Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 10 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i10.52401

PAPER

Path Selection Optimization Algorithms for Mobile Agent Based on Push-All-Data Strategy

Faisal Alzyoud[1], Monther Tarawneh[2], Faiz Al-Shrouf[3], Ahmed Almaghthawi[4](✉), Meaad Alrehaili[5], Hasan Kanaker[6]

[1]Department of Computer Science, Isra University, Amman, Jordan

[2]Department of Information Technology, College of Information Technology and Communications, Tafila Technical University, Tafila, Jordan

[3]Department of Software Engineering, Faculty of Information Technology, Isra University, Amman, Jordan

[4]Department of Computer Science, College of Science & Art at Mahayil, King Khalid University, Abha, Saudi Arabia

[5]Department of Computer Sciences and Artificial Intelligence, Collège of Computer Science, and Engineering, University of Jeddah, Jeddah, Saudi Arabia

[6]Department of Information Security, Faculty of Information Technology, Petra University, Amman, Jordan

aalmaghthwi@kku.edu.sa

ABSTRACT

With the advent of 5G and 6G technologies and the growing ubiquity of the Internet, the Mobile Agent (MA) paradigm is increasingly seen as a promising alternative to the conventional client-server model. MAs, which are software entities capable of moving and processing data across different systems, offer potential efficiencies in data management. However, their operation in dynamic and mobile environments can lead to challenges, such as incomplete or delayed tasks. This study addresses these issues by focusing on reducing the relocation time of MAs. A numerical procedure and a streamlining strategy were developed to expedite the transfer of an agent from the source to the target hub. Utilizing the itinerary design pattern and the Ant Colony Optimization (ACO) algorithm, implemented via the Java Agent Development Framework (JADE), this study sought the most efficient path for the MA. The proposed algorithm demonstrated a significant improvement, selecting the optimal path in just 271.511 seconds. This performance represents a substantial enhancement over previous approaches using the master-slave design pattern with either the Genetic Algorithm (GA) or the Node Compression Algorithm (NCA). The implications of this improvement are far-reaching, potentially enhancing the efficiency and reliability of data management systems in a variety of applications.

KEYWORDS

Mobile Agent (MA), Ant Colony Optimization (ACO), itinerary design pattern, Java Agent Development Framework (JADE), efficiency in data management

1 INTRODUCTION

In today's rapidly expanding digital environment with countless numbers of Internet of Things (IoT) devices and increasing ubiquitous computing, as well as the massive adoption of edge computing, mobile agent (MA) systems are more important than ever. Such systems are of great importance within data distribution, processing, and management settings where the aforementioned conditions prevail. They demonstrate competitive benefits in terms of scalability, flexibility, and the ability to

Alzyoud, F., Tarawneh, M., Al-Shrouf, F., Almaghthawi, A., Alrehaili, M., Kanaker, H. (2025). Path Selection Optimization Algorithms for Mobile Agent Based on Push-All-Data Strategy. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(10), pp. 112–128. https://doi.org/10.3991/ijim.v19i10.52401

Article submitted 2024-11-08. Revision uploaded 2025-02-14. Final acceptance 2025-03-03.

operate with low network bandwidth usage. Given that 5G and even 6G technologies will soon reach the public, the ability of MAs to handle as much work as possible and transmit only the necessary data to reduce latency and network load will also be valuable.

Mobile agent systems, increasingly pivotal in internet applications, enable data interaction across networks by migrating between hosts. These systems, particularly effective in large-scale networks, have revolutionized communication cost management by localizing computation tasks [1]. The surge in mobile device usage has further propelled the advancement of internet and telecommunication technologies, making MAs a key solution for reducing communication costs and bolstering mobile social communications [2, 3].

Functioning independently from mobile devices, MAs can access web portals for data retrieval and transmit results back to the user. Beneficial in scenarios with direct PC-network connections, these agents support adaptable load transmission. Their performance is influenced by factors such as transfer speed and resource availability, rendering them versatile for both wired and wireless networks. The core properties of MAs—portability, communication, learnability, autonomy, and persistence—facilitate their effectiveness [4]. Portability, for instance, enables migration of code and state management across PCs, thus enhancing inter-agent communication.

The agent paradigm illustrates how MAs can alleviate network traffic by staging data transfers from remote hosts instead of direct server retrievals, as shown in Figure 1 [5]. However, in dynamic and mobile environments, challenges such as incomplete or inaccessible work can hinder the performance of these agents. Addressing these challenges is crucial for optimizing MA systems.

In this study, we introduce new optimization techniques for MAs that are optimized for a dynamic and sometimes unreliable networking environment given the current state of mobile and IoT applications. Ant Colony Optimization (ACO) is optimized to work better in an environment where the connectivity is disconnected and the bandwidth frequency is sometimes unreliable, as seen in the currently used wireless networks. Its primary purpose is to reduce the time it takes for the MAs to move, notably to maximize their efficiency in normal, dynamic circumstances. Using a novel numerical technique and a simplification method, the work employs itinerary design pattern in the use of the ACO algorithm via the Java Agent Development Framework (JADE) framework. Thus, our paper aims to optimize the pathfinding of MAs within a network.

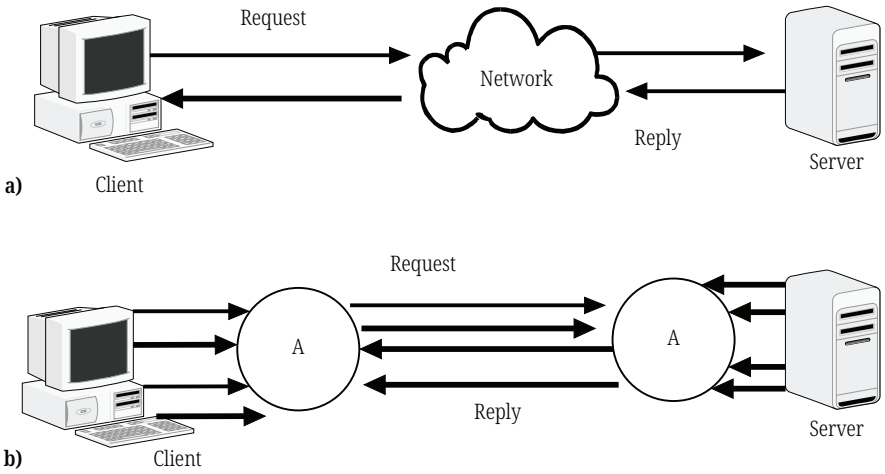


Fig. 1. Mobile Agent Paradigm data transfer scenario [5]

2 RELATED WORK

Mobile agents support adaptable and versatile load transfer moves from host to host, contingent upon transfer speed and other accessible assets. So, the MA innovation is useful for wired and wireless networks. MAs are distinguished from other computing programs by the following properties [6]. MAs can benefit from using the master-slave pattern [34], which can be further optimized through various performance improvement patterns that take advantage of the agent's behavior and characteristics. Mathematical approaches and models can also be used to support better mobility performance by considering the interaction between various factors. To demonstrate this, a group of MAs were tested using the Aglet platform, Tahiti server, and Java Execution Environment (JEE), with a set of master-slave samples. By measuring message response times in milliseconds, it was found that two specific design patterns (V-agent and P-agent) resulted in improved time performance. [7]. The use of MA technology enables the creation of intelligent, highly distributed systems [35] due to its inherent qualities of flexibility, capability, adaptability, and independence. These systems are designed to leverage the power of MAs, which possess the ability to transfer all forms of code transmission between systems.

