I J C R M
International Journal of Contemporary Research In Multidisciplinary

Research Article

Smart Fusion Re-Routing Algorithm: A Hybrid Approach to Congestion-Aware Urban Navigation by Enhancing the Edmonds-Karp Algorithm

Laishram Trinity [1*], Swavana Yaikhom [2]
[1] Student, M Tech (AI), NIELIT IMPHAL, Deemed to Be University, Imphal, India
[2] Asst professor, Dept. of Al, NIELIT, Deemed to be University, Imphal, Manipur, India

Corresponding Author: *Laishram Trinity

DOI: https://doi.org/10.5281/zenodo.18416462

Abstract

Urban road systems often suffer from congestion, which reduces traffic efficiency and increases delays. Traditional flow-based algorithms, such as Edmonds-Karp, respect network capacity but ignore dynamic congestion, whereas shortest path heuristics optimise distance or time but overlook flow constraints. This study proposes a hybrid routing model that combines Edmonds-Karp maximum flow, dynamic congestion modelling, and congestion-aware shortest path heuristics. Edmonds-Karp ensures capacity limits are respected, while congestion updates and weighted path selection enable adaptive rerouting. Vehicle movement was simulated using linear interpolation (LERP) to visualise congestion and routing decisions. Simulations on a sample city network showed that the hybrid approach reduced congestion buildup and improved travel efficiency compared with conventional methods, making it well-suited for autonomous navigation, intelligent transportation, and smart city traffic management.

Manuscript Information

- **ISSN No:** 2583-7397
- **Received:** 11-12-2025
- **Accepted:** 26-01-2026
- **Published:** 29-01-2026
- **IJCRM:**5(1); 2026: 352-357

- **Plagiarism Checked:** Yes
- **Peer Review Process:** Yes

How to Cite this Article

Trinity L, Yaikhom S. Smart Fusion Re-Routing Algorithm: A Hybrid Approach to Congestion-Aware Urban Navigation by Enhancing the Edmonds-Karp Algorithm. Int J Contemp Res Multidiscip. 2026;5(1):352-357.

Access this Article Online

www.multiarticlesjournal.com

KEYWORDS: LERP, BFS, Edmonds-Karp, intelligent transportation system, autonomous navigation, network rerouting.

1. INTRODUCTION

Urban road networks frequently experience significant congestion, which reduces traffic efficiency and increases travel time [3]. Traditional routing methods, such as the Edmonds-Karp algorithm, are based on flow and ensure that paths are viable concerning network capacity, but cannot adapt to fluctuating congestion levels [10]. Conversely, heuristics focus on finding the shortest path to minimise travel time or distance but neglect real-time flow limitations, often resulting in less-than-optimal routes during heavy traffic [9]. This study introduces a hybrid routing model that combines the Edmonds-Karp maximum flow algorithm with dynamic congestion modelling and a congestion-aware shortest path heuristic to enable adaptive and capacity-respecting navigation. The Edmonds-Karp algorithm guarantees that routing choices do not exceed the capacity of any link [2]. Meanwhile, congestion modelling keeps the edge utilisation updated as vehicles move through the networks. A modified shortest-path heuristic that considers both distance and congestion facilitate adaptive rerouting, which strikes a balance between efficiency and fairness. To improve the model's interpretability, linear interpolation (LERP) was used to simulate vehicle movements, offering a visual representation of congestion impacts and the selection of optimal paths of vehicles.

Experimental simulations conducted on a sample city network showed that this hybrid approach effectively mitigated congestion buildup and enhanced average travel efficiency compared to conventional routing strategies. The findings underscore the benefits of integrating flow-based feasibility with congestion-aware optimisation, positioning the model as a suitable solution for autonomous navigation, intelligent transportation systems, and smart city traffic management applications.

