criterion

In the decision matrix, we will add three assessment criteria: time, safety, and difficulty. As a result, we will obtain a decision matrix with six alternatives (routes) and four criteria for evaluation. The decision matrix is shown in Figure 3.

Decision Matrix:

Enter criterion name | Benefit☐ | Add Criterion

Route	Total Distance (Cost)	Time(min.) (Cost)	Safety(1-100) (Benefit)	Difficulty of the road(1-100) (Cost)
Calculate Weight **Weights**	0	0	0	0
Location 1 -> Location 2 -> Location 3 -> Location 4 -> Location 1	693	0	0	0
Location 1 -> Location 2 -> Location 4 -> Location 3 -> Location 1	692	0	0	0
Location 1 -> Location 3 -> Location 2 -> Location 4 -> Location 1	759	0	0	0
Location 1 -> Location 3 -> Location 4 -> Location 2 -> Location 1	692	0	0	0
Location 1 -> Location 4 -> Location 3 -> Location 2 -> Location 1	693	0	0	0
Location 1 -> Location 4 -> Location 2 -> Location 3 -> Location 1	759	0	0	0
Choosing the Best Option				

Fig. 3. Decision matrix

After defining the decision matrix, we will fill in the data for the matrix, which is presented in Figure 4.

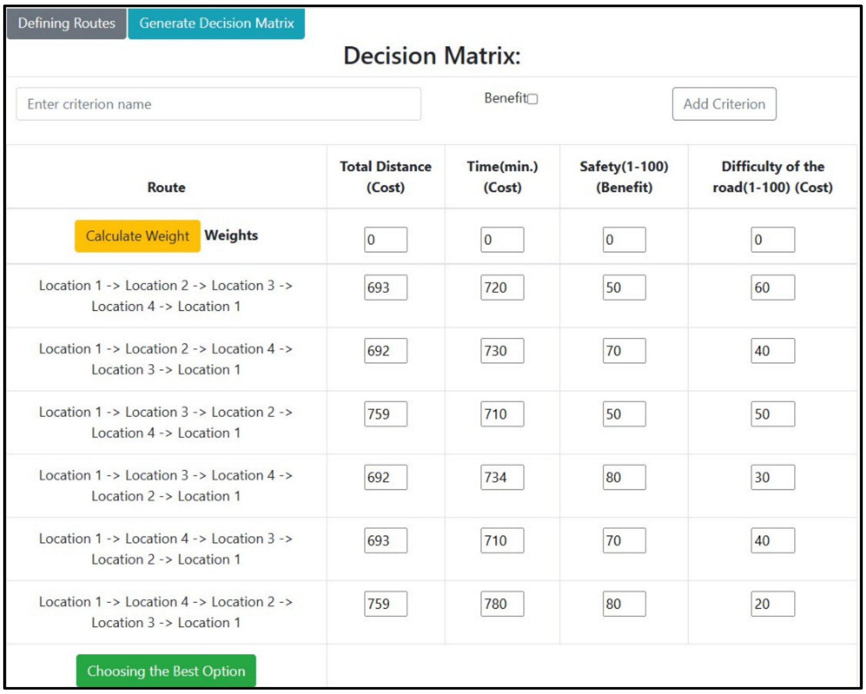


Fig. 4. Data-filled decision matrix

After filling in the data for the decision matrix, it is necessary to determine the weights of the evaluation criteria. As we know, we have proposed two approaches: one is the automatic calculation of weights using the corresponding algorithm (which is provided in the paper), and the other is the manual determination of weights. Let us use the first approach for weight determination; for this, we will click the "Calculate Weight" button, which will result in the calculation of the weights for the evaluation criteria. The decision matrix presented with the weights of the evaluation criteria is given in Figure 5.