Path selection optimization algorithms are a critical component of MA technology. These algorithms enable agents to determine the optimal path to take when traveling from one system to another, based on a set of predefined criteria. Path selection optimization algorithms are important to many areas, such as telecommunications, transportation, networking, and robotics. The main objective is to determine the best path between nodes in a network, considering distance, cost, and speed. The best-known algorithms are Dijkstra's, ACO, genetic algorithm (GA), particle swarm optimization (PSO), and reinforcement learning (RL)-based approaches.

Dijkstra's algorithm (DA) [39] identifies the shortest paths from one node to all other nodes. It is implemented using data structures such as binary heaps. They are used in many applications, such as GPS and network routing, where finding the shortest path is important. DA is applied to find the shortest road on maps [36]; it modifies paths iteratively based on the network condition, which enables it to get the optimal routing solutions. It is also used to optimize the shortest delivery path based on real data of the weighted graph [37]. DA has been combined with retrospective data structures to calculate the shortest paths in protein interactions (PPI), particularly lung cancer-causing proteins [38]. However, DA does not fit graphs with negative edge weights. An improvement made to DA over time, such as the A* and Bellman-Ford algorithms. The A* algorithm is an extension of DA [40] to find the shortest path with minimum cost. It is used in many applications [41], such as video gaming, geographic information systems (GIS), robotics, and networks. While it guarantees the shortest path, it consumes more memory by exploring all nodes and computational complexity. The Bellman-Ford algorithm was developed to treat graphs with possible negative edge weights and reduce complexity [42, 43]. However, it is slower than DA and can't handle graphs with negative weight cycles.

One common optimization algorithm used in MA technology is the ACO algorithm [8, 9]. This algorithm is inspired by the foraging behavior of ants, where ants leave pheromone trails to guide other ants to food sources. In the ACO algorithm, agents leave virtual pheromone trails on the network as they travel, which other agents can follow to reach their destination more efficiently. The first use of the ANT system was an innovative algorithm that used the well-known traveling salesman problem (TSP) [10, 11, 12] as its application example. Although AS showed promising early outcomes, it failed to outperform the state-of-the-art algorithms for solving

the TSP [13]. Another optimization algorithm commonly used in MA technology is the GA [14]. This algorithm is based on the principles of natural selection and genetic mutation. Agents are assigned a set of parameters that define their behavior, and these parameters are then evolved over time using a process such as genetic mutation. This enables agents to adapt to changing network conditions and optimize their path selection based on the current situation. Currently, GAs is combined with other search methods in hybrid algorithms to increase performance and accuracy [44]. This integration improves the ability of GAs to find optimal solutions. However, for a large-scale problem, GAs can be extremely costly, and we could have an uncertain optimal path.

Another optimization algorithm derived from the social behavior of birds is called PSO. It searches through a group of possible solutions to find the optimal one. It has been used in many areas such as engineering [47], robotics [48], finance [49], and bioinformatics [50]. Sometime it's trapped in a solution as an optimal and stops searching [45]. Also, computational complexity increases as the population increases, and the final optimal solution is influenced by the initial populations [46]. RL is another method of path optimization, which learns the path from source to destination to maximize the cumulative rewards [51]. It explores different paths and uses rewards to learn the best path from the environment. Research continues to improve RL for all pathfinding applications [52]. But it requires a large space to learn and find the optimal path, and it uses deep learning, which can be computationally expensive. Also, the used reward function affects the ability to find the best path.

Hybrid algorithm that utilizes methods from all methods implemented to increase accuracy and improve effectiveness. GA and master-slave design patterns are proposed to optimize the path selection and migration of MAs in a timely and efficient manner. The proposed algorithm determines the best path through mathematical optimization techniques. Another method employs a numerical procedure and streamlining strategy to optimize the use of ACO calculation in selecting the optimal path. The results of these methods are then compared with similar works that utilize master-slave design pattern with the GA and Node Compression Algorithm (NCA) [21, 22]. To compare the performance of different agent platforms, researchers have used mathematical models and empirical formulas to predict metrics such as scalability and durability [23]. According to the results, JADE and James are the two best-performing platforms [24]. Additionally, researchers have used the ACO technique to optimize the pull-all data migration pattern in distributed networks, resulting in lower network load and transmission time compared to the existing pattern [25]. Table 1 describes optimization algorithms developed over time and their updates.

Table 1. Summary of algorithms used in optimization developed over time

Algorithm	Description	Update
Dijkstra's Algorithm	Identify the shortest paths between two nodes. The shortest path may be longer than some paths sometimes.	Reduce computational complexity and improve performance in dynamic, realistic environments [36, 37, 38].
A* Algorithm	It utilizes heuristics to guide the search towards the goal, potentially reducing the number of encountered nodes.	Refinements in heuristic functions and the integration of machine learning to adjust heuristics based on environmental feedback [41].
Bellman-Ford Algorithm	It calculates the shortest paths from the source to other vertices in the graph and considers the weights of negative edges.	The optimizations detect negative cycles and minimize the time complexity of large networks [42, 43].

(Continued)

Table 1. Summary of algorithms used in optimization developed over time *(Continued)*

Algorithm	Description	Update
Ant Colony Optimization (ACO)	A probabilistic method that mimics the behaviors of ants to find a path between their colony and a food location.	Hybrid approaches combine ACO and other optimization techniques to enhance convergence rates and solution quality.
Genetic Algorithm (GA)	It optimize solution using evolutionary approach such as mutation.	It combine genetic algorithms with other techniques to reduce computations and increase accuracy. For example, GANCSA results in minimizing the time for allocating the best path after 10 iterations [31].
Particle Swarm Optimization (PSO)	It improves problem solutions by iteratively improving candidate solutions with respect to a given quality metric.	It improve the algorithm's ability to handle large spaces. Also, adjust parameters based on the complexity of space and search state [46].
Reinforcement Learning (RL) Based Approaches	It explore different paths and use rewards to learn the best path from the environment.	Apply deep RL non-static environments where classical methods fail such as real-time network routing
Hybrid approaches	It combine different approaches to increase accuracy and reduce computations.	Suggest future research to improve their effectiveness and reduce limitations [24]. Proposed master-slave samples applying Aglet platform, the Tahiti server, and executing in java Environment to improve time performance with V and P-agent design patterns [7]. Compare agent platform performance parameters and found that JADE and James are the best two platforms in terms of performance, scalability, and durability [30].

