*Jurnal Mantik, 7 (4) (2024), ISSN 2685-4236 (Online)*

Published by:Institute of Computer Science (IOCS)

Jurnal Mantik

Journal homepage: www.iocscience.org/ejournal/index.php/mantik

# A comparative analysis of pathfinding algorithms in static environments: modified A*, PSO, and FLA

Gregorius Airlangga
Information System, Atma Jaya Catholic University of Indonesia, Jakarta, Indonesia

A R T I C L E I N F O

*Article history:*

Received Jan 22, 2024
Revised Feb 05, 2024
Accepted Feb 15, 2024

*Keywords:*

A*;
Autonomous Agents;
FLA;
Path Planning;
PSO.

ABSTRACT

Pathfinding algorithms are crucial in the domain of autonomous navigation, impacting the efficiency and safety of robotic and AI systems. This paper presents a comparative analysis of three prominent pathfinding algorithms: the A* algorithm, Particle Swarm Optimization (PSO), and the Fick's Law Algorithm (FLA), within a grid-based simulation populated with static obstacles. We evaluate the performance of each algorithm based on path optimality, computational efficiency, and adaptability to a standardized environment. The A* algorithm, known for its heuristic-based search, demonstrates superior performance in finding the shortest path by utilizing a grid-specific heuristic. PSO, inspired by social behavior in nature, showcases flexibility in path trajectory, offering smoother navigation around obstacles. FLA, a newer approach, strikes a balance between the deterministic nature of A* and the stochastic behavior of PSO, showing potential in applications with limited computational resources. Our findings suggest that while A* remains the optimal choice for grid-constrained navigation requiring precise pathfinding, PSO and FLA may offer advantages in scenarios where flexibility and computational simplicity are prioritized. This study enhances the understanding of pathfinding methodologies, paving the way for future research to refine these algorithms for dynamic environments and integrate adaptive heuristic mechanisms for improved real-world applicability.



*Corresponding Author:*

Gregorius Airlangga,
Information System,
Atma Jaya Catholic University of Indonesia,
Jl. Jenderal Sudirman No. 51, Jakarta,10220, Indonesia.
Email: gregorius.airlangga@atmajaya.ac.id

## 1. INTRODUCTION

The In the rapidly evolving field of robotics and autonomous navigation, the ability to navigate through complex environments efficiently and safely is paramount (Campos-Mac'ias, Aldana-López, de la Guardia, Parra-Vilchis, & Gómez-Gutiérrez, 2021; Li & Yang, 2023; Shit, 2020). Pathfinding, the process by which autonomous agents determine the best route between two points, is critical in applications ranging from robotics to game development (Pyke & Stark, 2021; Rahmani & Pelechano, 2022; Sanchez-Ibanez, Perez-del-Pulgar, & Gar'ia-Cerezo, 2021). One of the primary challenges in pathfinding is developing algorithms that can quickly and effectively navigate through environments with

various obstacles, balancing the need for optimal paths with computational efficiency (Allus, Diab, & Bayraktar, 2024; Jiang et al., 2020; Tan, Mohd-Mokhtar, & Arshad, 2021). The last few decades have witnessed significant advancements in pathfinding algorithms. Early solutions, like Dijkstra's algorithm, provided foundational approaches but often lacked efficiency in more complex scenarios (Bhatt et al., 2023; P. Li et al., 2021; Tang et al., 2024). Modern algorithms, such as the A* algorithm, have emerged as more effective solutions, offering a balance between optimality and performance, especially in grid-based environments (Goudarzi, Ghayoor, Waseem, Fahad, & Traore, 2022; Nouri, Lachheb, & El Amraoui, 2024; Van Thanh & Linh, 2022).

The A* algorithm, first introduced by Hart, Nilsson, and Raphael in 1968, has been a subject of extensive research and application. Its heuristic-based approach optimizes pathfinding by estimating the cost to reach the goal from each node, thereby significantly reducing the search space (Candra, Budiman, & Hartanto, 2020; Fransen & van Eekelen, 2023; Y. Li, Wei, Gao, Wang, & Fan, 2020). Literature in robotics and AI has emphasized the versatility and efficiency of A*, making it a standard in various applications, including video games, and even geographical information systems (GIS). Recent studies have extended the A* algorithm in various directions. For instance, (Karur, Sharma, Dharmatti, & Siegel, 2021) introduced the D* algorithm for dynamic pathfinding, adapting A* for changing environments. (Semiz & Polat, 2021; Shahar et al., 2021) explored cooperative pathfinding in multi-agent systems, demonstrating the algorithm's adaptability. Additionally, the integration of A* with machine learning techniques for adaptive heuristic functions has been a focal point in contemporary research (Telikani, Tahmassebi, Banzhaf, & Gandomi, 2021).