2. LITERATURE SURVEY

Traffic routing optimisation shares many similarities with Virtual Machine (VM) placement in cloud computing, as both are NP-hard problems that require the efficient allocation of limited resources [1][6]. Heuristic approaches, such as power-aware best fit decreasing (PABFD), provide fast decisions but often miss global optimality, whereas metaheuristics, such as Genetic Algorithms (GA) and ant colony optimisation (ACO), achieve better efficiency through the adaptive balancing of competing objectives [5]. Flow-based algorithms, particularly Ford-Fulkerson, Edmonds–Karp, and Dinic, have been widely applied to graph-related tasks, such as image segmentation and traffic navigation. Edmonds-Karp improves Ford-Fulkerson by using BFS to always find the shortest augmenting path, whereas Dinic introduces the level graph strategy that restricts augmenting paths to nodes of increasing levels, making it the fastest and most efficient in dense networks such as traffic or image graphs [4]. Parallel to these, metaheuristics such as Cuckoo Search (CS), inspired by cuckoo brood parasitism and Lévy flights, demonstrate strong performance in balancing the exploration and exploitation of large search spaces, similar to the hybridisation of Edmonds-Karp's flow feasibility with adaptive congestion-aware heuristics [6]. Recent advances have further emphasised the computational efficiency of dynamic networks by proposing conditional re-invocation flow algorithms. Instead of recalculating paths for every minor edge weight change, flow recomputation is triggered only when the bottleneck edges shift traffic regimes or when the non-bottleneck edges lose capacity beyond their residual tolerance [8]. This approach drastically reduces the number of flow algorithm invocations from potentially millions to a few hundred, making real-time traffic navigation systems, such as Google Maps, scalable and practical. Collectively, these studies highlight the progression from brute-force recalculations to intelligent, hybridised strategies that integrate flow algorithms with adaptive heuristics, a direction that motivates our proposed Smart Fusion Re-Routing Algorithm (SFRA).

3. METHODOLOGY

3.1 Edmonds-Karp

The Edmonds-Karp algorithm is an implementation of the Ford-Fulkerson method for computing the maximum flow in a flow network [7], and its primary role in our hybrid model is to ensure that routing decisions respect the road capacity constraints.

3.1.2 Edmonds-Karp working concept

Flow network representation: Each road is modelled as a directed edge $e(u, v)$ with capacity $c(u, v)$.

Residual graph: At each iteration, a residual graph G_f is maintained where edge capacity reflects unused capacity, which is given by Equation 1.

$$C_f(u, v) = c(u, v) - f(u, v) \quad (1)$$

Augmenting path search: The algorithm repeatedly finds a shortest augmenting path from the source s to sink t using Breadth-First Search (BFS).

Flow update: Once a path is found, the minimum residual capacity, the bottleneck is determined by using Equation 2.

$$\Delta = \min_{(u,v)\in P} \{C_f(u, v)\} \quad (2)$$

and the flow is updated accordingly through Equation 3.

$$f(u, v) \leftarrow f(u, v) + \Delta, f(u, v) \leftarrow f(u, v) - \Delta \quad (3)$$

This process continues until no augmenting path exists.

3.2 Congestion Modelling (added layer)

Although Edmonds-Karp ensures capacity feasibility, it does not consider traffic dynamics. Thus, a congestion modelling layer was added.

3.2.1 Concept

Each edge $e(u, v)$ is assigned a congestion factor $\gamma(u, v) \in [0,1]$ that evolves with traffic, the factor value increased as the number of vehicles increased.

The congestion model is given by Equation 4.

$$\gamma(u, v) = \frac{vehicles\ on\ (u,v)}{capacity\ of\ (u,v)} \quad (4)$$

$\gamma(u, v) = 0$ indicates that it's the free-flowing edge.

$\gamma(u, v) = 1$ indicates that it's congested, effectively blocked.

This layer provides real-time awareness of traffic conditions and forms a basic layer for rerouting decisions in the hybrid layer.

Consider a scenario where the edge (0,1) has a capacity of 10, but already 9 cars are on it, the congestion factor. $\gamma(0,1)$=0.9
This congestion factor directly influences the cost of using that edge in the shortest-path heuristic, and it's given by Equation 5.

$$\omega(u,v) = \alpha.d(u,v) + \beta.\gamma(u,v) \quad (5)$$

Equation 5 helps avoid the congested direction, thus resulting in the creation of a new route.
$d(u,v)$ is the physical distance, weight of the moving from node u to node v.
Consider the road length as 2 km, $d(u,v) = 2$
$\gamma(u,v)$ is the congestion factor of the edge (u,v) which can be defined by Equation 6.