Other algorithms designed to balance the tradeoff between exploration of new paths and exploitation of existing knowledge to find the most efficient path include PSO [15, 16], tabu search (TS) [17, 18], and simulated annealing (SA) [19]. Various techniques and methods are used to improve the efficiency and performance of MA systems in terms of path selection and migration. These methods involve the use of mathematical optimization techniques and numerical procedures and are compared with similar works to determine their effectiveness. Additionally, researchers have compared the performance of different agent platforms and optimized data migration patterns to reduce network load and transmission time.

The push-all-data strategy is a communication approach used in MA systems where agents transfer all the data they have collected to the next node on their path without any filtering or selection. This technique is convenient in scenarios where network bandwidth is not a limiting factor, as it simplifies data transfer between nodes. However, it can result in unnecessary data transmission, especially when data size is large and network resources are limited. Therefore, designing efficient path selection optimization algorithms is essential to minimize the impact of the push-all-data strategy on the performance of MA systems.

Several path selection optimization algorithms that use a push-all-data strategy have been proposed. For instance, a Q-learning-based algorithm [25] incentivizes agents to choose paths with high quality and low delay. A GA-based algorithm [26] considers the dynamic environment of the network and considers factors such as distance, congestion, and topology. Another algorithm, based on ACO [27], factors in the network topology and remaining energy of nodes to select the best path for agents. A heuristic-based algorithm [28] considers the network topology and energy consumption of nodes to determine the optimal path. Finally, a multi-objective algorithm [29] optimizes path selection based on objectives such as total travel time and energy consumption, using a GA. DA was used to determine the shortest path (STP) in the network [32]. A simulation is performed to find the shortest path while incorporating real-world traffic conditions into the cost of each edge. Experiments have proven the accuracy and effectiveness of DA in determining the shortest path in real-world applications.

The hybrid algorithm that combines PPO and ACO [33] is a powerful tool that uses RL and ACO features to determine the optimal path in the network.

Mobile agent technology enables the creation of intelligent and highly distributed systems that leverage the power of MAs. Path selection optimization algorithms are critical components of MA technology, and several optimization algorithms, such as ACO, GA, PSO, Tabu Search (TS), and Simulated Annealing (SA), have been proposed to balance the tradeoff between exploration of new paths and exploitation of existing knowledge to find the most efficient path. These algorithms consider factors such as distance, congestion, topology, energy consumption, and network conditions to select the best path for agents. Efficient path selection optimization algorithms are essential to minimize the impact of the push-all-data strategy on the performance of MA systems.

While path selection optimization algorithms for MAs offer many benefits, they also have some limitations. Some optimization algorithms, such as GAs, require a significant amount of computation to find an optimal solution. This can limit their usefulness in real-time scenarios where decisions need to be made quickly. Path selection algorithms that rely on network information can be impacted by network congestion. In highly congested networks, the algorithm may not be able to find the optimal path, leading to delays or reduced performance. Also, if agents don't have complete knowledge of the network, the algorithm may not be able to find the optimal path. MAs can pose security risks if they are not properly designed and implemented. For example, an agent could be intercepted and modified by an attacker, leading to data theft or other security breaches. In general, the performance of path selection optimization algorithms may decrease as the number of nodes in the network increases. This can limit their usefulness in large-scale systems.

Despite the advances in MA technologies, there is still a significant gap in optimal routes and route matching under real-time changes. Moreover, to the best of our knowledge, our work directly targets this gap–we propose an algorithm that considers quality of service parameters such as delay and jitter–which is crucial for paths with video streaming, for example, or real-time analysis computations based on IoT devices. The improvements we propose are timely. Nowadays, the operational efficiency of MAs is in dire need of improvement, as the overall system latency needs to be reduced to a minimum. Improving response times is also essential in all applications important to people's well-being, such as healthcare. In the case of healthcare, real-time data processing can save lives, while in smart cities, improving data efficiency can make urban infrastructure more responsive.

This paper proposes a new approach that integrates ACO with the path design pattern for efficient pathfinding in dynamic network environments. By applying real-time feedback mechanisms and adaptive routing protocols that dynamically respond to changes in network topology and traffic conditions, the limitations of traditional path selection algorithms are addressed. The results showed a good change in path efficiency and data processing speeds, as access time was reduced compared to traditional methods.

3 METHODS

To start, we developed a MA program that can navigate through a series of network hubs to execute tasks. The developed program is important for testing and demonstrating real-world application scenarios.

Task pattern design: specifically focused on routing through multiple destinations and always ensuring the next destination is reached before returning to

the resource. This methodology ensures reliability and efficiency in unstable routing environments.

Push-all data migration strategy: To deal with data volatility and the possibility of data loss during transfer, data migration. It's designed to determine the time required to process data between the hubs. The data size was taken into consideration to accurately estimate any changes affecting the process.

ACO implementation: The ACO algorithm is used to find the shortest path from source to destination. The ACO algorithm can handle any network problem, such as latency. The ACO's ability to adapt to any changes in the network is important to update route efficiency in any network condition.

Evaluation: To evaluate the system, we conducted seven tests for each case and computed the average.

Route efficiency: An ant-based algorithm is used to compute the shortest route from the source to the destination by sending multiple ants, and the route taken by the most ants was considered the shortest. This will reduce time and avoid bottlenecks in the network.

3.1 Implementation

The development of distributed systems has led to the creation of numerous techniques and algorithms that enable efficient data processing across multiple networked hubs. One such technique is the MA model, which involves the creation of an autonomous program that moves from one hub to another, performing tasks without requiring user intervention. To ensure that data is processed efficiently, a push-all data migration strategy is used to determine the time required to transfer data between hubs. This is complemented by the ACO algorithm, which allows for the identification of the shortest path between hubs by adjusting the pheromone level on edges based on the number of ants that have traversed them. In this way, the MA model, along with the push-all data migration strategy and ACO algorithm, provides an effective solution for distributed data processing. Figure 2 summarizes the architecture of the proposed work.

1. Mobile agent: The initial step involves the creation of a MA program that operates independently and traverses a network of computers. This program possesses features such as the ability to move from one hub to another to perform tasks without requiring user intervention. Our work focuses on data processing across hubs, from the starting point to the interface, and subsequently returning the results to the source. Figure 3 illustrates the architectural diagram of the MA model.
2. Itinerary design pattern: Initially, an "ItineraryAgent" of type "Agent" was defined. This agent includes the "ItineraryBehaviour" behavior of type "OneShotBehaviour," which sets the agent's itinerary and starts the relocation process. The execution of the agent is controlled by the methods "beforeMove" and "afterMove," which control the relocation and enable the execution of the task, respectively. The agent's task is performed in the "JobBehaviour" class of type "OneShotBehaviour," which is added to the agent whenever it arrives at a new destination. Before moving, the agent checks if the destination container exists through the Agent Management System (AMS). If the destination container does not exist, the agent attempts to relocate to the next destination in its itinerary. This check is carried out by the "GetAvailableLocationBehaviour" behavior, which is available in the system's provided model codes.