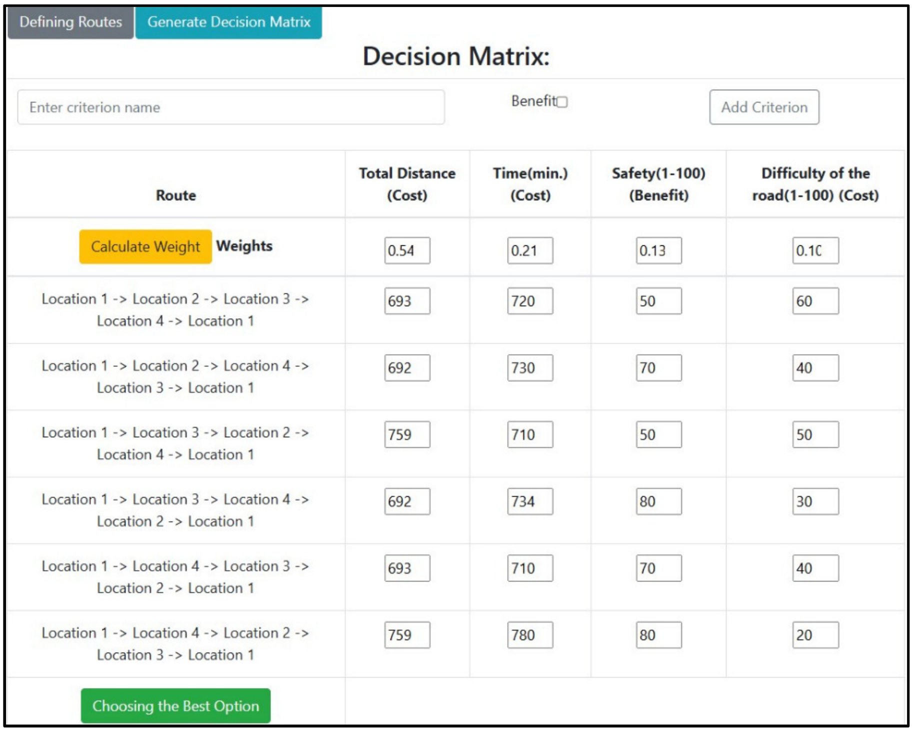


Fig. 5. Decision matrix with evaluation criteria weights

Now we can proceed to select the optimal route; for this, we will click the "Choosing the Best Option" button. As a result, a ranked list of alternatives will be displayed, which is presented below in Figure 6.

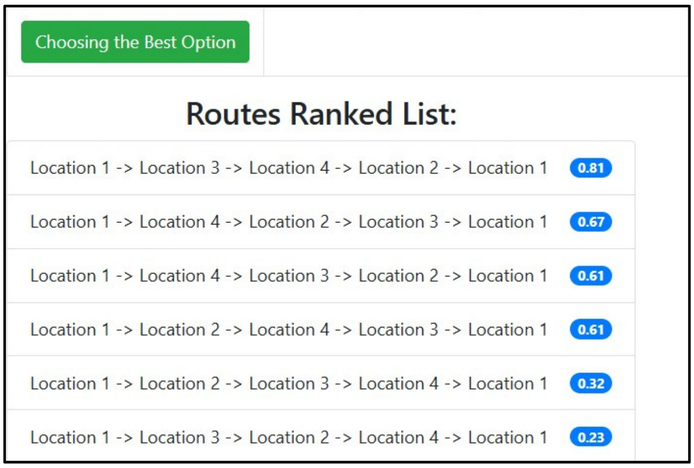
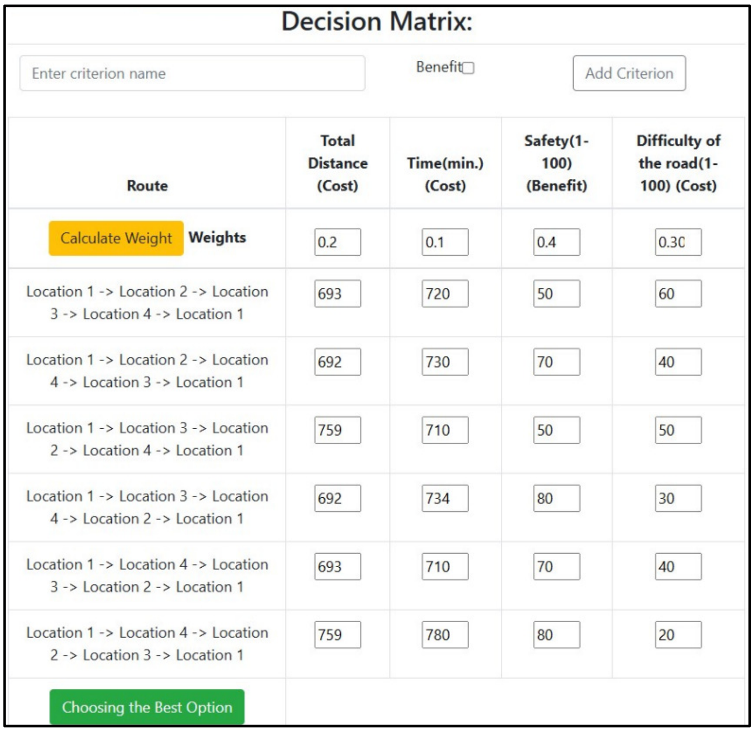


Fig. 6. Ranked list of the routes

The ranked list of the routes shown in Figure 6 indicates that the optimal route is "Location 1 → Location 3 → Location 4 → Location 2 → Location 1".