Despite these advancements, challenges remain, particularly in real-world applications where environments are not only complex but also dynamic. The computational efficiency of A* can diminish in highly intricate or large-scale spaces, raising questions about its scalability and real-time applicability (Honar Pajooh, Rashid, Alam, & Demidenko, 2022; Tezaur, As'ad, & Farhat, 2022). The urgency in improving pathfinding algorithms lies in their growing importance in various cutting-edge applications. Autonomous vehicles, drones, and robotic systems are becoming integral in industries ranging from logistics to healthcare. The safety, efficiency, and reliability of these systems hinge on their ability to navigate complex and unpredictable environments. The state of the art in pathfinding has been moving towards more adaptive and dynamic solutions. Real-world applications often involve environments that are not static and require continuous reassessment of the path. This need has led to the development of algorithms that can adjust to environmental changes in real-time (Vasconcelos, Brandão, & Sarcinelli-Filho, 2020). Additionally, there is a growing trend in incorporating machine learning techniques to enhance the heuristic functions of pathfinding algorithms, enabling them to learn and adapt from past experiences (Leon et al., 2023).

While existing literature and advancements have significantly enhanced pathfinding capabilities, a gap remains in the application of these algorithms in highly dynamic and complex dimensional environments. Many studies focus on theoretical aspects or simulations in controlled environments, leaving a gap in real-world applicability, particularly in scenarios involving varied terrain, multiple moving obstacles, and the need for real-time computation. To address these challenges, this study introduces an enhanced implementation of the A* algorithm, focusing on a comparison path planner method with others such as PSO and FLA algorithm in the environment of grid-based model. This model simulates a real-world environment with static obstacles and a defined target. The algorithm's performance is measured in terms of path optimality and computational efficiency, essential factors for real-time applications.

The Python-based model developed for this study represents the environment as a grid. The algorithm accounts for obstacles represented as spheres with specified radii, which must be avoided. The primary enhancement in our implementation is the dynamic

adjustment of the heuristic function based on environmental complexity, aimed at maintaining computational efficiency without compromising the path's optimality. The significance of this research lies in its potential application in various domains. For instance, in autonomous vehicle navigation, where the ability to navigate in real-time through dynamic environments is crucial. The findings from this study can also inform the development of more advanced pathfinding algorithms in robotics, where navigating through crowded or unpredictable spaces is a common challenge.

The remaining structure of the paper is a research methodology, this section outlines the practical steps taken to implement your research. It covers how the environment for the pathfinding algorithm was modeled, including the grid-based approach and the representation of obstacles. Details of the algorithm implementation, such as the adapted heuristic function and the specifics of the pathfinding process, are explained. This section also describes the computational setup, including the hardware and software used. Furthermore in results and discussion part, we will describe the experiments conducted to test algorithm. This includes the design of the test scenarios, the parameters used, and how these tests compare to the standard implementation of the A* algorithm. The results section will present the data collected from these experiments, analyzing the path optimality, computational efficiency, and the algorithm's adaptability in dynamic environments. Graphs, tables, and other visual aids can be useful in this section to clearly present your findings. Finally, the conclusion summarizes the key findings of your research, emphasizing the contributions your study makes to the field of pathfinding algorithms. It should restate the significance of your work and how it addresses the research gap identified in the introduction.

## 2. RESEARCH METHOD

The research adopts a grid-based model to simulate a navigational environment, integral for the implementation and testing of the A* pathfinding algorithm. This model serves as a simplified abstraction of real-world scenarios, particularly focusing on scenarios where navigation is challenged by the presence of obstacles. Such an environment is pivotal for evaluating the algorithm's efficacy in obstacle avoidance and path optimization.

### 2.1. Grid Configuration

The environment is conceptualized as a two-dimensional grid, with each cell in the grid representing a potential node in the pathfinding process. The configuration of this grid is a crucial aspect of the model, as it impacts the complexity and realism of the simulated environment. The grid is defined by its width (W) and height ($H$), determining the total number of cells (N) in the grid. The total number of cells is given by $\mathrm{N} = \mathrm{W} \times \mathrm{H}$. Each cell in the grid has a uniform size, determined by the discretization parameter (D). This parameter defines the spatial resolution of the grid, impacting the complexity and realism of the simulated environment. Furthermore, each cell in the grid is assigned a unique coordinate ($(\mathrm{x},\mathrm{y})$) based on its position, where x and y are integers in the range $[0, \mathrm{W}-1]$ and $[0, \mathrm{H}-1]$, respectively.