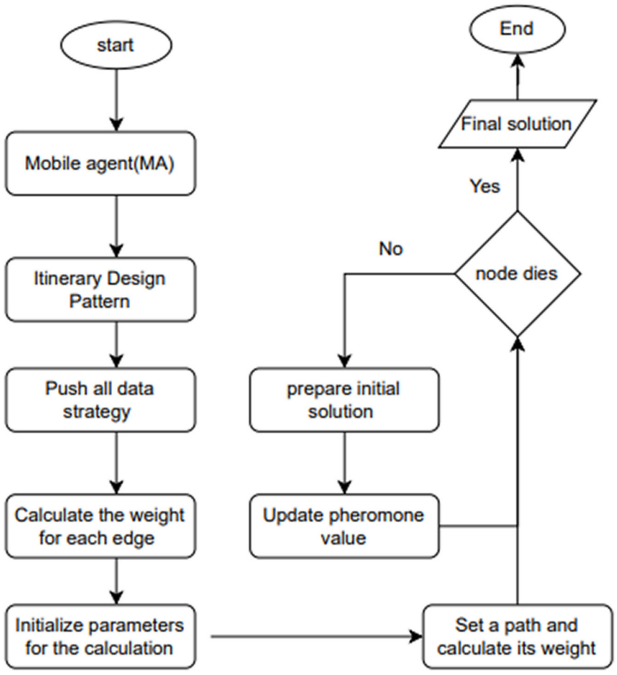


Fig. 2. Architecture of the proposed work

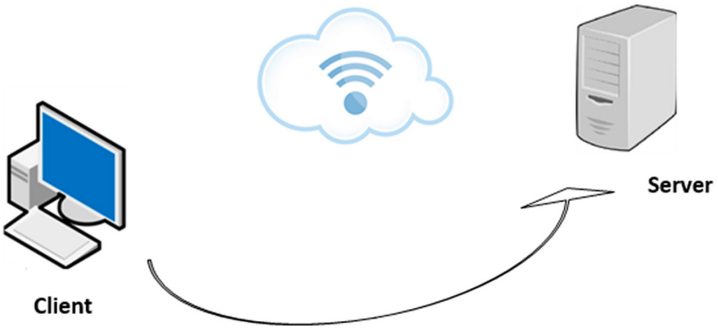


Fig. 3. Architecture of the mobile agent

3. Itinerary design pattern: Initially, an "ItineraryAgent" of type "Agent" was defined. This agent includes the "ItineraryBehaviour" behavior of type "OneShotBehaviour," which sets the agent's itinerary and starts the relocation process. The execution of the agent is controlled by the methods "beforeMove" and "afterMove," which control the relocation and enable the execution of the task, respectively. The agent's task is performed in the "JobBehaviour" class of type "OneShotBehaviour," which is added to the agent whenever it arrives at a new destination. Before moving, the agent checks if the destination container exists through the AMS. If the destination container does not exist, the agent attempts to relocate to the next destination in its itinerary. This check is carried out by the "GetAvailableLocationBehaviour" behavior, which is available in the system's provided model codes.
4. Push all data migration strategy: In the second stage, the push-all data migration strategy is utilized to determine the time required to transfer data between each hub and other hubs in the network. This strategy involves creating a graph based on the itinerary from the starting hub to the final hub, passing through intermediate hubs that transfer data. The migration is carried out to all the destinations that

the agent will visit, requiring the agent to know all its destinations in advance. In the three stages of this process, the time between each pair of edges is calculated. To perform these calculations, we used three different data sets: 10 KB, 50 KB, and 100 KB.

5. Ant colony optimization algorithm: After the data is pushed, the ACO algorithm proceeds in two steps. Firstly, as the ants move through the edges, they locally update the amount of pheromone on the visited edges using a local updating rule. Secondly, after all the ants have finished their routes, a global updating rule is applied to adjust the pheromone level on the edges that belong to the best route found so far, based on the number of ants that have traversed those edges. This allows the algorithm to identify the shortest path as the one with the most ants. The ACO algorithm effectively addresses several critical weaknesses inherent in MA systems. By utilizing pheromone trails for real-time feedback, ACO dynamically adapts to fluctuating network environments, ensuring stable performance. It enhances network resource utilization by analyzing traffic data and exploring multiple paths, thus avoiding bottlenecks and boosting overall efficiency. The decentralized structure of ACO allows for scalability in complex networks without a proportional increase in computational demands. Additionally, its strategy of exploring multiple paths enhances fault tolerance by providing alternatives when primary routes fail due to node disruptions. In the realm of IoT applications, ACO also improves energy efficiency by factoring in energy costs during path selection, which helps in conserving power and extending device battery life. Collectively, these features equip ACO to effectively surmount the unique operational challenges presented by MA systems.

4 RESULTS AND DISCUSSION

A MA and itinerary design pattern using JADE. The system utilizes the itinerary design pattern and JADE technology to facilitate MA movement between hubs in a network. During the transition phase, the push-all data strategy is employed to calculate the time between edges in the network. The time calculation is based on variable data and is expressed in milliseconds. To identify the optimal path, the system calculates the path weights for each path. The weight derived from the total time taken and the size of data transferred between hubs. This calculation involves mathematical equations that utilize Newton's law and the linear prediction model (LPM). The system employs a set of paths with different weights to find the optimal solution. The weight of an edge is used to improve the search algorithm. Specifically, the weight of each node is calculated using Newton's law and the volume of data transferred during the transition phase. The system uses advanced technologies and mathematical models to enable efficient MA movement between hubs in a network.

$$f(xi) = \alpha i, \forall i = 0,1\ldots\ldots\ldots n \tag{1}$$

$$Pn(xk) = \sum_{k=0}^{N} \alpha k \; ek(x) \tag{2}$$

By assuming n = 1, we derived the LPM as follows:

$$Pn(x1) = \sum_{k=0}^{N} \alpha k \, ek(x1) = \alpha 0 + \alpha 1(X - X0) \tag{3}$$

Using (1) and (2), we get the following formula:

$$\alpha 1 = \frac{f(x1) - f(x0)}{(x1 - x0)} = f(x0, x1) \tag{4}$$

The experiments were conducted on a personal computer with an Intel Core i7 processor and 16GB of RAM, running a 64-bit operating system. Java Execution Environment (JEE) and JADE were used during the experiments. The experiments were conducted in three stages, with each stage representing a different file size: 10KB, 50KB, and 100KB. To calculate the actual weight of each edge, the system employs Newton's law, which considers the time between nodes presented in Figure 4, as well as the size of the data transferred. The edges are color-coded—blue for 100KB, red for 50KB, and green for 10KB—to separate the metrics related to each data size.

