Received: 19 July 2025 | Revised: 18 September 2025 | Accepted: 22 October 2025 | Published online: 26 December 2025

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2025, Vol. 00(00) 1–5
DOI: 10.47852/bonviewJCCE52026863

BON VIEW PUBLISHING

Killer Whale Algorithm for Solving the Maximum Flow Problem on Transportation Networks

Afaf Edinat[1], Mohammad Shehab[1,*], Fatima Haimour[2], Hanaa Fathi[1], and Katrina Sundus[3]

[1] *College of Information Technology, Amman Arab University, Jordan*

[2] *College of Information Technology, Zarqa University, Jordan*

[3] *Faculty of Science and Information Technology, Al-Zaytooneh University of Jordan, Jordan*

Abstract: The most extreme stream issue (MFP) may be a principal subject in the chart hypothesis with applications in computer science, operations inquiry, and designing. It plays a key part in ranges such as activity control, communication frameworks, and the dissemination of assets. Classical calculations, especially Ford–Fulkerson and its expansions, have given the establishment of the means for tackling this issue. In any case, when systems develop bigger, more complex, or dynamically alter, these approaches regularly battle to preserve effectiveness and accuracy. Much like transportation frameworks, each course in an arrangement encompasses a capacity restraint, confining how much stream can move between hubs. Conventional strategies, which accept settled structures and unsurprising imperatives, are not continuously suited to taking care of these challenges. To deal with these limits, this think piece shows the Killer Whale Optimization (KWO) calculation; it is pushed by the chasing methods of killer whales. The strategy was based on multi-flow problems using real and made-up datasets. To show its useful meaning, a case study was made on a project stream in Kota Kinabalu, where made-up systems of different sizes were used to test flexibility and reliability. The result seems like KWO usually beats the Ford–Fulkerson method, especially because it does faster merging and shows more flexibility when dealing with complicated stream situations. These advantages make KWO a good choice for real world stream optimization problems. Beyond theoretical interest, the results suggest that the calculation may be used for project management, transport planning, and resource assignment in various organized systems.

Keywords: traffic congestion, transportation problem, max flow problem, meta-heuristic algorithms, killer whale algorithm

1. Introduction

The Maximum Flow Problem (MFP) is like a classic problem in graph theory that deals with the moving of resources through a network. It tries to find the most amount of flow that can go from a given source node to one or more sink nodes [1]. Such an organization is, as a rule, modeled as a directed chart, where vertices speak to focus within the framework and edges signify the conceivable courses between them. Each edge is relegated to capacity, which indicates the greatest amount of stream that it can oblige. The central point of MFP is to maximize the general exchange from source to sink while guaranteeing that no edge surpasses its doled-out capacity [2]. To address this, an assortment of calculations has been proposed over a long time, each advertising diverse techniques and efficiencies for understanding the issue.

The most widely used algorithms are the Ford–Fulkerson method [3] and Dinic's algorithm [4], both of which provide static solutions to the MFP. In addition to these classical approaches, a difference of heuristic and metaheuristic optimization techniques have been used to solve MFP more efficiently, including the Grey Wolf Optimization (GWO) algorithm [5], Whale Optimization Algorithm (WOA) [6], and other similar methods.

Transportation problems and traffic congestion have turned out to be major worldwide urban road transportation issues. These problems occur when real traffic volumes exceed existing road capacities. This issue generally affects many aspects of human life, such as economic productivity, the natural setting and human welfare. Many reasons may lead to traffic congestion, including slow improvements in traffic facilities, unrestricted car ownership rules, and abnormal driver behavior on the roads [7].

The transportation problem is a type of combinatorial optimization challenge that finds widespread application in various real-world scenarios [8]. One of the most important features of the transportation problem from a network perspective is the maximum capacity of the link [9]. Many applications and industrial fields require the transportation problem to be solved to achieve maximum effective profits and benefits for their business or to find the fastest solution in case of disasters and catastrophes [10]. Traffic congestion arises due to the unmanaged interaction between supply and demand. The ability of a roadway to handle vehicles—known as its intake capacity—requires significant time and financial investment to develop, establishing a fixed upper limit on traffic flow for extended durations. Meanwhile, demand varies across different time intervals, and unlike physical goods, transport services cannot be stockpiled to align supply with changing demand levels [11].

Numerous approaches and solutions have been introduced in academic research to address the transportation problem. Additionally, from a network-based viewpoint, the Maximum Flow (MaxFlow) problem shares structural similarities with the transportation problem.

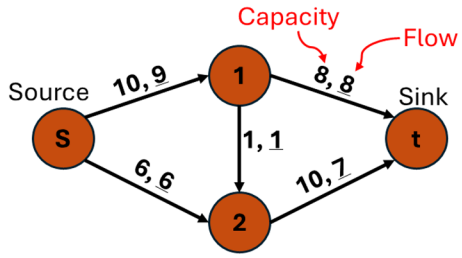
This study presents a novel approach for addressing the MFP utilizing a metaheuristic method known as the Killer Whale

***Corresponding author:** Mohammad Shehab, College of Information Technology, Amman Arab University, Jordan. Email: m,shehab@aau.edu.jo

Optimization Algorithm (KWOA) [12]. The MFP is a fundamental challenge commonly encountered in the analysis of weighted, directed graphs [13]. Its practical significance spans across various domains, including computer science, engineering, operations research, and real-world systems such as municipal water pipelines, electrical circuits, and transportation networks [14].

Numerous strategies, methods, and algorithms have been proposed over the years to tackle the MFP [15]. The essence of the problem lies in determining the highest possible flow from a single origin node to a single destination node within a flow network, ensuring the most efficient utilization of a weighted directed graph, as illustrated in Figure 1. Each edge in the network carries a weight that defines its maximum flow capacity.

Figure 1
MPF network with flow and capacity labels



To find a workable solution, we need to meet two key conditions: first, the flow on each edge should stay within its given capacity limit, and second, at every intermediate node in the network, incoming and outgoing flows must be in perfect equilibrium. Once we meet these criteria, our focus changes to maximizing the overall flow from the source node, which is the starting point, to the sink node, our destination.

KWOA is a fascinating example of methods of metaheuristic optimization. First introduced in 2017 by Biyanto et al. [13], it takes inspiration from the collaborative hunting behaviors of killer whales. Killer whale societies often revolve around a matriline, where an experienced female takes the lead. She is responsible for finding prey and deciding on the most effective route for the hunt. The rest of the group responds to her direction, coordinating their movements to pursue the target. Their effectiveness depends on keeping their actions straightforward while maintaining high speed, which highlights how much their success depends on cooperation.

The MFP could be a central subject in organized optimization, with coordinated suggestions for the proficiency and steadiness of numerous real-world frameworks. It plays a basic part in ranges such as urban transportation arranging, coordination and supply chain administration, activity control, and the dissemination of assets in large-scale frameworks. At its center, the issue includes deciding the most prominent sum of stream that can be exchanged from a source hub to a sink hub inside an organization. This handle must regard the capacity limits relegated to each edge while, moreover, maintaining flow preservation at intermediate nodes, where the overall influx rises to the full surge. Over a long time, classical approaches, just like the Ford–Fulkerson strategy and its refinement, the Edmonds–Karp calculation, have been broadly connected to address such challenges. However, as advanced systems proceed to grow in estimate and complexity, conventional strategies confront developing impediments. These challenges have incited expanding consideration toward the improvement of adaptable and computationally productive procedures that can create near-optimal arrangements, particularly when managing huge or time-sensitive systems. Although traditional algorithms like Ford–Fulkerson and its improved version, Edmonds–Karp, have served us well for a long time, managing real-world transportation systems remains tough. These systems are becoming larger, more complex, and continually evolving. This is why researchers are exploring more adaptable and quicker methods to find nearly perfect solutions, especially when faced with large datasets or tight deadlines [16].