$$\gamma(u,v) = \frac{vehicles\ currently\ using\ the\ edges}{capacity\ of\ edge} \quad (6)$$

If $road\ empty \leftarrow \gamma = 0$
If $fully\ saturated \leftarrow \gamma = 1$

The weighting coefficients are the α, β. They control how much importance we give to distance vs congestion.

$$High\ \alpha, low\ \beta \rightarrow prioritize\ shortest\ distance$$
$$Low\ \alpha, High\ \beta \rightarrow prioritize\ avoiding\ congestion$$

3.3 Approached Hybrid algorithm

By combining the two terms, the algorithm avoids extreme cases.
Equation 5 was reused here to compute the numericals.

The sensitivity of the algorithm is presented in Table 1, which shows how the choice between Road A (5 km, γ=0.9) and Road B (7 km, γ=0.2) changes when α and β are varied.
Road A: Distance = 5 Km (d=5 km) Congestion = 0.9 (γ=0.9)
Road B: Distance = 7 Km (d=7 km) Congestion = 0.2 (γ=0.2)
Equation 5 is substituted with the values of roads A and B.
Road A cost: $\omega_A = \alpha.\ 5 + \beta.\ 0.9$
Road A cost: $\omega_B = \alpha.\ 7 + \beta.\ 0.2$

Table 1: Sensitivity Analysis Table

α	β	ω_A	ω_B	Decision
1	1	5 + 0.9 = 5.9	7 + 0.2 = 7.2	A
1	5	5 + 4.5 = 9.5	7 + 1.0 = 8.0	B
2	5	10 + 4.5 = 14.5	14 + 1.0 = 15.0	A
3	5	15 + 4.5 = 19.5	21 + 1.0 = 22.0	A
2	10	10 + 9.0 = 19.0	14 + 2.0 = 16.0	B
4	2	20 + 1.8 = 21.8	28 + 0.4 = 28.4	A

Interpretation of Table 1

- If β is small (low importance of congestion), the algorithm selects Road A (shorter distance dominates).
- If β is large (high importance of congestion), the algorithm selects Road B (less crowded dominates).
- When α increases, distance dominates more, maintaining the preference toward Road A.

Fig 1: Decision boundary for the hybrid cost function with the recommended directions.

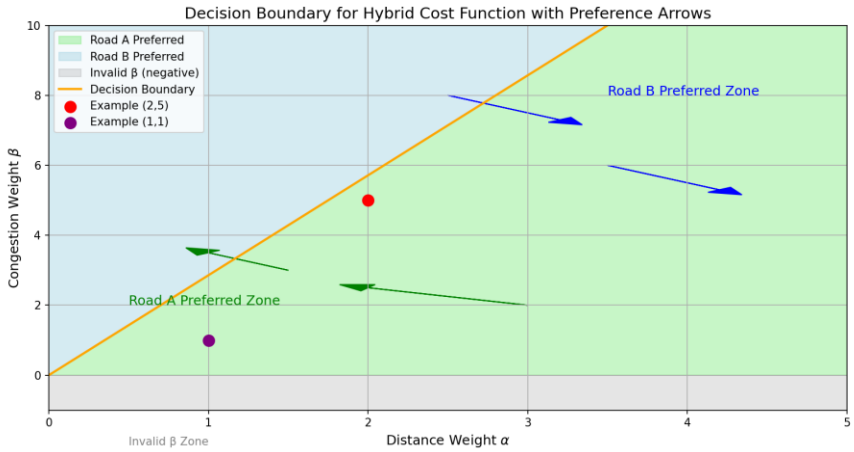


Figure 1 clearly shows that Road A is optimal when distance is weighted more heavily relative to congestion (lower β values), whereas Road B becomes optimal when congestion is weighted more heavily (higher β values). The arrows highlight the sensitivity of the decision to changes in α and β values.