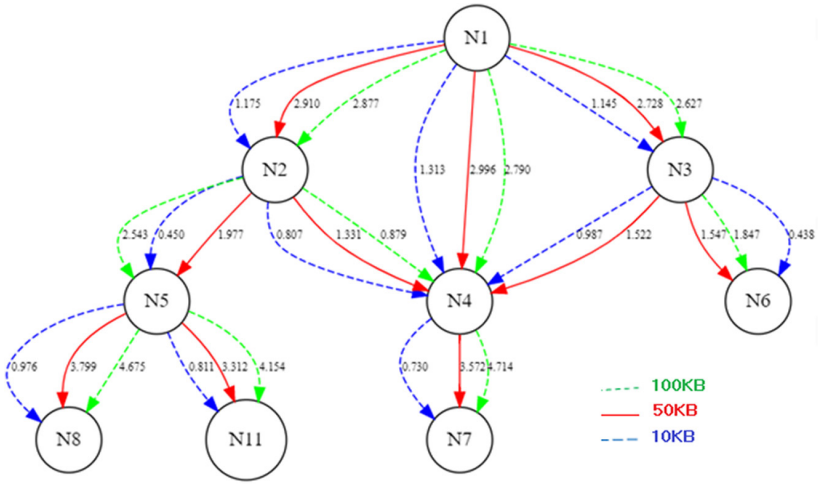


Fig. 4. Average time for 10KB, 50KB, and 100KB

Table 2 presents the values of $\alpha 0$ and $\alpha 1$, which were obtained through LPM computations for each edge. On the other hand, the edge weights are determined through a linear prediction formula that considers the size of the MA in KB, assuming a data size of 100 KB. The weights of each edge are shown in Figure 5.

Table 2. Values ($\alpha 0$) and ($\alpha 1$) and LPM for each (edge)

Set	α_0	α_1	LPM
N1–N2	1.18	1.74	–0.863 + 1.735X
N1–N3	1.15	1.58	–0.667 + 1.583X
N1–N4	1.31	1.68	–0.896 + 1.683X
N2–N4	0.81	0.52	0.384 + 0.524X
N3–N4	0.99	0.54	0.458 + 0.535X
N2–N5	0.45	1.53	–0.115 + 1.527X
N3–N6	0.44	1.11	–0.047 + 1.109X
N4–N7	0.73	2.84	1.498
N5–N8	0.98	2.82	–1.779 + 2.823X
N5–N11	0.81	2.5	–1.217 + 2.501X

The graph coverage algorithm involves checking all available paths between node (1) and node (20) to ensure that all paths are covered. This helps in defining the anchor points and determining the starting and ending points for generating a directed graph. By implementing this algorithm, we can successfully create a directed graph that covers all possible routes between the anchor points. The weights on the edges represent the distance or cost to traverse that edge. The graph can be used to model various real-world scenarios, such as transportation networks, communication networks, social networks, etc.

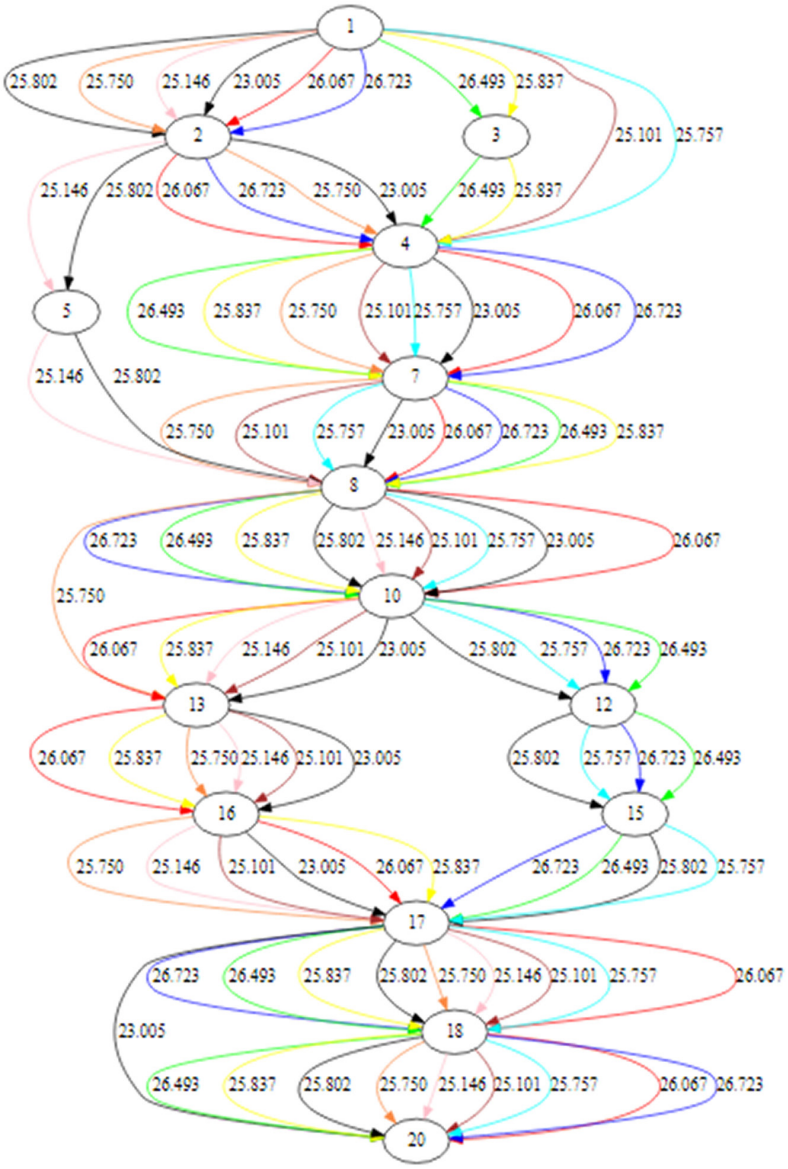


Fig. 5. First 10 possible paths and their weights

Figure 6 shows the time taken for each edge in milliseconds (MS) when using two algorithms, ACO and GA, with data sets of different sizes (10 KB, 50 KB, and 100 KB). The start node and end node for each edge are listed, along with the corresponding time taken by each algorithm for each data size. The comparison is also made with the time taken by the same data size when using a master-slave design pattern and GA, as described in the work of Allawi, Al-Hroob and Al-Shrouf [8].