Metaheuristics such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) face challenges when it comes to solving graph-based flow problems. They struggle with finding the right balance between exploring new possibilities and exploiting known solutions, often converge too quickly, and depend heavily on fine-tuning parameters. Most biology-inspired techniques were not designed to exploit the unique directional and topographical properties of natural networks. This gap sheds light on the importance of developing new inferential strategies that had better align with the structural and dynamic characteristics of transportation systems [17].

According to the challenges of transportation networks, this paper presents innovative approach: the KWOA, specifically designed to address the maximum flow problem. Inspired by the remarkable social behaviors of killer whales (Orcinus orca), this method draws from cooperative hunting techniques, echolocation for positioning, and teamwork within groups to devise effective optimization strategies. The KWOA algorithm treats the search for additional paths in the network as a collaborative and adaptive effort. By simulating natural behaviors, such as prey trapping, coordinating group tactics, and moving in wave-like patterns, KWOA achieves a better balance between exploring the bigger picture and refining details, enabling it to navigate through complex network structures more efficiently.

This research striped by the following research questions:

1) How the strategies of collective hunting and the social interactions of killer whales are interpreted into mathematical principles successfully, guide the search for maximum flow in capacity-constrained networks?
2) What length does the proposed KWOA contest or exceed the performance of conventional demand methods and accepted inference techniques, in terms of solution accuracy and convergence efficiency?
3) How well does KWOA adapt when utilized to large-scale, highly linked, and dynamically evolving transportation networks?

This study makes two significant contributions. First, it introduces the KWO algorithm in the field of network flow optimization, marking the first application of this methodology to the maximum flow problem. Second, it inspires an innovative path-oriented coding system that integrates the dynamics of the remaining network, allowing the algorithm to intelligently recognize enhanced paths and optimize flow distribution.

The organization of this paper is as follows: after the introductory section, Section 2 presents a review of related work in the field. Section 3 presents the foundational concepts of the MFP, while Section 4 provides a detailed discussion of the KWOA. In Section 5, the Ford–Fulkerson method is explained. Section 6 examines a practical case study in transportation, and Section 7 presents the experimental results obtained from applying the KWOA to the MFP. Lastly, the study deduces a resume of the key findings and provide motion for potential directions in future research.

2. Literature Review

El-Omari [18] launched a novel bio-inspired metaheuristic, the Sea Lion Optimization (SLnO) algorithm that mark maximum flow problems (MFPs). Comparative experiments applied real-world datasets showed that SLnO executes remarkably well, surpassing traditional approaches such as the Ford–Fulkerson and Whale Optimization Algorithms, especially in large-scale network scenarios.

Akram et al. [19] proposed the Pythagorean Fuzzy Maximum Flow Algorithm (PFMFA), which incorporates linguistic capacities and flows into its optimization framework. PFMFA is particularly valuable when exact numerical data cannot be provided or is too limited to guide the process. The outcomes highlighted how well the algorithm performs and showed its practical importance in tackling optimization problems that involve elements of uncertainty and imprecision.

Akter et al. [20] highlighted the classic Edmonds–Karp algorithm with the goal of addressing some of its performance limitations. The authors introduced refinements designed to improve both the speed and adaptability of the method when applied to different maximum flow problems. Instead of changing the main framework of Edmonds–Karp, they focused on changing some specific computing steps, letting the algorithm fit better to different network situations and work more efficient.

Mohamed [21] showed a wide review of crossover methods that mix machine learning with metaheuristic methods to solve Vehicle Routing Problems (VRPs). The study tried newer improvements in the field, looked at the pros and cons of current methods, and pointed out possible directions for future work to make productivity, flexibility, and practical use in transportation systems better.

The Maximum Value Dynamic Network Flow Problem (MVDFP) was shown by Nixon et al. [22] as a system to maximize the value of flows in dynamic networks. In this study, the value of a node counted only if a minimum flow limit was kept over time. To solve this problem, the authors used a mixed-integer programming (MIP) model and made heuristic algorithms. Their tests showed the proposed method worked well in handling the complexity of the problem.

Çağlar et al. [23] looked at transportation networks focusing on link prediction and maximum flow. Using real data, they applied the Ford–Fulkerson algorithm to check traffic capacity and the Jaccard index to find alternative routes. Their study found big bottlenecks and suggested practical ways to improve traffic flow. In a related work, Mezili et al. [24] made a hybrid method for the flow shop scheduling problem by combining genetic algorithms with penguin search optimization. The GA part used selection, crossover, and mutation, while PSeOA modeled penguin searching to increase search differences. This combination improved analysis and results, giving better performance than GA, PSeOA, or traditional methods alone.

Viswanathan and Ravichandran [25] studied energy saving route design by adding a gain factor to their search method. They checked performance using path length, travel time, and energy use, based on real-time images, ground-truth data, and DSMs from ISPRS. Compared to other advanced algorithms, their method showed more robustness and faster convergence. The results showed about 6% shorter path, 11% better efficiency, and 5% less energy use.

Durban et al. [26] looked at how initialization steps affected Maximum Flow algorithms. They made an Optimized Backoff (OBO) plot to improve throughput in UORA setting. Their method allowed decentralized control without prior knowledge of network estimate or device coordination. Simulations showed the method got near-optimal results, showing the value of correct initialization for improving performance.

A new GA-LDA framework tries to optimize word subsets for LDA, so it improves prediction, developed by Ozcalci and Kilic [27]. They tested it using 928 abstracts from a Turkish-language academic journal about accounting and finance, from 2005 to 2020. The genetic algorithm found the best word subsets for LDA, using perplexity scores as a guide. The tests showed that GA-enhanced LDA improved both classification accuracy and topic modeling results.

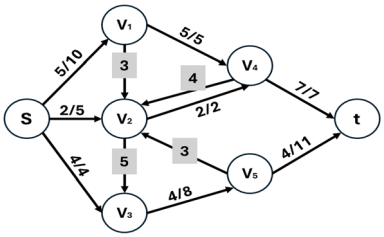
Current metaheuristic methods have many problems when compared to traditional methods. They often stuck too early on one solution (premature convergence), have trouble balancing between searching for new possibilities and improving old ones, and struggle with complex step-by-step problems like MFP. At the same time, many traditional methods also have problems: they do not scale well for bigger problems, cannot search in parallel, and need lots of fine-tuning to work in network settings.

KWOA brings many interesting benefits. It has strong global and local search abilities, reducing the chance of getting stuck in bad solutions. KWOA is a dynamic, efficient, and nature-inspired choice for solving complex optimization problems in transportation networks. Using KWOA is not just adding another algorithm; it tries to find faster, smarter, and more scalable solutions for real-world transportation problems. KWOA's strength is in encouraging collaboration, adaptability, and going beyond local optima. Inspired by the cooperative behavior of killer whales, KWOA gives a practical framework that can be used for everyday problems. Our proposed method not only helps research in optimization but also improves efficiency, flexibility, and resilience of transportation systems, helping build smarter cities.