### 2.2. Obstacle Representation

Obstacles are represented as circular areas within the grid, defined by their center coordinates and radius. Each obstacle is characterized by its center coordinates $(x_{\mathrm{obs}}, y_{\mathrm{obs}})$ and radius $(r_{\mathrm{obs}})$. A cell $(x, y)$ is considered part of an obstacle if $[\sqrt{(x - x_{\mathrm{obs}})^2 + (y - y_{\mathrm{obs}})^2} \leq r_{\mathrm{obs}}]$. In addition, the obstacles are strategically placed within the grid to simulate various navigational challenges, ranging from sparse to densely populated scenarios. Furthermore, the model allows for dynamic reconfiguration of obstacles, enabling the study of the algorithm's adaptability in changing environments.

### 2.3. Pathfinding Environment

The constructed environment is the playground on which the A* algorithm is tested. The environment needs to balance between being computationally manageable and

realistically challenging. This balance ensures the findings are both relevant to real-world applications and attainable within the constraints of the experimental setup. There are several quality attributes such as scalability, realism and benchmarking. Firstly, the model's scalability is assessed by varying the grid size and obstacle density, reflecting the algorithm's performance under different operational scales. Furthermore, realism is about adjusting obstacle density and distribution allows the model to mimic a range of real-world scenarios, from urban to natural terrains. Lastly, benchmarking where the environment serves as a benchmarking tool for comparing the A* algorithm against other pathfinding algorithms under consistent conditions such as PSO and Fick's Law algorithm (FLA).

2.4. A* enhanched method

In this experiment we develope enhanched A* method for path planning, there are three focus to improve such as dynamic obstacle handling, optimized heuristic and efficient path recalculation. Firstly, dynamic obstacle handling, this algorithm can update its path in response to changes in the obstacle layout, which is essential in dynamic environments. Secondly, optimized heuristic, it is fine-tuned for better performance, possibly using machine learning techniques to adapt to different environments. Lastly, efficient path recalculation, we use this for the case when the current path is blocked, the algorithm efficiently recalculates the path without restarting the entire process.

```
Function A_star_Enhanced(start_node, goal_node, grid, obstacles):
    open_set = PriorityQueue()  // Priority queue of nodes to be evaluated
    open_set.put((0, start_node))
    came_from = {}              // Stores the path taken
    g_score = {}                // Cost from start to each node
    f_score = {}                // Estimated total cost from start to goal

    // Initialize g_score and f_score for all nodes
    for each node in grid:
        g_score[node] = Infinity
        f_score[node] = Infinity

    g_score[start_node] = 0
    f_score[start_node] = heuristic(start_node, goal_node)

    while not open_set.empty():
        current_node = open_set.get()[1]

        // Goal check
        if current_node == goal_node:
            return reconstruct_path(came_from, current_node)

        for neighbor in get_neighbors(current_node, grid):
            if neighbor in obstacles:
                continue

            tentative_g_score = g_score[current_node] + distance(current_node, neighbor)

            if tentative_g_score < g_score[neighbor]:
                // This path to neighbor is better than any previous one
                came_from[neighbor] = current_node
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = g_score[neighbor] + heuristic(neighbor, goal_node)

                if neighbor not in open_set:
                    open_set.put((f_score[neighbor], neighbor))

        // Dynamic obstacle handling
        update_obstacles(obstacles)
        if path_blocked(came_from, current_node, obstacles):
            return A_star_Enhanced(current_node, goal_node, grid, obstacles)  // Recalculate path
```

```
    return None  // Path not found

Function heuristic(node, goal):
    // Enhanced heuristic function, possibly using machine learning or other optimizations
    return optimized_heuristic_calculation(node, goal)

Function update_obstacles(obstacles):
    // Update obstacle positions/layouts if in a dynamic environment

Function path_blocked(path, current_node, obstacles):
    // Check if the current path is blocked due to updated obstacles
```