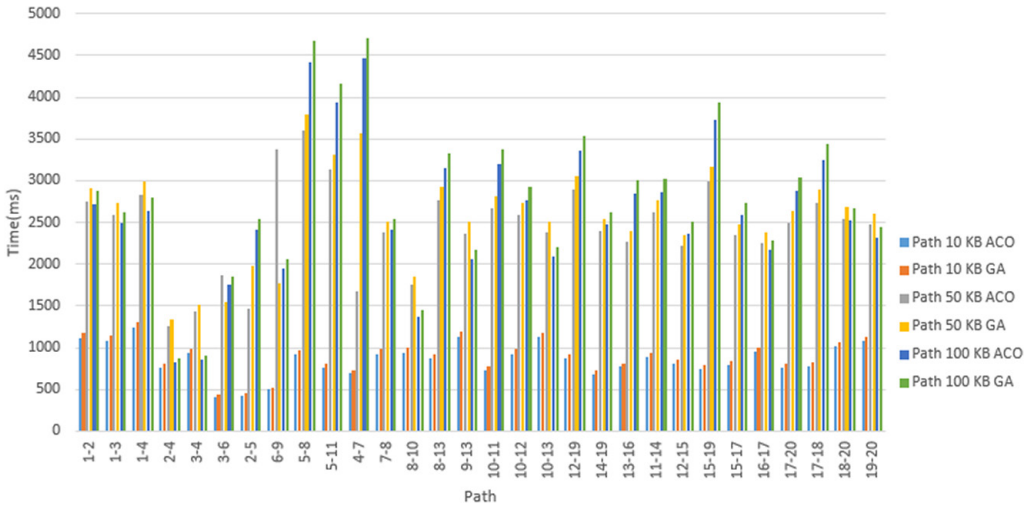
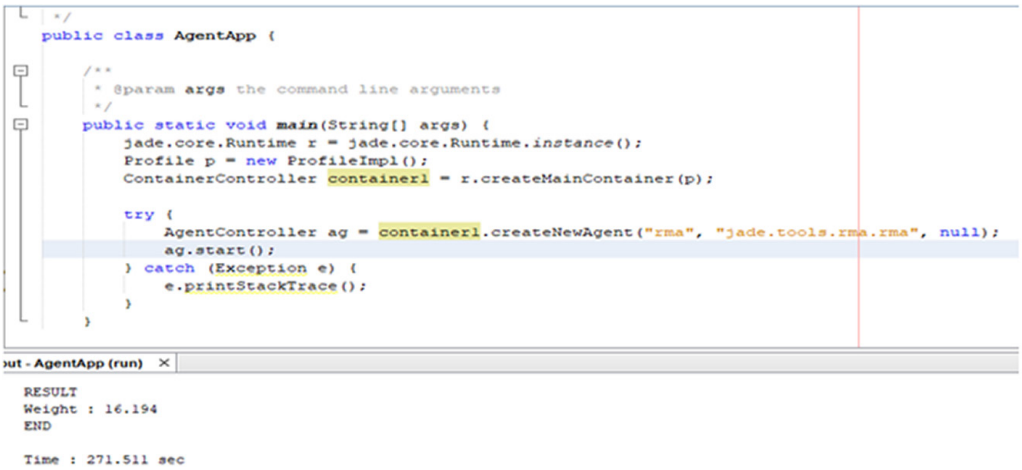


Fig. 6. Average time for 10KB, 50KB, and 100KB

Using the NetBeans Platform, we utilized artificial intelligence to optimize the execution time and calculate the shortest path. To achieve this, we employed an ACO algorithm, which involved sending a specific number of artificial ants that followed their path based on the pheromone, which depended on the total weight of all edges. We then calculated the time taken for the optimal path using this algorithm. The following parameters were used in this case:

- Q: 0.0005 – This parameter determines the amount of pheromone deposited.
- RHO: 0.2 – This parameter varies the level of pheromone evaporation.
- ALPHA: 0.01 – This parameter controls the importance of the pheromone trail.
- BETA: 9.5 – This parameter controls the importance of the distance between the source and destination.
- NUMBER_OF_ANTS: 500 – This parameter sets the number of artificial ants utilized in the algorithm.

The experiment shows that the path with the minimal weight was selected as shown in Figure 7, which illustrates the practical application of this scenario.



```
 */
public class AgentApp {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        jade.core.Runtime r = jade.core.Runtime.instance();
        Profile p = new ProfileImpl();
        ContainerController containerl = r.createMainContainer(p);

        try {
            AgentController ag = containerl.createNewAgent("rma", "jade.tools.rma.rma", null);
            ag.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
ut - AgentApp (run)
RESULT
Weight : 16.194
END

Time : 271.511 sec
```

Fig. 7. Illustrates the practical application of this scenario

When comparing adaptive algorithms, it is important to consider both quality and time as essential measures. Figure 8 shows a comparison between the results of 10 independent iterations of ACO, GA, and NCA, with time measurement being held constant.

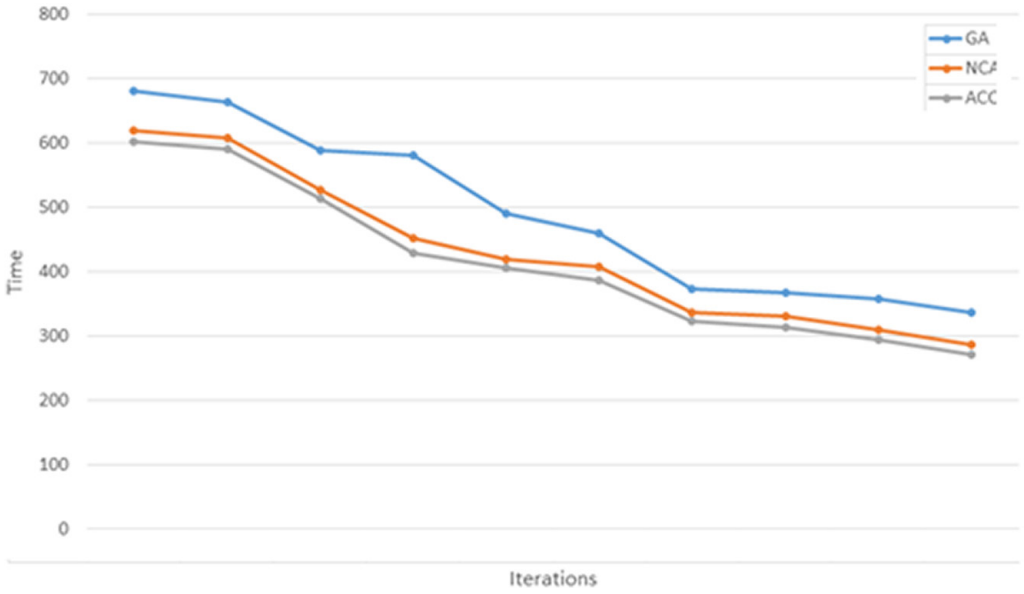


Fig. 8. Comparison between the results of 10 independent iterations of ACO, GA, and NCA

5 CONCLUSION AND FUTURE WORK

A proposed model utilizing the MA and Itinerary design pattern was developed in this study to determine the shortest path for MA migration between source and destination with minimum time using the ACO algorithm. The ACO algorithm was selected for its ability to identify the shortest path from a range of alternatives. The results of the ACO algorithm were compared to those of the GA and NCA algorithms, where the ACO algorithm was found to be more effective. The proposed model was developed using JADE within the Java development kit and NetBeans IDE environment.

The ACO algorithm showed better performance compared to classical methods such as master-slave design pattern with GA or NCA, but there are some challenges. For example, the stability of ACO performance can be sensitive to dynamic changes in the network environment, and ACO performance is great for high-density connected networks, but may need further improvement for less connected networks.

In future research, alternative agent design patterns should be explored, and other platforms should be considered for computation. Additionally, the DA should be compared to previous work, and the pull-all data strategy should be explored as an alternative to the push-all data strategy.

6 ACKNOWLEDGMENTS

This study was conducted without any external funding.

7 REFERENCES

[1] S. A. Mostafa, M. S. Ahmad, A. Mustapha, and M. A. Mohammed, "A concise overview of software agent research, modeling, and development," *Software Engineering*, vol. 5, no. 1, pp. 8–25, 2017.