3. Maximum Flow Problem

A flow network is a type of directed graph in which each edge is assigned a specific capacity. The network includes two main nodes: a source, where the flow originates, and a sink, where the flow is ultimately delivered. Each edge has a non-negative capacity, representing the maximum amount of flow it can carry. The actual flow along any edge must not exceed this capacity. Furthermore, the concept of flow conservation applies to all intermediate (non-terminal) nodes, meaning that the total flow entering a node must be equal to the total flow exiting it. Figure 2 shows sample flow network graph.

Figure 2
Sample flow network graph



A flow network is commonly represented as a directed graph, denoted by $G = (V, E)$, where V is the set of vertices and E comprises, the directed edges connecting them. Within this structure, two specific vertices are identified: the source node $s \in V$, from which flow originates, and the sink node $t \in V$, where flow is received. Each edge, expressed as $e = (u, v) \in E$, is assigned a non-negative capacity $c(u, v)$. The flow throughout the network is defined by a function f that maps an integer value to each edge, ensuring that the flow $f(u, v)$ is always non-negative and does not exceed the corresponding edge's capacity, i.e., $0 \leq f(u, v) \leq c(u, v)$ for all $(u, v) \in E$. Moreover, this flow function f must comply with three essential constraints at each vertex in the graph:

1) **Capacity constraints:** $\forall\, u, v \in V,\ f(u, v) \leq c(u, v)$. The amount of flow allowed on an edge is limited by its capacity.
2) **Skew symmetry:** $\forall\, u, v \in V,\ f(u, v) = -f(v, u)$. The flow from u to v must be the opposite of the net flow from v to u and $f(u, u) = 0$.
3) **Flow conservation:** $\forall\, u \in V - \{s, t\} \sum f(s, v) = 0, v \in V$.

For any intermediate node v—that is, a node that is neither the source (s) nor the sink (t)—the principle of flow conservation applies. This means the total amount of flow entering the node must exactly match the total amount leaving it. In other words, whatever comes in must go out. Furthermore, all flow that starts at the source node (s)

must eventually reach the sink node (t), ensuring overall consistency between input and output. This relationship is formally described in Equation (1):

$$|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t) \tag{1}$$

The MFP is about figuring out the greatest amount of flow that can travel through a network from a specific starting point to an endpoint, all while staying within the limits set by the capacities of the connecting paths.

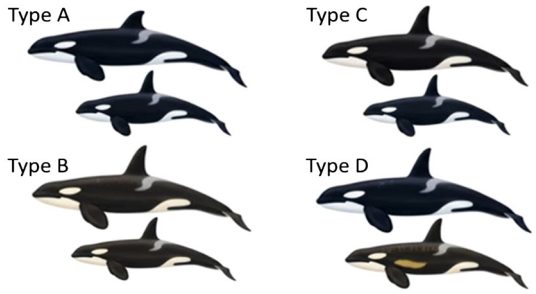
4. Killer Whale Optimization Algorithm

This section provides an overview of the KWOA, which was introduced in 2017. The algorithm takes cues from the social interactions and coordinated hunting methods observed in killer whales. These remarkable creatures live in close-knit family groups called matrilines, led by a dominant individual. During hunts, the leader plays a crucial role in spotting prey and coordinating the most effective attack strategy, while the rest of the group follows and lends support. The algorithm reflects this behavior by incorporating leadership-based coordination and also harnesses the memory capabilities of each matriline, which boosts its overall optimization performance.

4.1. Inspiration

Killer whales (Orcinus orca) are top-level predators in the marine ecosystem. They are categorized into four distinct morphological types—A, B, C, and D—as illustrated in Figure 3. Among these, type A is characterized by the largest body size compared to the other forms of killer whales [28].

Figure 3
Killer whale ecotypes A–D



Killer whales, as dominant predators in the ocean, are generally divided into two specialized groups based on their hunting behavior: (1) Fish-feeding residents, which typically remain within a specific region while hunting, and (2) Mammal-hunting transients, which follow the seasonal migration patterns of their prey. These whales rely on echolocation to locate their targets, using a variety of vocal signals. Their acoustic communication includes three primary sound types: clicks, whistles, and pulsed calls [29].

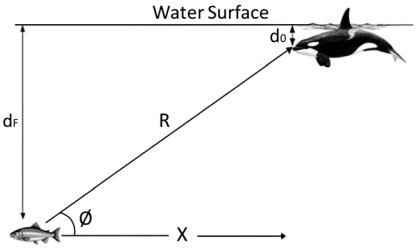
4.2. Implementation of KWOA

This section outlines the mathematical formulation of the KWOA.

4.2.1. Gorging geometry

To implement echolocation in killer whales, we utilize a mathematical model that represents their behavior in searching for prey. This model treats the whales as moving agents, allowing us to identify the optimal solution for the objective function, as illustrated in Figure 4.

Figure 4
Foraging geometry of KWOA at depth d_0 in pursuit of a prey at range R and a depth d_F



The mathematical description of the angle (θ) as the following formula (2):

$$\theta = \sin^{-1}\left(\frac{d_F - d_o}{R}\right) = \tan^{-1}\left(\frac{d_F - d_o}{X}\right) \tag{2}$$

d_F: represents the depth of the prey.
d_0: represents the killer whale's sonar.
R: represent the slant range between the prey and the killer whale.
X: represents the horizontal range.
θ: represents the angle between the slant and horizontal range.

4.2.2. Velocity of movement

Each search agent requires a defined velocity—both in terms of speed and direction—to transition from its current position to the target (prey) location. Consequently, the movement behavior of a killer whale pursuing its prey can be mathematically represented by the following equation:

$$\begin{cases} \overrightarrow{v_l} \leftarrow \overrightarrow{v_l} + \overrightarrow{U}(0, \emptyset_1) \oplus \left(\overrightarrow{p_l} - \overrightarrow{x_l}\right) + \overrightarrow{U}(0, \emptyset_2) \oplus \left(\overrightarrow{p_g} - \overrightarrow{x_l}\right) \\ \overrightarrow{x_l} \leftarrow \overrightarrow{x_l} + \overrightarrow{v_l}.t \end{cases} \tag{3}$$

Each search agent is represented by three vectors in a D-dimensional space, where D denotes the number of dimensions in the search environment. These vectors define the agent's positions as follows:

$\overrightarrow{x_l}$: represents the current position.
$\overrightarrow{p_l}$: represents the previous best position.
$\overrightarrow{v_l}$: represents velocity position.
t: represents time.

A search agent's position within the search space is represented by a coordinate point. At each iteration of the algorithm, this position is assessed as a potential solution to the problem. If it produces a better outcome compared to the previous position, it is selected as the new reference point for the following step. The best value found so far is noted as the personal best position, referred to as *p*best$_l$. This value is continuously updated by comparing it to new objective function outcomes in subsequent iterations, ensuring that the most optimal solution is retained as the new personal best. Updated positions are generated by adjusting the agent's velocity.

4.2.3. Clustering for search space

The optimization process takes place in a multi-dimensional space, where each dimension is limited between $[-x]$ and $[x]$. The primary objective of optimization methods is to identify the most optimal set of variable values within these limits for each dimension. In the KWOA, multiple search agents operate collaboratively in groups. The entire search space is divided into clusters, with each cluster corresponding to a specific agent. Each agent begins its search from the central point of its assigned cluster and focuses on exploring that region to identify the best possible local solution.