3.3.1 Components of the graph

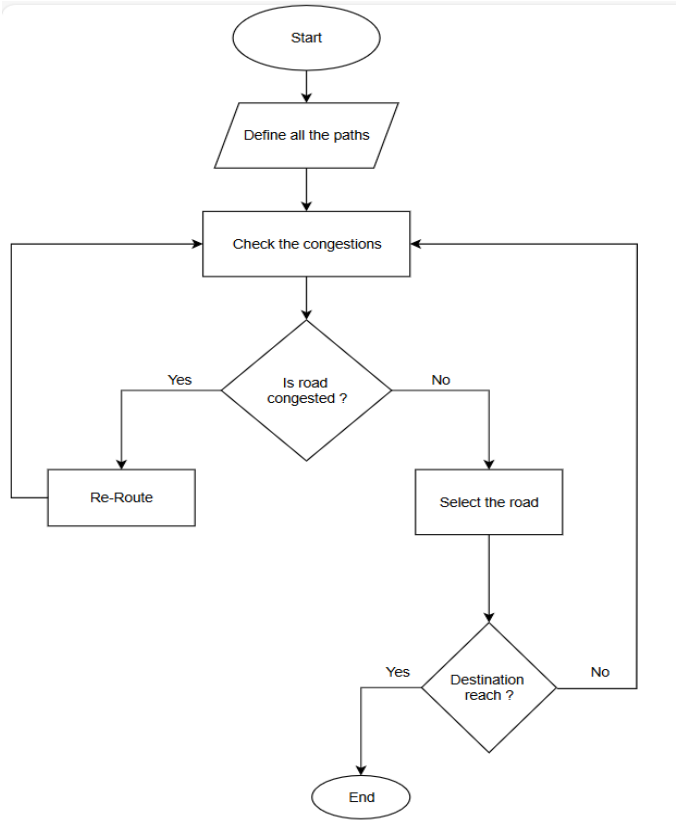
Light Green Region: Road A is preferred. This occurs when the combination of α and β results in a lower hybrid cost for Road A.
Light Blue Region: Road B is preferred, indicating scenarios where congestion has more impact relative to distance, making Road B optimal.
Grey Region: Invalid β values (negative congestion weight), which are not physically meaningful.

3.3.2 Decision Boundary (Orange Line):

- Represents the set of α–β combinations where the hybrid costs for both roads are equal ($C_A = C_B$)
- This boundary separates the zones where each road is preferred.

This visualisation effectively communicates both qualitative and quantitative aspects of the hybrid cost decision, making it clear which road is preferred under different weighting scenarios and how the decision shifts with α–β variations.

Figure 2: Flowchart of Hybrid Routing for Congestion-Aware Navigation



The algorithm will first sense the routes of different routes. If there is congestion in the route, then this route will not be taken and will be further rerouted. And if there is a non-congested road, then the algorithm will select this non-congested road and check whether the destination has reached or not. If the algorithm will stop if not again, it will be rerouted.

Algorithm: Hybrid Routing for Congestion-Aware Navigation
Input: City road network (nodes, edges, capacities), EK path, Hybrid path
Output: Animated car simulation showing congestion and optimal routing

1. Begin
2. The city road network and edge capacities of the road network were loaded.
3. A directed graph representing the road network was constructed.
4. The EK (congested) and hybrid (optimal) paths are defined.
5. Each path was converted into coordinates for plotting and animation.
6. The plot is initialised with nodes, edges, and edge labels (capacities).
7. Multiple cars are initialised for each path.

Red cars for the EK path
Green cars for the Hybrid path

8. Initialise the congestion tracking for the EK path edges.
9. **For each animation frame:**

a. Move EK cars along the path (slower) and increase the congestion on the edges.
b. Move Hybrid cars along the path (faster).
c. Update the EK path edge colours based on the congestion level (darker red for higher congestion).

10. The cars' positions along the paths were displayed in real time.
11. Step 9 was repeated until the animation ended.
12. Optionally, the animation can be saved as a GIF for further analysis.
13. End

Pseudocode

BEGIN
LOAD city road network (nodes, edges, capacities)
BUILD directed graph G
DEFINE EK path (congested) and Hybrid path (optimal)
CONVERT paths to coordinates for plotting
INITIALIZE plot with nodes, edges, and capacities
INITIALIZE multiple cars:
Red cars for the EK path
Green cars for the Hybrid path
INITIALIZE congestion tracker for EK path edges

WHILE animation is not finished:
FOR each car in the EK path:
MOVE car along path (slower)
UPDATE congestion on the current edge
FOR each car in the Hybrid path:
MOVE car along path (faster)
UPDATE EK path edge colours based on congestion
DISPLAY cars on the plot
END WHILE
OPTIONALLY SAVE animation as GIF
END

4. Complexity Analysis

The proposed algorithm constructs a directed graph from the capacity matrix and extracts the paths for congestion-aware navigation. The complexity is analysed as follows:

1. **Graph Construction**

The algorithm iterates over all entries of the capacity matrix of size $n \times n$ times. An edge is inserted into the graph for each non-zero entry. Because each insertion takes a constant time, the total complexity of the graph construction is $O(n^2)$.