Let us consider the case where the weights of the evaluation criteria are determined manually. Figure 7 shows the decision matrix along with the manually defined weights of the evaluation criteria.



Route	Total Distance (Cost)	Time(min.) (Cost)	Safety(1-100) (Benefit)	Difficulty of the road(1-100) (Cost)
Calculate Weight **Weights**	0.2	0.1	0.4	0.30
Location 1 -> Location 2 -> Location 3 -> Location 4 -> Location 1	693	720	50	60
Location 1 -> Location 2 -> Location 4 -> Location 3 -> Location 1	692	730	70	40
Location 1 -> Location 3 -> Location 2 -> Location 4 -> Location 1	759	710	50	50
Location 1 -> Location 3 -> Location 4 -> Location 2 -> Location 1	692	734	80	30
Location 1 -> Location 4 -> Location 3 -> Location 2 -> Location 1	693	710	70	40
Location 1 -> Location 4 -> Location 2 -> Location 3 -> Location 1	759	780	80	20

Fig. 7. Decision matrix with manually defined weights of the evaluation criteria

This approach for defining the weights of the evaluation criteria is effective when the user wants to assign greater importance to a specific evaluation criterion. For example, in our case, the most prioritized aspect for the user is the safety of the route.

After manually defining the weights of the evaluation criteria, let us select the best route by clicking the "Choosing the Best Option" button. As a result, a ranked list of alternatives will be displayed (Figure 8).

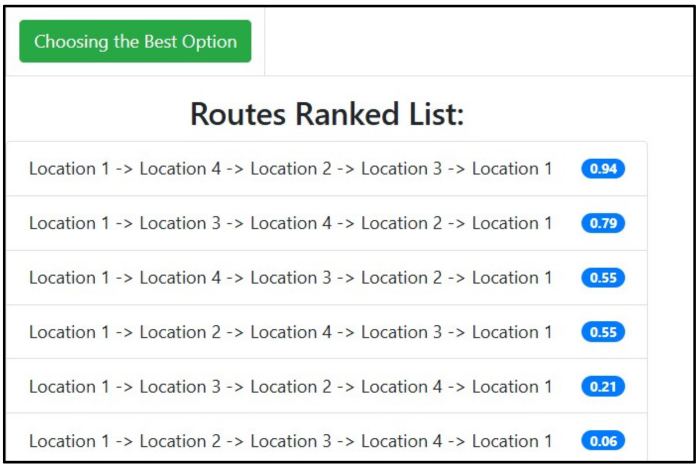


Fig. 8. Ranked list of the routes

As shown in Figure 8, the optimal route is "Location 1 -> Location 4 -> Location 2 -> Location 3 -> Location 1". This means the optimal route has changed compared to the previous case, even though the decision matrix remains the same. This is natural since we changed the weights of the evaluation criteria. This reflects the flexibility of the proposed algorithm, which allows the user to give high priority to the criteria they prefer when selecting the optimal route.

5. Conclusions

The paper presented an algorithm for selecting the optimal route. It was based on route permutation and multi-criteria analysis. It ensured efficient and quick decisions for optimal route selection. One of the advantages of the algorithm is that it takes multiple criteria into account for selecting the optimal route, allowing users to choose an optimal route tailored to their specific needs.

Particular importance was given to applying the algorithm's software presented in the paper in areas of human activity where route planning was relevant. It should also be noted that the proposed software tool requires further development to become more user-friendly. For this, a map should be integrated into its interface, enabling users to mark specific locations and automatically determine the distance matrix using map services. Also, simplifying the input of data into the decision matrix will greatly simplify the software's usability. We are actively working to improve the proposed software solution to turn it into an effective software product.

Conflicts of Interest

The authors declare no potential conflicts of interest in relation to this study.

References

[1] Bęczkowska, S. (2019). The method of optimal route selection in road transport of dangerous goods. *Transportation Research Procedia, 40*, 1252-1259. https://doi.org/10.1016/j.trpro.2019.07.174.