The clustering procedure in this algorithm begins with a matrix containing M data points and K initial cluster centers, each described by N dimensions. The number of points assigned to cluster L is denoted as NC(L), and the Euclidean distance between point I and cluster L is represented as $D(I, L)$. The primary objective is to identify a partition into K clusters that minimizes the sum of squared distances within each cluster by optimally reallocating points across clusters. Below is a summary of the key steps involved in this clustering process:

Step 1: For each point I (where I = 1, 2, ..., M), identify the closest and second-closest cluster centers, denoted as $IC1(I)$ and $IC2(I)$, respectively. Assign point I to the cluster $IC1(I)$.

Step 2: Update the cluster centers to be the mean of the points assigned to each cluster.

Step 3: Initially, assign all clusters to the live set.

Step 4: Apply the optimal-transfer (OPTRA) stage: For each point I (where I = 1, 2, ..., M), check the most recent update. If cluster L (where L = 1, 2, ..., K) received an update during the previous quick-transfer (QTRAN) stage, it will be included in the live set for this stage. If a cluster has not been updated in the last M OPTRA steps, it will not be part of the live set. If point I is assigned to cluster $L1$, and $L1$ is in the live set, move on to Step 4a. If it is not in the live set, then proceed to Step 4b.

Step 4a: First, calculate the minimum quantity, $R2$ = [NC(L) $D(I,L)2$] / [NC(L) + 1], for all clusters L (where $L \neq L1$, and L = 1,2,…,K). Identify $L2$ as the cluster with the smallest $R2$. If this value is greater than or equal to [NC($L1$) $D(I,L1)2$] / [NC($L1$) − 1], then no reallocation is needed, and $L2$ becomes the new IC2(I). It is important to remember that the value [NC($L1$) * $D(I,L1)2$] / [NC($L1$) − 1] will be saved and remain constant for point I until cluster $L1$ is updated. If the condition is not met, point I will be assigned to cluster $L2$, and $L1$ will be updated to be the new IC2(I). If a restructuring occurs, the centers of the clusters will be recalculated as the averages of the points assigned to them. The two clusters involved in moving point I in this step will now be included in the live set.

Step 4b: This stage resembles Step 4a, but in this case, the minimum $R2$ is computed solely using the clusters present in the live set.

Step 5: If the live set is empty, we will stop here. If it is not, we will move on to Step 6 after going through the data set once.

Step 6: Perform the QTRAN stage:

Evaluate each point (I) (where (I = 1, 2, …, M)) sequentially. Assign ($L1$ = IC1(I)) and ($L2$ = IC2(I)). There is no need to check point (I) if both clusters ($L1$) and ($L2$) exhibit identical values in the last (M) iterations. Compute the values ($R1$ = [NC($L1$) * $D(I, L1)^2$] / [NC($L1$) −1]) and ($R2$ = [NC($L2$) * $D(I, L2)^2$] / [NC($L2$) + 1]). (As previously mentioned, ($R1$) retains values and will continue to do so until cluster ($L1$) is updated.) If ($R1$) is less than ($R2$), then point (I) is assigned to cluster ($L1$). Otherwise, update (IC1(I)) and (IC2(I)) and refresh the centers of clusters ($L1$) and ($L2$). The two cluster areas are acknowledged for their role in relocation at this stage.

Step 7: If there has not been a transfer in the last M steps, return to Step 4. If there has been a transfer, proceed to Step 6.

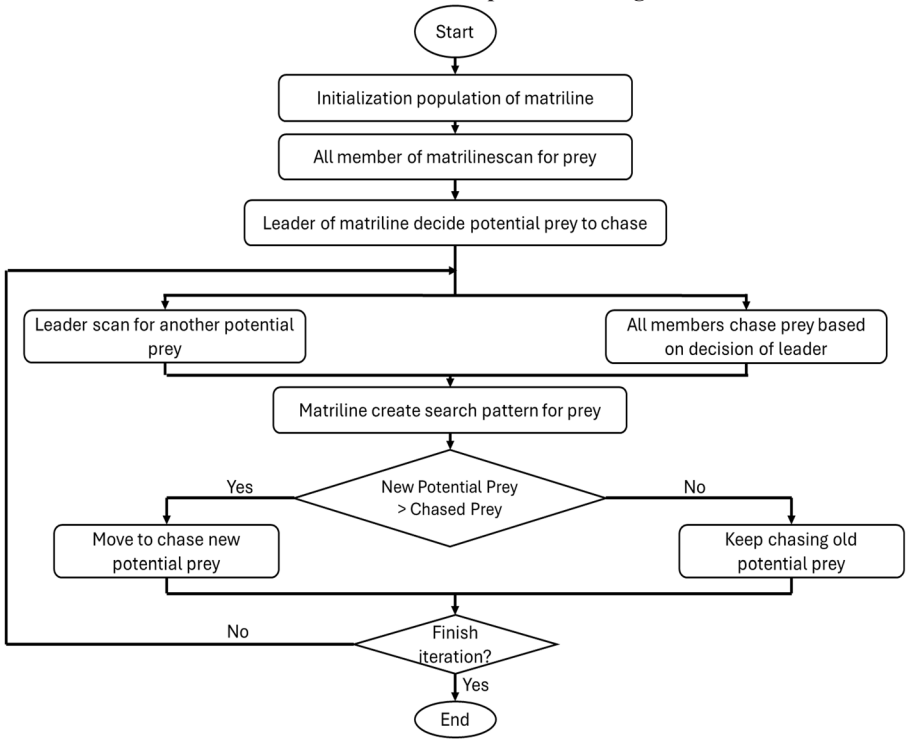
4.2.4. The flowchart of KWOA

During initialization, key parameters are established, including the count of matrilines, the dimension of the objective function, the nature of the global optimum (whether it is a minimum or maximum), and the variable boundaries for optimization. Additionally, the number of clusters required and the total iterations for the clustering procedure are specified. Each matriline consists of a leader along with its associated members. We use a clustering method to efficiently find the global optimum of the objective function while reducing the chances of getting stuck in a local optimum, as outlined in the overall process shown in Figure 5. Each cluster's search area is determined by the spatial distribution of its members, enabling focused exploration around promising regions while preventing excessive overlap between clusters.

5. Ford–Fulkerson Method

The Ford–Fulkerson (FF) method is a systematic approach that revolves around three key ideas: residual networks, augmenting paths, and network cuts. It all starts with a directed graph (G) and an initial

Figure 5
Flowchart of Killer Whale Optimization Algorithm



flow (f), where we kick things off by setting all flow values to zero. This means that for every pair of nodes (u, v) in our graph, the flow ($f(u, v)$) begins at 0.

In each step of the algorithm, we look for an augmenting path—a route from the source to the sink where we can send more flow without exceeding the limits on the edges. Once we find this path, we increase the flow along it. We keep repeating this process until we can no longer find any augmenting paths. In the sections that follow, we will explore these three essential concepts of the Ford–Fulkerson method in more detail, along with a comprehensive explanation and analysis of the algorithm itself.

5.1. Residual network

The residual graph of a network shows the edges that can handle more flow. In a graph $G(V, E)$, starting from the source S and ending at the destination D, let f represent the flow within G. Now, think about two vertices, u and v, in V. The maximum additional flow that can be sent from u to v without going over the capacity $c(u, v)$ is called the residual capacity of the edge (u, v). You can determine this residual capacity using the formula in Equation (4).

$$Cf(u, v) = c(u, v) - f(u, v) \tag{4}$$

For more explanation of Equation (4), we will provide a small example: if we have $c(u,v) = 16$ and $f(u,v) = 11$, the capacity constraint on the edge (u,v) can be increased from $f(u,v) = 0$ in the initial stage to $cf(u,v) = 5$ by calculating Equation (4). Figure 6 shows an example of the network flow; every residual edge admits a flow greater than 0. In panel a, the residual graph flow is shown with a shaded augmenting path. Panel b illustrates the flow of the graph resulting from the augmentation through the path, while panels c and d show the residual network introduced from the flow in panel c.