2. **Path Extraction:** The path-coordinate function processes at most mmm edges in a given path. As the path length $m \leq n$, the cost is bounded by $O(n)$.
3. **Interpolation Function:** The interpolate-path function computes intermediate positions using simple arithmetic operations, which run in constant time $O(1)$.

Combining these, the overall time complexity of the algorithm is dominated by the graph construction step, yielding a time complexity of $O(n^2)$.

For space complexity, the algorithm requires memory to store the capacity matrix and the graph. $(O(n^2))$, the graph representation $(O(n^2))$, and additional linear storage for paths. Hence, the total space usage is given by $O(n^2)$

Table 2: Comparative Complexity Analysis

Algorithm	Time Complexity	Space Complexity	Key Limitation
Edmonds-Karp (Max Flow)	$O(VE^2)$	$O(V+E)$	High computational cost in dense networks.
Dijkstra's (shortest path)	$O(V+V\log V)$	$O(V+E)$	Ignores congestion
Proposed Hybrid Algorithm	$O(n^2)$	$O(n^2)$	Quadratic scaling; suitable for small- to medium-sized networks, but may require optimisation for very large networks.

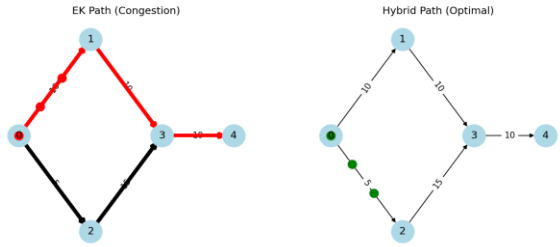
Table 2 analyses the comparison highlights that while classical flow-based algorithms such as Edmonds–Karp guarantee feasible routing under network capacity constraints, they are computationally expensive for large, dense networks. On the other hand, shortest-path heuristics are efficient but fail to adapt to real-time congestion conditions, often producing suboptimal routes under heavy traffic. The proposed hybrid graph construction achieves a balance by operating in quadratic time, which is lightweight for small- to medium-scale networks, while simultaneously incorporating congestion-aware routing. This trade-off ensures practical applicability in urban road systems where responsiveness and adaptability are as important as computational efficiency.

5. Result

The performance of the proposed hybrid routing algorithm was evaluated using a simulated urban road network in Figure 3. Two paths from the source node (0) to the destination node (4) were analysed.

- EK Path (Congested): Edges 0→1→3→4 using the traditional Edmonds-Karp algorithm.
- Hybrid Path (Optimal): Edges 0→2→3→4 using the proposed hybrid approach.

Figure 3: Traffic simulation comparing the EK algorithm and the proposed hybrid approach, showing reduced congestion and improved travel efficiency.



5.1 Observations

- EK Path: Vehicles experienced slower movement because of the congestion. The edge colours dynamically transitioned to dark red as traffic increased, indicating bottlenecks. The overall travel time was longer owing to the cumulative congestion effects.
- Hybrid Path: Vehicles travel faster along uncongested edges. The edge colours remained stable, indicating minimal congestion. The hybrid path significantly reduced travel time compared to the EK path.

5.2 CONCLUSION

The simulation confirmed that the hybrid routing algorithm effectively minimised congestion by dynamically selecting the optimal paths. The visual results highlight that the EK path is susceptible to congestion, whereas the hybrid approach maintains a smooth traffic flow, demonstrating its suitability for congestion-aware urban navigation.