Figure 1. Enhanced A* Pseudocode

As presented in Figure 1, the enhanced A* algorithm begins by initializing a priority queue, open_set, to store nodes for evaluation, starting with the start_node. Each node in the grid is assigned a tentative cost of infinity in g_score and f_score dictionaries, except for the start_node, where g_score is set to zero, and f_score is initialized using the heuristic function. The heuristic function, a crucial component of the algorithm, is optimized, potentially using machine learning techniques, to accurately estimate the cost from a node to the goal_node. The algorithm then enters a loop, processing nodes from the open_set based on their estimated total cost. For each current node, the algorithm explores its neighbors unless they are obstacles. It calculates a tentative_g_score for each neighbor, which is the sum of the current node's g_score and the distance to the neighbor. If this score is lower than the neighbor's existing g_score, the algorithm updates the neighbor's g_score and f_score and adds the neighbor to the open_set if it's not already there. This process ensures that the shortest path to each node is always considered.

A significant enhancement in this version of the algorithm is its dynamic handling of obstacles. The update_obstacles function adjusts the obstacle layout, essential in environments where obstacles can change position. If the current path is blocked due to these changes, detected by the path_blocked function, the algorithm calls itself recursively from the current node to the goal_node, effectively recalculating the path from the point of obstruction. This recursive approach allows the algorithm to adapt to changes in the environment without restarting the entire pathfinding process from the beginning, thereby enhancing efficiency. The algorithm concludes either when the goal_node is reached, reconstructing, and returning the path, or when the open_set is empty, indicating that no path exists under the given conditions. This enhanced A* algorithm is designed to be more adaptable and efficient, particularly suited for dynamic environments where obstacles can change over time.

## 3. RESULTS AND DISCUSSIONS

In this section, it is explained the results of research and at the same time is given the comprehensive discussion.