[2] H. Zhang and Y. Vorobeychik, "Empirically grounded agent-based models of innovation diffusion: A critical review," *Artificial Intelligence Review*, vol. 52, pp. 707–741, 2019. https://doi.org/10.1007/s10462-017-9577-z

[3] L. Zhou, J. Lin, Y. Li, and Z. Zhang, "Innovation diffusion of mobile applications in social networks: A multi-agent system," *Sustainability*, vol. 12, no. 7, p. 2884, 2020. https://doi.org/10.3390/su12072884

[4] M. Kamran, A. Saeed, and A. Almaghthawi, "Federated-learning topic modeling based text classification regarding hate speech during COVID-19 pandemic," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 11, 2023. https://doi.org/10.14569/ijacsa.2023.0141157

[5] P. Bagga and R. Hans, "Applications of mobile agents in healthcare domain: A literature survey," *International Journal of Grid Distribution Computing*, vol. 8, no. 5, pp. 55–72, 2015. https://doi.org/10.14257/ijgdc.2015.8.5.05

[6] M. O. Oyediran *et al.*, "A survey on migration process of mobile agent," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, 2016. https://doi.org/10.9734/BJMCS/2016/26024

[7] F. A. Shrouf, A. Turani, A. A. Baker, and A. A. Omri, "Analysis of mobile agent optimization patterns," *British Journal of Applied Science & Technology*, vol. 4, no. 12, pp. 1841–1857, 2014. https://doi.org/10.9734/BJAST/2014/7093

[8] M. Dorigo and T. Stützle, "The ant colony optimization metaheuristic: Algorithms, applications, and advances," in *Handbook of metaheuristics, International Series in Operations Research & Management Science*, F. Glover and G. A. Kochenberger, Eds., vol. 57, Boston, MA: Springer, 2003, pp. 250–285. https://doi.org/10.1007/0-306-48056-5_9

[9] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999. https://doi.org/10.1162/106454699568728

[10] J. K. Lenstra and D. Shmoys, *The Traveling Salesman Problem: A Computational Study*, 2009.

[11] F. AlZyoud, M. Tarawneh, A. Almaghthawi, and A. Altalidi, "A new approach for cluster head selection in wireless sensor networks," *International Journal of Online & Biomedical Engineering*, vol. 20, no. 9, pp. 39–50, 2024. https://doi.org/10.3991/ijoe.v20i09.48795

[12] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, vol. 840. Springer.

[13] M. Dorigo and T. Stützle, "Ant colony optimization: Overview and recent advances," in *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, M. Gendreau and J. Y. Potvin, Eds., vol. 272, Cham, Switzerland: Springer, 2019, pp. 311–351. https://doi.org/10.1007/978-3-319-91086-4_10

[14] S. Mirjalili, "Genetic algorithm," *Evolutionary Algorithms and Neural Networks: Theory and Applications*, vol. 780, Cham, Switzerland: Springer, 2019, pp. 43–55. https://doi.org/10.1007/978-3-319-93025-1_4

[15] F. Y. Alzyoud, M. Tarawneh, A. Almaghthawi, A. Altalidi, I. Asiri, and M. Alrehaili, "Optimizing broadcast utilization for efficient disaster management using wireless Ad Hoc networks and novel energy-saving algorithms," *International Journal of Interactive Mobile Technologies*, vol. 18, no. 20, pp. 142–156, 2024. https://doi.org/10.3991/ijim.v18i20.49395

[16] T. M. Shami, S. Mirjalili, Y. Al-Eryani, K. Daoudi, S. Izadi, and L. Abualigah, "Velocity pausing particle swarm optimization: A novel variant for global optimization," *Neural Computing and Applications*, vol. 35, pp. 9193–9223, 2023. https://doi.org/10.1007/s00521-022-08179-0

[17] A. Dirin and C. A. Saballe, "Machine learning models to predict students' study path selection," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 16, no. 1, pp. 158–183, 2022. https://doi.org/10.3991/ijim.v16i01.20121

[18] C. J. Joshua, P. Jayachandran, A. Q. Md, A. K. Sivaraman, and K. F. Tee, "Clustering, routing, scheduling, and challenges in bio-inspired parameter tuning of vehicular Ad Hoc networks for environmental sustainability," *Sustainability*, vol. 15, no. 6, p. 4767, 2023. https://doi.org/10.3390/su15064767

[19] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993. https://doi.org/10.1214/ss/1177011077

[20] A. Almaghthawi, F. Bourennani, and I. R. Khan, "Differential evolution-based approach for tone-mapping of high dynamic range images," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 7, 2020. https://doi.org/10.14569/IJACSA.2020.0110745

[21] F. Al-Refa'e, F. Al-Shrouf, and S. R. Masadeh, "Development of optimized itinerary agent design pattern using ant-colony algorithm," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 10, no. 1, pp. 75–81, 2021. https://doi.org/10.30534/ijatcse/2021/111012021

[22] M. O. Oyediran, T. M. Fagbola, S. O. Olabiyisi, and E. O. Omidiora, "Development of an optimized mobile agent migration pattern for pull-all data strategy," *Journal of Advances in Mathematics and Computer Science*, vol. 16, no. 6, pp. 1–8, 2016. https://doi.org/10.9734/BJMCS/2016/26024

[23] O. M. Terentiev, T. I. Prosiankina-Zharova, V. A. Lahno, and Y. V. Usatiuk, "The features of the predictive computing modeling power system load in terms of reforming energy market," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 2, pp. 163–182, 2020.

[24] V. Selvi and R. Umarani, "Comparative analysis of ant colony and particle swarm optimization techniques," *International Journal of Computer Applications*, vol. 5, no. 4, pp. 1–6, 2010. https://doi.org/10.5120/908-1286

[25] Y. Xie, Q. Zhang, and L. Zhou, "A novel path selection optimization algorithm based on q-learning for mobile agent networks," in *Proceedings of the 2018 3rd International Conference on Intelligent Transportation Engineering*, Singapore, 2018, pp. 247–252. https://doi.org/10.1145/3235146.3235156

[26] J. Hu, H. Chen, and X. Tang, "Path selection optimization for mobile agent networks with dynamic environments," *IEEE Access*, vol. 6, pp. 60519–60531, 2018. https://doi.org/10.1109/ACCESS.2018.2879472

[27] J. You, X. Zhang, and X. Yang, "A path selection optimization algorithm for mobile agent networks based on ant colony optimization," in *Proceedings of the 2015 International Conference on Cloud Computing and Big Data (CCBD 2015)*, 2015, pp. 356–361.

[28] M. Yang and Y. Zhao, "A path selection algorithm for mobile agents with a push-all-data strategy," *Journal of Computer and System Sciences*, vol. 81, no. 2, pp. 363–372, 2015. https://doi.org/10.1016/j.jcss.2014.06.008

[29] X. Wang and J. Liu, "Path selection optimization for mobile agent networks with multi-objective based on push-all-data strategy," *International Journal of Grid and Distributed Computing*, vol. 11, no. 7, pp. 21–32, 2018.