6. Future Scope

The smart fusion rerouting algorithm opens several promising directions for both academic research and practical applications. From a research perspective, future studies should focus on the theoretical analysis of the hybrid cost function to establish optimal guarantees and convergence properties under varying weight configurations. Comparative evaluations with advanced flow algorithms, such as Push-Relabel and Capacity Scaling, could further validate the efficiency of dense and dynamic networks. Incorporating predictive learning techniques, including reinforcement learning and evolutionary optimisation, offers the potential to move beyond reactive congestion management toward proactive, pattern-driven navigation [11]. Additionally, extending the framework to multi-source multi-sink flow problems and stochastic traffic models would enhance its robustness under real-world uncertainties.

On the application side, the algorithm can be integrated with live GPS data, IoT sensors, and edge computing platforms to enable real-time congestion-aware routing in smart cities. Its congestion modelling layer can be adapted to multimodal transportation systems, covering cars, buses, and emergency vehicles, thereby supporting holistic mobility planning. The visualisation module can also be scaled to interactive 3D or AR-based navigation systems, offering greater interpretability

for both end users and urban planners. Finally, integration with autonomous vehicle routing frameworks and intelligent transportation systems would position the model as a practical and scalable solution for next-generation urban-traffic management.

REFERENCES

1. Barlaskar E, Singh YJ, Issac B. Energy-efficient virtual machine placement using enhanced firefly algorithm. *Multiagent Grid Syst*. 2016;12(3):167–198. doi:10.3233/MGS-160250.
2. Cao XB, Du WB, Zhang J, Chen CL. Effect of adaptive delivery capacity on networked traffic dynamics. *Chin Phys Lett*. 2011;28(5):058902. doi:10.1088/0256-307X/28/5/058902.
3. Choudhury S, Solovey K, Kochenderfer MJ, Pavone M. Efficient large-scale multi-drone delivery using transit networks. *J Artif Intell Res*. 2021;70:757–788. doi:10.1613/jair.1.12450.
4. Even S, Tarjan RE. Network flow and testing graph connectivity. *SIAM J Comput*. 1975;4(4):507–518. doi:10.1137/0204043.
5. Gao Y, Guan H, Qi Z, Hou Y, Liu L. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *J Comput Syst Sci*. 2013;79(8):1230–1242. doi:10.1016/j.jcss.2013.02.004.
6. Issac B, Singh YJ, Barlaskar E. Enhanced cuckoo search algorithm for virtual machine placement in cloud data centres. *Int J Grid Util Comput*. 2018;9(1):1. doi:10.1504/IJGUC.2018.10011385.
7. Lammich P, Sefidgar SR. Formalising network flow algorithms: a refinement approach in Isabelle/HOL. *J Autom Reason*. 2019;62(2):261–280. doi:10.1007/s10817-017-9442-4.
8. Ma D, Xiao J, Ma X. A decentralised model predictive traffic signal control method with fixed phase sequence for urban networks. *J Intell Transp Syst*. 2021;25(5):455–468. doi:10.1080/15472450.2020.1734801.
9. Menelaou C, Panayiotou CG, Polycarpou MM, Kolios P, Timotheou S. Minimising traffic congestion through continuous-time route reservations with travel time predictions. *IEEE Trans Intell Veh*. 2019;4(1):141–153. doi:10.1109/TIV.2018.2886684.
10. Shaikh PW, Gao K, El-Abd M, Khanafer M. A review on swarm intelligence and evolutionary algorithms for solving the traffic signal control problem. *IEEE Trans Intell Transp Syst*. 2022;23(1):48–63. doi:10.1109/TITS.2020.3014296.
11. Tshakwanda PM, Arzo ST, Devetsikiotis M. Advancing 6G network performance: AI/ML framework for proactive management and dynamic optimal routing. *IEEE Open J Comput Soc*. 2024;5:303–314. doi:10.1109/OJCS.2024.3398540.

About the corresponding author

Laishram Trinity is an M. Tech (Artificial Intelligence) student at NIELIT Deemed to be University, Imphal, India. His academic interests include artificial intelligence, machine learning, and data-driven systems, with a focus on applying advanced AI techniques to solve real-world computational and societal challenges.

Swavana Yaikhom is an Assistant Professor in the Department of Artificial Intelligence at NIELIT Deemed to be University, Imphal, Manipur, India. Her research interests include artificial intelligence, machine learning, and intelligent computing systems, with a strong emphasis on teaching, research, and mentoring in emerging AI technologies.