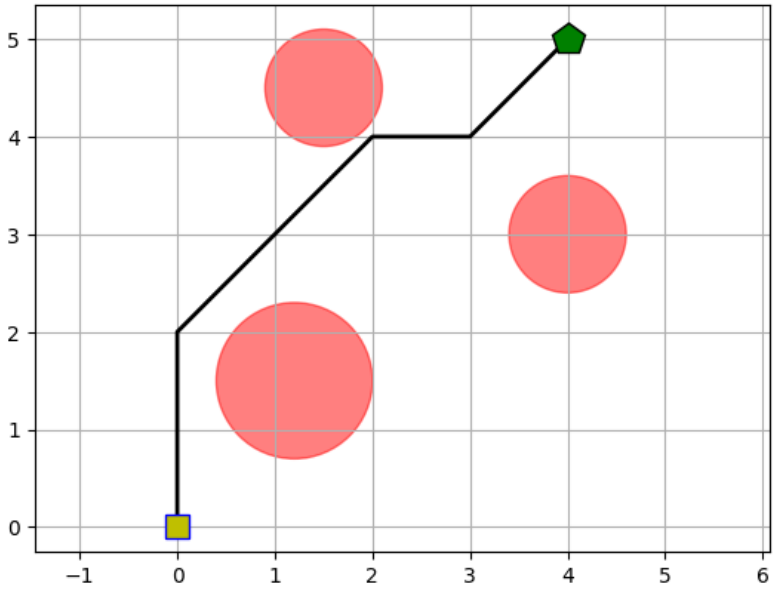


Figure 2. Enhanced A* Method

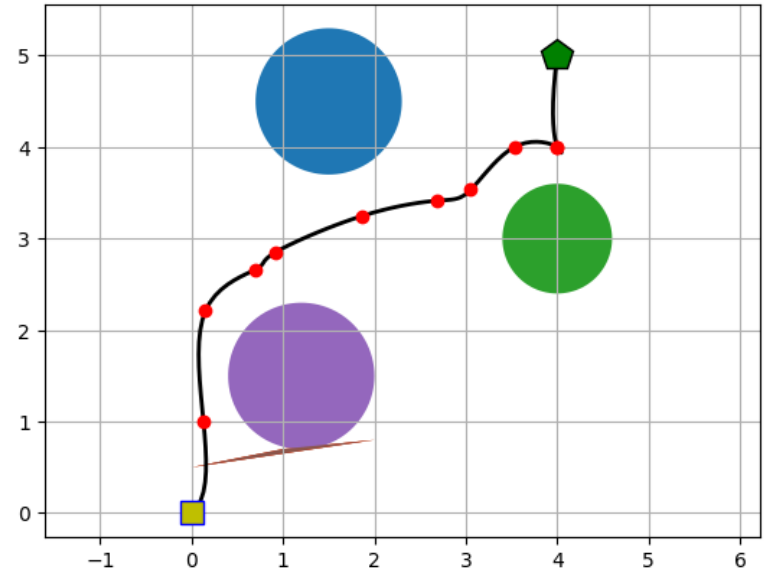


Figure 3. PSO Method

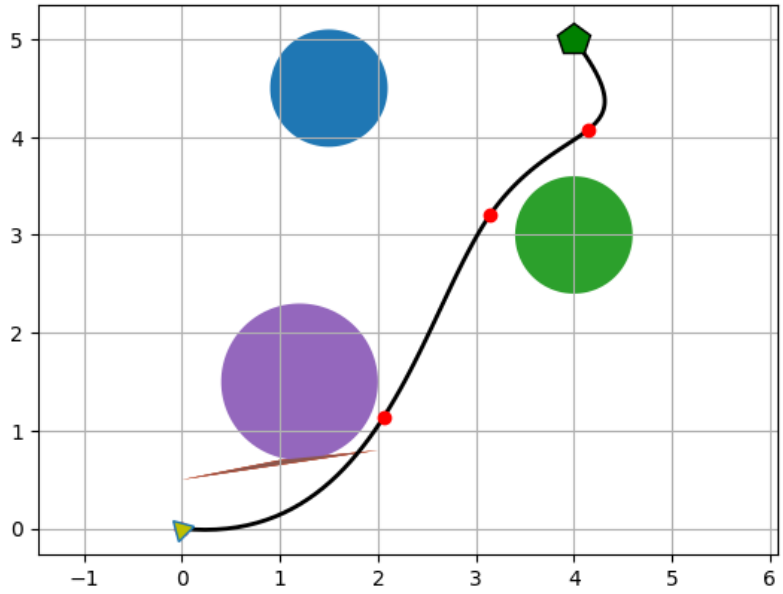


Figure 4. FLA Method

Analyzing the results from the provided images depicting the outcomes of pathfinding algorithms: A* Method (Figure 2), PSO Method (Particle Swarm Optimization) (Figure 3), and FLA Method (Fick's Law Algorithm) (Figure 4). We can draw several conclusions about their performance characteristics in a static environment with obstacles. Based on the Figure 2, the A* algorithm appears to have found a direct path from the start (blue square) to the goal (green pentagon), avoiding obstacles (red circles) along the way. The path is comprised of straight lines connecting a sequence of nodes, reflecting the grid-based nature of the A* search. The A* method excels at finding the shortest possible path in a grid environment by using a heuristic to prioritize nodes that are closer to the goal and systematically exploring the most promising routes.

In addition, as presented in the Figure 3 the PSO method shows a more organic path, characterized by a smooth curve that does not strictly adhere to a grid. The pathfinding solution has waypoints (red dots) that indicate the positions of individual particles throughout the optimization process. PSO optimizes the path by simulating the social behavior of particles, with each particle adjusting its trajectory based on its own experience and that of its neighbors. This method is good at navigating around obstacles in a fluid manner, but it may not always provide the shortest path due to the nature of the particle swarm's exploration. Then, as presented in the Figure 4, the FLA method presents a path that seems to be a compromise between the grid-based rigidity of A* and the smoothness of PSO. The path has distinct waypoints, similar to PSO, suggesting an iterative process of optimization. However, the path also appears to take into consideration the grid layout of the environment. The FLA is inspired by the foraging behavior of fruit flies; it starts with random searches and hones in on the path as the algorithm iterates, leading to a path that may not be the shortest but is found through a less computationally intensive process compared to A*.

Therefore, in order to do comparative analysis we consider four attributes such as efficiency, path quality, computational complexity and adaptability. Firstly, in terms of efficiency, A* is generally more efficient in grid-based environments, as it directly targets the shortest path. PSO and FLA may require more iterations to converge on a solution.

Secondly, in related to the Path Quality attribute, A* typically provides the shortest path. PSO and FLA tend to find paths that are less direct but may be more suitable for

scenarios where the movement pattern is not restricted to a grid. Then, for the computational complexity attribute, A* has a predictable computational complexity due to its heuristic, while PSO and FLA can vary greatly depending on the number of particles or agents and the complexity of the search space. Lastly, in terms of adaptability, PSO and FLA may adapt better to dynamic environments or when the search space is not discretized into a grid, as they do not rely on a predefined heuristic that is tailored to a grid layout. In conclusion, while A* is efficient and effective for grid-based pathfinding with a clear shortest path, PSO and FLA offer alternative approaches that may be more suitable for continuous spaces or applications where the computational simplicity and adaptability to dynamic changes are more critical than the absolute shortest path. into computational complexity and efficiency underscore the importance of algorithm selection based on specific application requirements, including resource availability and the need for real-time responsiveness. Future research should explore the integration of these algorithms with machine learning techniques to enhance their adaptability and efficiency. Moreover, investigating their performance in environments with dynamic obstacles could provide deeper insights into their practical applicability in real-world scenarios.