Table 1 provides a detailed explanation of the calculations performed using Formula 4 on the graph example in Figure 6, offering further insight into how the Ford–Fulkerson method works.

5.2. Augmenting paths

The augmenting path is a straightforward route in the residual network graph, referred to as GF(S, T), that begins at S and concludes at T. In Figure 6, the shaded path represents this augmenting path.

5.3. Flow cuts

The concept of a flow cut is essential to illustrate that the Ford–Fulkerson algorithm continues to iterate until no augmenting paths remain. To demonstrate this, we need to introduce a flow cut in the network graph. The cut (S, T) in the network graph flow $G = (V, E)$ represents a division of the directed graph, where $T = V - S$, with the source node s belonging to set S and the sink node t belonging to set T.

5.4. Ford–Fulkerson algorithm analysis

According to the Ford–Fulkerson algorithm outlined in Table 2, we find several paths through the network and adjust the flow along each edge based on how much capacity is left. This process helps us determine the maximum flow for the graph—represented as $G = (V, E)$—by updating the flow for each pair of connected vertices. If two vertices are not connected, we simply assume there is no flow between them and that their capacity is zero. We calculate the residual capacity using the specified formula. Analyzing the Ford–Fulkerson algorithm shows that

Figure 6

The flow network G and flow

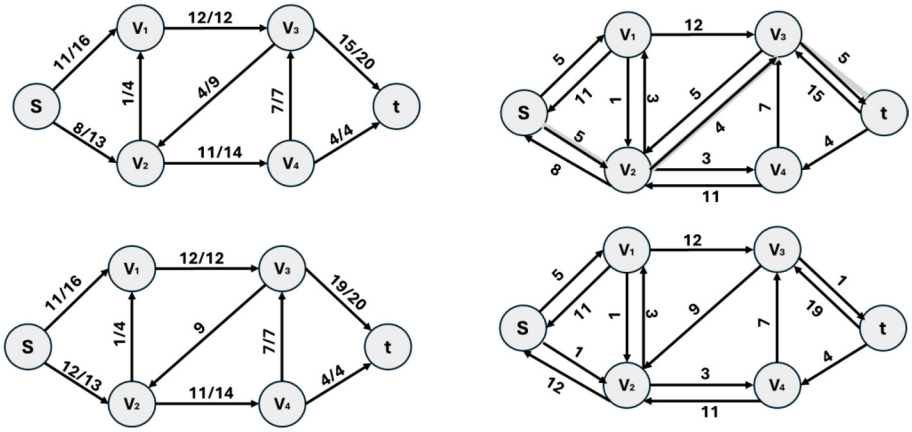


Table 1

Capacity flow calculation

From	To	C	Flow network C	$F(u,v)$	Flow F	$C_f(u,v)$
S	V1	C(S,V1)	16	F(S,V1)	11	5
S	V2	C(S,V2)	13	F(S,V2)	8	5
V1	V3	C(V1,V3)	12	F(V1,V3)	12	0
V2	V1	C(V2,V1)	4	F(V2,V1)	1	3
V2	V4	C(V2,V4)	14	F(V2,V4)	11	3
V3	V2	C(V3,V4)	9	F(V3,V2)	4	5
V3	T	C(V3,T)	20	F(V3,T)	15	5
V4	T	C(V4,T)	4	F(V4,T)	4	0

it runs in polynomial time, specifically $O(E\,|F^{\wedge}|)$, where E is the number of edges and $F^{\wedge}$ is the maximum flow found by the algorithm.

6. Real-World Example of Maximum Flow on Transportation Problem

The proposed method uses the KWOA to address the MFP in real-world transportation situations [30]. As shown in Table 3, vehicles are assigned Passenger Car Unit (PCU) values according to the Urban Traffic System. They are divided into four main groups. The first group, which has cars, taxis, and light goods vehicles (LGVs), has a PCU value of 1.00. The second group includes motorcycles and scooters, which are smaller and given a PCU value of 0.75. The third group covers medium and heavy goods vehicles (MGVs and HGVs), both assigned a PCU value of 2.00. The fourth group consists of buses and large trucks, which have the largest impact on traffic and are assigned a PCU value of 3.00.

Table 4 lists the various location numbers along with their names. Each location has a camera that was placed to count the transportation flow on that edge. Each edge connects two nodes together: from and to nodes, as in the directed network graph, measured by the distance between them in kilometers. Moreover, the maximum flow value is counted as the depicted vehicles at that location in a 5-minute time interval and calculated according to the PCU value shown in Table 4. The final total capacity of the PCU value is calculated according to Equation (5).

$$\text{Total Capacity PCU} = \text{Max flow in 5 min PCU}*12 \tag{5}$$

Equation (5) calculates the total capacity measured for urban roads and the total capacity in PCU to achieve the capacity in 1 h. The samples of the maximum flow were taken and counted over 5 min; by multiplying this figure by the factor of 12, we obtain the total flow capacity for 1 h, as mentioned in the work by Viswanathan and Ravichandran [31].

Figure 7 shows the distance in kilometers between the studied location nodes as a directed network graph of the transportation

Table 2
Ford–Fulkerson algorithm

Ford–Fulkerson algorithm	Complexity analysis
For each edge (u,v) ∈E[G] **do f[u,v] ← 0** **f[v,u] ← 0**	**θ (E)**
while there exists a path P from S to T in the residual network Gf **do Cf(P)←min{cf(u,v):(u,v) is in P}** **for each edge (u,v) in P** **do F[u,v]← F[u,v]+Cf(P)** **F[u,v] ← - F[u,v]**	**θ (|F*|)** **theta**

Table 3
PCU value for urban road

Vehicle class	PCU value
Car, LGV	1.00
Scooter, Motorcycle	0.75
MGV & HGV	2.00
Bus & Truck	3.00

Figure 7
Weighted directed graph according to the distance between nodes

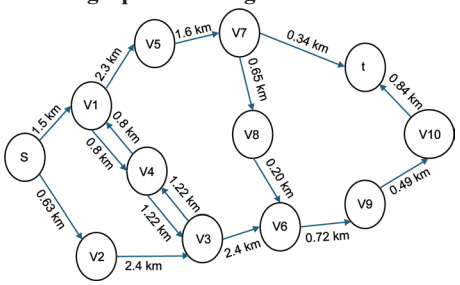


Table 4
A real dataset of traffic flow of Kota Kinabalu as an example of the maximum flow problem

Loc. no.	Location name	From	To	Distance in km	Max flow in 5 min in PCU	Total capacity
1	Jalan Tun Fuad Stephen 1	S	V1	1.5	226.25	2715
2	Jalan Pasir	S	V2	0.63	175.5	2106
3	Jalan Tun Fuad Stephen 2	V1	V5	2.3	220	2640
4	Jalan Tuaran 1	V2	V3	2.4	242.75	2913
5	Jalan Kompleks Sukan& Jalan Bunga Nasar 1	V3	V4	1.22	130.75	1569
6	Jalan Kompleks Sukan& Jalan Bunga Nasar 2	V4	V3	1.22	70.25	843
7	Jalan Istiadat 1	V4	V1	0.8	82.25	987
8	Jalan Istiadat 2	V1	V4	0.8	74	888
9	Jalan Tuaran 2	V3	V6	2.4	156.25	1875
10	Jalan K.K Bypass	V5	V7	1.6	118.5	1422
11	Jalan Tuaran 3	V6	V9	0.72	231.75	2781
12	Jalan Tunku Abdul Rahman 1	V7	V8	0.65	87	1044
13	Jalan Tunku Abdul Rahman 2	V8	V9	0.2	95	1140
14	Jalan Kemajuan	V9	V10	0.49	211	2532
15	Jalan Laiman Diki	V7	T	0.34	120.75	1449
16	Jalan Coastal	V10	T	0.84	95.75	1149

problem we have. The nodes are represented as circles, and the edges are arcs connecting these nodes. Each arc has a value that indicates the distance in kilometers between the two nodes. Moreover, every arc has a direction, which specifies our directed graph. Figure 7 shows the maximum flow capacity of the studied scenario, which is the total capacity PCU in Table 1. Based on these values, we construct our maximum flow graph on which the experiments will be conducted.