[2] Fan, W. (2004). *Optimal transit route network design problem: Algorithms, implementations, and numerical results*. The University of Texas at Austin, TX, USA.

[3] Samà, M., Pellegrini, P., D'Ariano, A., Rodriguez, J., & Pacciarelli, D. (2017). On the tactical and operational train routing selection problem. *Transportation Research Part C: Emerging Technologies, 76*, 1-15. https://doi.org/10.1016/j.trc.2016.12.010.

[4] Jaillet, P., Qi, J., & Sim, M. (2016). Routing optimization under uncertainty. *Operations Research*, *64*(1), 186-200. https://doi.org/10.1287/opre.2015.1462.

[5] Wei, M., & Meng, Y. (2014). Research on the optimal route choice based on improved Dijkstra. In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications* (WARTIA) (pp. 303-306). IEEE. https://doi.org/10.1016/j.trc.2016.12.010.

[6] Korkmaz, T., & Krunz, M. (2001). Multi-constrained optimal path selection. In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. *Twentieth Annual Joint Conference of the IEEE Computer and Communications Society* (pp. 834-843). IEEE. https://doi.org/10.1109/INFCOM.2001.916274.

[7] Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. *Handbooks in Operations Research and Management Science, 7*, 225-330. https://doi.org/10.1016/S0927-0507(05)80121-5.

[8] Reinelt, G. (2003). *The traveling salesman: computational solutions for TSP applications* (Vol. 840). Springer.

[9] Hoppen, C., Kohayakawa, Y., Moreira, C.G., Ráth, B., & Sampaio, R.M. (2013). Limits of permutation sequences. *Journal of Combinatorial Theory, Series B, 103*(1), 93-113. https://doi.org/10.1016/j.jctb.2012.09.003.

[10] Bóna, M. (2008). Combinatorics of permutations. *ACM SIGACT News, 39*(4), 21-25. https://doi.org/10.1145/1466390.1466394.

[11] Bhadra, D., Dhar, N.R., & Salam, M. A. (2022). Sensitivity analysis of the integrated AHP-TOPSIS and CRITIC-TOPSIS method for selection of the natural fiber. *Materials Today: Proceedings, 56*, 2618-2629. https://doi.org/10.1016/j.matpr.2021.09.178.

[12] Odu, G.O. (2019). Weighting methods for multi-criteria decision making technique. *Journal of Applied Sciences and Environmental Management, 23*(8), 1449-1457. https://doi.org/10.4314/jasem.v23i8.7.

[13] Žižović, M., Miljković, B., & Marinković, D. (2020). Objective methods for determining criteria weight coefficients: A modification of the CRITIC method. *Decision Making: Applications in Management and Engineering*, *3*(2), 149-161. https://doi.org/10.31181/dmame2003149z.

[14] Papathanasiou, J., Ploskas, N., Papathanasiou, J., & Ploskas, N. (2018). *TOPSIS* (pp. 1-30). Springer International Publishing.

[15] Basheleishvili, I., & Bardavelidze, A. (2019). Designing the decision-making support system for the assessment and selection of the university's academic staff. *International Journal on Information Technologies & Security, 11*(2).

[16] Basheleishvili, I., & Bardavelidze, A. (2021). Design and Development of a Group Decision Support System for the Evaluation and Selection of Research Projects. *Engineering, Technology & Applied Science Research, 11*(6), 7881-7886. https://doi.org/10.48084/etasr.4557.

[17] Basheleishvili, I. (2020) Developing the expert decision-making algorithm using the methods of multi-criteria analysis. *Cybernetics and Information Technologies, 20*(2), 22-29. https://doi.org/10.2478/cait-2020-0013.

[18] Basheleishvili, I. (2018). Development of Method of Multifunctional Personnel Assessment Using a TOPSIS Method. *Journal of Technical Science and Technologies, 7*(1), 31-36. https://doi.org/10.31578/jtst.v7i1.137.

[19] Basheleishvili, I., & Kapanadze, G. (2025). Development of A Decision Support System Algorithm for Human Resource Evaluation. *Decision Making Advances, 3*(1), 31-39. https://doi.org/10.31181/dma31202542.

[20] Basheleishvili, I., Tsiramua, S., & Bardavelidze, A. (2022). Algorithmization and realization of the software tool for the software code quality assessment. *International Journal on Information Technologies & Security, 14*(2).