[30] F. Alshrouf, S. R. Masadeh, and F. Alzyoud, "Mathematical model for comparing performance evaluation of mobile agent platforms," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 2, pp. 315–329, 2020.

[31] R. Q. Allawi, A. T. Imam, A. Alhroob, and F. Al-Shrouf, "GANCSA: A novel approach for finding the best path for migration of mobile-agents," *Pervasive and Mobile Computing*, vol. 83, p. 101596, 2022. https://doi.org/10.1016/j.pmcj.2022.101596

[32] F. B. Zhan, "Three fastest shortest path algorithms on real road networks: Data structures and procedures," *Journal of Geographic Information and Decision Analysis*, no. 1, pp. 69–82, 1997.

[33] G. P. Dubey *et al.*, "Optimal path selection using reinforcement learning based ant colony optimization algorithm in IoT-Based wireless sensor networks with 5G technology," *Computer Communications*, vol. 212, pp. 377–389, 2023. https://doi.org/10.1016/j.comcom.2023.09.015

[34] J. J. Nga, *Master-Controlled Networking for Mobile Robotic Application*, UTAR, 2023.

[35] A. Sharma and C. Diwaker, "Multi-agent-based load balancing in mobile edge computing," in *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, M. Gendreau and J. Y. Potvin, Eds., vol 272, Cham, Switzerland: Springer, 2024, pp. 469–478. https://doi.org/10.1007/978-981-97-0700-3_36

[36] L. S. Jabbar, E. I. Abass, and S. D. Hasan, "A modification of shortest path algorithm according to adjustable weights based on Dijkstra algorithm," *Engineering and Technology Journal*, vol. 41. no. 2, pp. 359–374, 2023. https://doi.org/10.30684/etj.2022.136107.1296

[37] A. Almaghthawi, E. A. A. Ghaleb, N. A. Akbar, L. Asiri, M. Alrehaili, and A. Altalidi, "Federated-learning intrusion detection system based blockchain technology," *International Journal of Online & Biomedical Engineering*, vol. 20, no. 11, pp. 16–30, 2024. https://doi.org/10.3991/ijoe.v20i11.49949

[38] P. K. Rangarajan, B. M. Gurusamy, E. Rajasekar, S. I. Venkata, and S. Chereddy, "Retroactive data structure for protein–protein interaction in lung cancer using Dijkstra algorithm," *International Journal of Information Technology*, vol. 16, no. 2, pp. 1239–1251, 2024. https://doi.org/10.1007/s41870-023-01557-4

[39] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290. https://doi.org/10.1145/3544585.3544600

[40] C. Han and B. Li, "Mobile robot path planning based on improved A* algorithm," in *2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2023, pp. 672–676. https://doi.org/10.1109/ITAIC58329.2023.10408799

[41] J. Rao *et al.*, "Path planning for dual UAVs cooperative suspension transport based on artificial potential field-A* algorithm," *Knowledge-Based Systems*, vol. 277, p. 110797, 2023. https://doi.org/10.1016/j.knosys.2023.110797

[42] V. R. Gannapathy, V. Narayanamurthy, S. K. Subramaniam, A. F. B. T. Ibrahim, I. S. M. Isa, and S. Rajkumar, "A mobile and web-based security guard patrolling, monitoring and reporting system to maintain safe and secure environment at premises," *International Journal of Interactive Mobile Technologies*, vol. 17, no. 11, pp. 4–14, 2023. https://doi.org/10.3991/ijim.v17i11.35483

[43] M. Aliyan, M. Z. Hasan, H. Qayoom, M. Z. Hussain, S. Nosheen, and M. Mustafa, "Analysis and performance evaluation of various shortest path algorithms," in *2024 3rd International Conference for Innovation in Technology (INOCON)*, 2024, pp. 1–16. https://doi.org/10.1109/INOCON60754.2024.10512150

[44] H. Mikram, S. El. Kafhali, and Y. Saadi, "HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment," *Simulation Modelling Practice and Theory*, vol. 130, p. 102864, 2024. https://doi.org/10.1016/j.simpat.2023.102864

[45] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Computing*, vol. 22, pp. 387–408, 2018. https://doi.org/10.1007/s00500-016-2474-6

[46] M. Juneja and S. K. Nagar, "Particle swarm optimization algorithm and its parameters: A review," in *2016 International Conference on Control, Computing, Communication and Materials (ICCCCM)*, 2016, pp. 1–5. https://doi.org/10.1109/ICCCCM.2016.7918233

[47] B. S. G. de Almeida and V. C. Leite, "Particle swarm optimization: A powerful technique for solving engineering problems," in *Swarm Intelligence-Recent Advances, New Perspectives and Applications*, 2019, pp. 31–51.

[48] J. Zhang, Y. Lin, and M. Zhou, "Virtual-source and virtual-swarm-based particle swarm optimizer for large-scale multi-source location via robot swarm," *IEEE Transactions on Evolutionary Computation*, 2024. https://doi.org/10.1109/TEVC.2024.3391622

[49] J. Liu and S. Fu, "Financial big data management and intelligence based on computer intelligent algorithm," *Scientific Reports*, vol. 14, no. 1, p. 9395, 2024. https://doi.org/10.1038/s41598-024-59244-8

[50] B. Soni, S. Roy, and S. Warsi, "Particle swarm optimization in bioinformatics, image processing, and computational linguistics," in *Research Anthology on Bioinformatics, Genomics, and Computational Biology, Hershey*, PA: IGI Global, 2024, pp. 1270–1292. https://doi.org/10.4018/979-8-3693-3026-5.ch056

[51] F. M. Luo, T. Xu, H. Lai, X. H. Chen, W. Zhang, and Y. Yu, "A survey on model-based reinforcement learning," *Science China Information Sciences*, vol. 67, 2024. https://doi.org/10.1007/s11432-022-3696-5

[52] S. Milani, N. Topin, M. Veloso, and F. Fang, "Explainable reinforcement learning: A survey and comparative review," *ACM Computing Surveys*, vol. 56, no. 7, pp. 1–36, 2023. https://doi.org/10.1145/3616864

8 AUTHORS

Faisal Alzyoud is with the Department of Computer Science, Isra University, Amman, Jordan.

Monther Tarawneh is with the Department of Information Technology, College of Information Technology and Communications, Tafila Technical University, Tafila, Jordan.

Faiz Al-Shrouf is with the Department of Software Engineering, Faculty of Information Technology, Isra University, Amman, Jordan.

Ahmed Almaghthawi is with the Department of Computer Science, College of Science & Art at Mahayil, King Khalid University, Abha 62529, Saudi Arabia (E-mail: aalmaghthwi@kku.edu.sa).

Meaad Alrehaili is with the Department of Computer Sciences and Artificial Intelligence, Collège of Computer Science, and Engineering, University of Jeddah, Jeddah, Saudi Arabia.

Hasan Kanaker is with the Department of Information Security, Faculty of Information Technology, Petra University, Amman, Jordan.