Figure 8 shows the maximum flow capacity of the studied scenario, which is the total capacity PCU in Table 1. Based on these values, we construct our maximum flow graph on which the experiments will be conducted.

Figure 8
Weighted directed graph according to the total capacity PCU

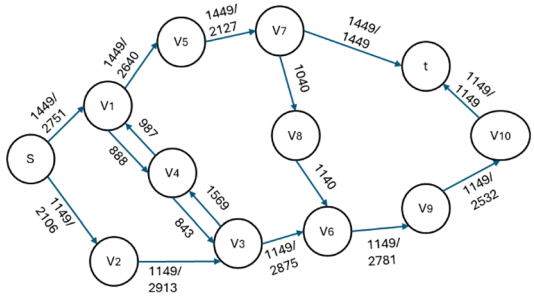
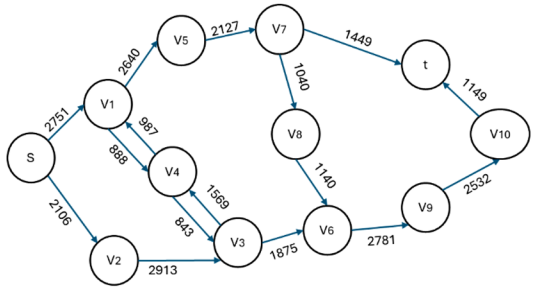


Figure 9
Ford–Fulkerson algorithm results



6.1. Ford–Fulkerson algorithm and KWOA

Figure 9 presents the outcomes of our experiments using the Ford–Fulkerson algorithm. The first route, S → V1 → V5 → V7 → T, has a capacity of up to 1449 vehicles per hour. Meanwhile, the second route, S → V2 → V3 → V6 → V9 → T, can handle a maximum of 1149 vehicles per hour. Overall, the Ford–Fulkerson algorithm is able to achieve a total flow of 2598 vehicles per hour.

Table 5 presents a comparison of the run times between KWOA and Ford–Fulkerson when applied to the Kota Kinabalu datasets. As the table shows, KWOA performs more efficiently than Ford–Fulkerson on this data.

7. Experimental Results and Discussion

In this section, we examined both the Ford–Fulkerson algorithm and the KWOA to solve the MFP using a real-world dataset, alongside a variety of experimental results from networks of different sizes. to setup a benchmark and assess the effectiveness of the proposed metaheuristic approach in tackling the MFP, KWOA is compared with the Ford–Fulkerson algorithm. The Ford–Fulkerson algorithm is a well-established, traditional method that guarantees optimal solutions by systematically recognized paths to increase flow, making it a foundational reference in network flow theory.

By evaluating KWOA against this classic algorithm, the solution quality is analyzed, computational efficiency, and scalability, especially in large and complex networks where conventional methods can struggle. This comparison confirms the importance of balancing optimality with execution speed, aspect that KWOA can achieve near-optimal flow solutions in substantially shorter runtimes, particularly as network sizes grow. Our findings designate that bio-inspired metaheuristics as KWOA provide a practical and scalable alternative for real-world applications, such as managing

Table 5
KWOA run time contrast to Ford–Fulkerson

Location no.	From	To	Distance in km	Total capacity in PCU	(Ford–Fulkerson)	KWOA
1	$	V1	1.5	2715	226.25	222.02
2	$	V2	0.63	2106	175.50	171.27
3	V1	V5	2.3	2640	220.00	215.77
4	V2	V3	2.4	2913	242.75	238.52
5	V3	V4	1.22	1569	130.75	126.52
6	V4	V3	1.22	843	70.25	66.02
7	V4	V1	0.8	987	82.25	78.02
8	V1	V4	0.8	888	74.00	69.77
9	V3	V6	2.4	1875	156.25	152.02
10	V5	V7	1.6	1422	118.50	114.27
11	V6	V9	0.72	2781	231.75	227.52
12	V7	V8	0.65	1044	87.00	82.77
13	V8	V9	0.2	1140	95.00	90.77
14	V9	V10	0.49	2532	211.00	206.77
15	V7	T	0.34	1449	120.75	116.52
16	V10	T	0.84	1149	95.75	91.52

urban traffic, where quick decision-making is crucial. Thus, this comparison not only underscores the effectiveness of KWOA but also situates it within the broader context of network optimization, linking classical algorithms with modern, adaptable computational techniques. A simulation program using MATLAB was developed to evaluate how well FF and KWOA perform. All the experiments were set up and tested with MATLAB version 7.12.0 (R2011a) on a 64-bit system.

7.1. Generated dataset

The dataset sizes in this study varied from 50 to 1000, following the approach described in earlier research by Mirjalili and Lewis [6]. For each scenario, we conducted 10 experimental trials to calculate the average execution time. Table 6 shows the average execution time in seconds for the KWOA compared to the Ford–Fulkerson algorithm across different dataset sizes. Interestingly, the KWOA performed significantly faster than the Ford–Fulkerson method, processing a dataset with 500 nodes in about 6.4 times less time.

Table 6
Run time of KWOA compared to Ford–Fulkerson

Network size	Average run times/seconds (Ford–Fulkerson)	Average run times/seconds (Killer Whale)
50	0.0672	0.0466
100	0.1563	0.0597
150	0.2156	0.0644
200	0.3125	0.0706
250	0.3641	0.0753
300	0.4328	0.0784
350	0.4938	0.0815
400	0.5750	0.0842
450	0.7000	0.0858
500	0.7500	0.0936
550	0.8469	0.1092
600	0.9422	0.1373
650	1.0906	0.1483
700	1.2109	0.1591
750	1.3891	0.1982
800	1.5656	0.2170
850	1.7250	0.2279
900	2.0266	0.2549
950	2.1672	0.2862
1000	2.4010	0.3230

The Ford–Fulkerson algorithm was chosen in this study to serve as a benchmark for evaluating the effectiveness of the KWOA, as illustrated in Figures 10 and 11. As previously noted, each dataset underwent 11 experimental runs to ensure reliable average results. Based on the outcomes of both algorithms, Figure 12 clearly indicates that the KWOA outperformed Ford–Fulkerson, particularly in larger network

Figure 10
The experimental results for Ford–Fulkerson

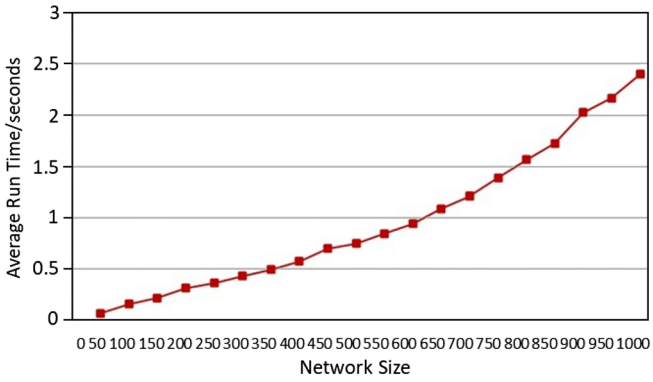


Figure 11
The experimental results for KWOA

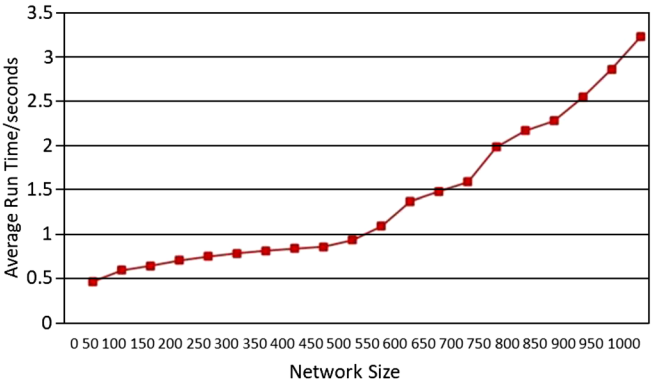
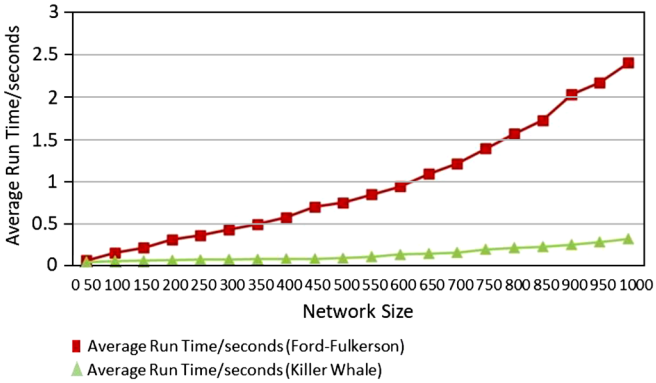


Figure 12
The experimental results for Ford–Fulkerson versus KWOA



scenarios. This figure highlights the performance improvements achieved in this work when compared to the Ford–Fulkerson approach. Table 7 shows the theoretical and experimental runtime results obtained across several benchmark algorithms, namely Ford–Fulkerson (FF), Grey Wolf Optimizer (GWO), Killer Whale Optimization Algorithm (KWOA), Pythagorean Genetic Algorithm (PGA), Sea Lion Optimization (SLnO), and Whale Optimization Algorithm (WOA). It presents both execution time metrics (mean, standard deviation, median, minimum, and maximum runtime) and convergence-related

Table 7
Theoretical and experimental run time

Method	Max flow	Mean runtime	Std runtime	Median runtime	Min runtime	Max runtime	Mean iterations	Std iterations	Median iterations	Min iterations	Max iterations
FF	1520	23.9202	1.3236	23.6926	22.2389	26.1491	1387.9	71.32	1407.5	1262	1487
GWO	1520	16.623	0.6612	16.5142	15.6384	17.6189	1057.4	43.06	1067.5	994	1135
KWOA	1520	12.9077	0.9446	12.9154	11.5546	14.6028	880.6	49.12	875	794	942
PGA	1520	18.4389	1.1298	18.4744	16.1762	20.2183	1121.6	60.58	1116	1008	1226
SLnO	1520	14.4197	0.8337	14.5616	13.3384	15.8902	968.3	54.82	965	892	1086
WOA	1520	15.7894	0.5878	15.9728	14.9398	16.4166	989.3	47.99	990	927	1045

statistics (mean standard deviation, median, minimum, and maximum number of iterations).

The results illustrate while the classical Ford–Fulkerson algorithm consistently reach the correct maximum flow value, it reveals the highest mean runtime (23.92 s) and required the largest number of iterations on average (1387.9), desired limited scalability. In contrast, KWOA perform the most favorable balance between runtime and iterations, with an average runtime of only 12.91 s and an average of 880.6 iterations, significantly outperforming both FF and other metaheuristics such as GWO (16.62 s, 1057.4 iterations) and WOA (15.79 s, 989.3 iterations). Also, the small standard deviation figures for KWOA show that the algorithm consistently converges across different attempts. These results emphasize how KWOA effectively reduces computational costs and accelerates convergence, making it a valuable approach for addressing maximum flow challenges in large transportation systems.

8. Conclusions

This study investigates the KWOA, which uses a population-based search to explore multiple paths at the same time. KWO method give elastic alternative to traditional exact algorithms. Applying KWOA to synthetic networks of varying sizes (from 50 to 1000 nodes) and a real-world urban traffic dataset from Kota Kinabalu, it demonstrated higher computational efficiency than the Ford–Fulkerson method in larger networks while maintaining strong accuracy in flow estimation. KWOA spotlights the potential of bio-inspired optimization to upgrade adaptive, intelligent, and sustainable solutions for modern transportation systems.

KWOA also has limitations. Like other metaheuristic methods, it does not always guarantee an optimal solution and requires careful parameter tuning to achieve the best results. Its population-based search can initiate additional computational overhead, which may slow performance in smaller networks. Furthermore, the current implementation assumes static edge capacities and does not account for dynamic or time-dependent flows, shorten its applicability in rapidly changing traffic conditions.

Future work should focus on adapting KWOA for dynamic networks, varying capacities, and multi-commodity flow scenarios to improve its versatility. Combining KWOA with exact algorithms could strike a balance between efficiency and guaranteed optimality. Additionally, integrating real-time IoT traffic data would facilitate live optimization in urban environments.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data are available from the corresponding author upon reasonable request.

Author Contribution Statement

Afaf Edinat: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Mohammad Shehab:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Fatima Haimour:** Methodology, Validation, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Supervision. **Hanaa Fathi:** Conceptualization, Software, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Katrina Sundus:** Software, Validation, Writing – original draft, Writing – review & editing.

References

[1] Al-Qudah, Y., Jaradat, A., Sharma, S. K., & Bhat, V. K. (2024). Mathematical analysis of the structure of one-heptagonal carbon nanocone in terms of its basis and dimension. *Physica Scripta*, *99*(5), 055252. https://doi.org/10.1088/1402-4896/ad3add

[2] Alipour, H., Muñoz, M. A., & Smith-Miles, K. (2023). Enhanced instance space analysis for the maximum flow problem. *European Journal of Operational Research*, *304*(2), 411–428. https://doi.org/10.1016/j.ejor.2022.04.012

[3] Ekanayake, E. M. U. S. B., Daundasekara, W. B., & Perera, S. P. C. (2022). New approach to obtain the maximum flow in a network and optimal solution for the transportation problems. *Modern Applied Science*, *16*(1), 30. https://doi.org/10.5539/mas.v16n1p30

[4] Sriwahyuni, L., Marwan, M., & Awanis, Z. Y. (2023). Optimization of water flow on regency municipality waterworks-network of Jonggat Central Lombok regency using Ford Fulkerson algorithm and Dinic algorithm. *Eigen Mathematics Journal*, *6*(1), 49–54. https://doi.org/10.29303/emj.v6i1.157

[5] Shehab, M., Sihwail, R., Daoud, M., Al-Mimi, H., & Abualigah, L. (2024). Nature-inspired metaheuristic algorithms: A

comprehensive review. *The International Arab Journal of Information Technology*, *21*(5), 815–831. https://doi.org/10.34028/iajit/21/5/4

[6] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

[7] Opoku, O. A., Yeboah, O., Gyamfi, E., & Afful, G. (2022). Rising car ownership and traffic congestion in the university of cape coast campus. *International Journal of Research in Science & Engineering*, *24*, 10–21. https://doi.org/10.55529/ijrise.24.10.21

[8] Shaikh, F. A., Memon, N. A., & Chandio, I. A. (2025). Exploring factors causing traffic congestion through average index method: A case study of Hyderabad city. *Pakistan Journal of Engineering, Technology and Science*, *13*(1), 51–60.

[9] Shahin, R., Hosteins, P., Pellegrini, P., Vandanjon, P.-O., & Quadrifoglio, L. (2024). A survey of flex-route transit problem and its link with vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, *158*, 104437. https://doi.org/10.1016/j.trc.2023.104437

[10] Yang, S., Zhang, J., & Zhou, S. (2024). The cost transportation game for collaboration among transportation companies. *Annals of Operations Research*, *336*(3), 1479–1503. https://doi.org/10.1007/s10479-023-05466-4

[11] Liu, H., Wu, Z., & Yu, L. (2024). Optimization of emergency stockpile siting: A review of models, influencing factors, and future research directions. *Journal of Engineering Management and Systems Engineering*, *3*(4), 226–235. https://doi.org/10.56578/jemse030404

[12] Abualigah, L., Al-Okbi, N. K., Elaziz, M. A., & Houssein, E. H. (2022). Boosting marine predators algorithm by Salp Swarm Algorithm for multilevel thresholding image segmentation. *Multimedia Tools and Applications*, *81*(12), 16707–16742. https://doi.org/10.1007/s11042-022-12001-3

[13] Biyanto, T. R., Matradji, N., Irawan, S., Febrianto, H. Y., Afdanny, N., RaBiyanto, T. R., ..., & Bethiana, T. N. (2017). Killer whale algorithm: An algorithm inspired by the life of killer whale. *Procedia Computer Science*, *124*, 151–157. https://doi.org/10.1016/j.procs.2017.12.141

[14] Daulat, S., Rokstad, M. M., Klein-Paste, A., Langeveld, J., & Tscheikner-Gratl, F. (2024). Challenges of integrated multi-infrastructure asset management: A review of pavement, sewer, and water distribution networks. *Structure and Infrastructure Engineering*, *20*(4), 546–565. https://doi.org/10.1080/15732479.2022.2119480

[15] Srivastava, V., & Biswas, B. (2022). LM-MFP: Large-scale morphology and multi-criteria-based feature pooling for image parsing. *Soft Computing*, *26*(13), 6201–6218. https://doi.org/10.1007/s00500-022-07138-0

[16] Saini, A., & Rahi, O. P. (2024). Optimal power flow approaches for a hybrid system using metaheuristic techniques: A comprehensive review. *International Journal of Ambient Energy*, *45*(1), 2345839. https://doi.org/10.1080/01430750.2024.2345839

[17] Meng, Z., Yıldız, B. S., Li, G., Zhong, C., Mirjalili, S., & Yildiz, A. R. (2023). Application of state-of-the-art multiobjective metaheuristic algorithms in reliability-based design optimization: A comparative study. *Structural and Multidisciplinary Optimization*, *66*(8), 191. https://doi.org/10.1007/s00158-023-03639-0

[18] El-Omari, N. K. T. (2020). Sea lion optimization algorithm for solving the maximum flow problem. *International Journal of Computer Science and Network Security*, *20*(8), 30–68. https://doi.org/10.22937/IJCSNS.2020.20.08.5

[19] Akram, M., Habib, A., & Allahviranloo, T. (2022). A new maximal flow algorithm for solving optimization problems with linguistic capacities and flows. *Information Sciences*, *612*, 201–230. https://doi.org/10.1016/j.ins.2022.08.068

[20] Akter, D., Uddin, M. S., & Shami, F. A. (2021). Modification of EDMONDS-KARP algorithm for solving maximum flow problem. *International Journal of Innovation and Applied Studies*, *31*(4), 703–711.

[21] Mohamed, A. W. (2024). A review of hybrid machine learning and metaheuristics for vehicle routing problems. *Metaheuristic Optimization Review*, *2*(2), 48–58. https://doi.org/10.54216/MOR.020205

[22] Nixon, T., Curry, R. M., & Allaissem B., P. (2025). Mixed-integer programming models and heuristic algorithms for the maximum value dynamic network flow scheduling problem. *Computers & Operations Research*, *175*, 106897. https://doi.org/10.1016/j.cor.2024.106897

[23] Çağlar, A., Öztemiz, F., & Yakut, S. (2024). Link prediction and maximum flow in transportation network. *Computer Science*, *9*(2), 169–177. https://doi.org/10.53070/bbd.1593501

[24] Mzili, T., Mzili, I., Riffi, M. E., Pamucar, D., Simic, V., Abualigah, L., & Almohsen, B. (2024). Hybrid genetic and penguin search optimization algorithm (GA-PSeOA) for efficient flow shop scheduling solutions. *Facta Universitatis, Series: Mechanical Engineering*, *22*(1), 77–100. https://doi.org/10.22190/FUME230615028M

[25] Viswanathan, S., & Ravichandran, K. S. (2025). Gain-based green ant colony optimization for 3D path planning on remote sensing images. *Spectrum of Operational Research*, *2*(1), 92–113. https://doi.org/10.31181/sor21202510

[26] Durban, J., Fearnbach, H., Paredes, A., Hickmott, L., & LeRoi, D. (2021). Size and body condition of sympatric killer whale ecotypes around the Antarctic Peninsula. *Marine Ecology Progress Series*, *677*, 209–217. https://doi.org/10.3354/meps13866

[27] Ozcalci, M., & Kilic, M. (2025). GA-LDA approach for topic modeling in Turkish accounting and finance articles: Performance optimization in text classification. *Spectrum of Operational Research*, *2*(1), 305–322. https://doi.org/10.31181/sor21202521

[28] Branch, T. A. (2025). Most "flight" baleen whale species are acoustically cryptic to killer whales, unlike "fight" species. *Marine Mammal Science*, *41*(3), e13228. https://doi.org/10.1111/mms.13228

[29] Giovannini, G., Miller, P. J. O., Wensveen, P. J., & Samarra, F. I. P. (2025). Sound production during feeding in Icelandic herring-eating killer whales (Orcinus orca). *Ethology Ecology & Evolution*, *37*(3), 298–317. https://doi.org/10.1080/03949370.2024.2437373

[30] Abdullah, N., & Hua, T. K. (2017). The application of the shortest path and maximum flow with bottleneck in traffic flow of Kota Kinabalu. *Journal of Computer Science & Computational Mathematics*, *7*(2), 37–43. https://doi.org/10.20967/jcscm.2017.02.003

[31] Viswanathan, S., & Ravichandran, K. S. (2025). Gain-based green ant colony optimization for 3D path planning on remote sensing images. *Spectrum of Operational Research*, *2*(1), 92–113. https://doi.org/10.31181/sor21202510

How to Cite: Edinat, A., Shehab, M., Haimour, F., Fathi, H., & Sundus, K. (2025). Killer Whale Algorithm for Solving the Maximum Flow Problem on Transportation Networks. *Journal of Computational and Cognitive Engineering*. https://doi.org//10.47852/bonviewJCCE52026863