## 4. CONCLUSION

This study contributes significantly to the field of pathfinding in AI and robotics by comparing the efficacy of A*, PSO, and FLA algorithms within a grid-based simulation filled with static obstacles. By examining these algorithms, this research highlights the importance of selecting the right pathfinding strategy based on environmental constraints and application needs, advancing our understanding of algorithmic suitability in varied contexts. The A* algorithm has reaffirmed its value in scenarios requiring precise navigation and optimality, emphasizing its application in structured environments where the shortest path is paramount. This reinforces the algorithm's relevance in areas such as logistics and urban planning, where efficiency and predictability are key. The exploration of PSO has unveiled its potential in navigating complex environments with a degree of flexibility and adaptability not seen in more deterministic algorithms. This underlines its suitability for dynamic scenarios, such as autonomous vehicle navigation, where the environment can change unpredictably. Meanwhile, the investigation into FLA has revealed a promising approach that marries structure with adaptability, offering a viable pathfinding solution when resources are constrained or when a balance between speed and optimality is required. This could have implications for developing more efficient algorithms for mobile robotics in resource-limited settings. Importantly, this research not only delineates the strengths and applications of each algorithm but also sets the stage for future exploration. It invites further investigation into the integration of machine learning techniques with traditional pathfinding algorithms to enhance their adaptability and efficiency. This is particularly pertinent in the face of increasingly complex and dynamic environments encountered in real-world applications. Furthermore, the study suggests the necessity for more nuanced approaches to dynamic obstacle handling, a critical aspect of autonomous systems' navigation capabilities. In conclusion, this work advances the current state of knowledge by providing a comparative analysis that not only identifies the most suitable pathfinding algorithms for various applications but also encourages the development of more sophisticated, adaptive, and efficient navigation solutions. Future experiments should aim to extend these findings by exploring hybrid approaches that combine the strengths of deterministic and stochastic algorithms, thereby opening new avenues for research in AI and robotics.

## REFERENCES

Allus, A., Diab, A. M., & Bayraktar, E. (2024). Improving efficiency and cost of ordering algorithms in

pathfinding using shell layers. *Expert Systems with Applications*, *238*, 121948.
Bhatt, C., Sharma, R., Chauhan, R., Vishvakarma, A., Manchanda, M., & Sharma, S. (2023). Implementation and Visualization of Path Finding Algorithms. *2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA)*, 240–243.
Campos-Mac\'\ias, L., Aldana-López, R., de la Guardia, R., Parra-Vilchis, J. I., & Gómez-Gutiérrez, D. (2021). Autonomous navigation of MAVs in unknown cluttered environments. *Journal of Field Robotics*, *38*(2), 307–326.
Candra, A., Budiman, M. A., & Hartanto, K. (2020). Dijkstra's and a-star in finding the shortest path: A tutorial. *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)*, 28–32.
Fransen, K., & van Eekelen, J. (2023). Efficient path planning for automated guided vehicles using A*(Astar) algorithm incorporating turning costs in search heuristic. *International Journal of Production Research*, *61*(3), 707–725.
Goudarzi, A., Ghayoor, F., Waseem, M., Fahad, S., & Traore, I. (2022). A Survey on IoT-Enabled Smart Grids: Emerging, Applications, Challenges, and Outlook. *Energies*, *15*(19), 6984.
Grieshaber, S. (2020). Equity and research design. In *Doing early childhood research* (pp. 177–191). Routledge.
Honar Pajooh, H., Rashid, M. A., Alam, F., & Demidenko, S. (2022). Experimental performance analysis of a scalable distributed hyperledger fabric for a large-scale IoT testbed. *Sensors*, *22*(13), 4868.
Jiang, X., Lin, Z., He, T., Ma, X., Ma, S., & Li, S. (2020). Optimal path finding with beetle antennae search algorithm by using ant colony optimization initialization and different searching strategies. *IEEE Access*, *8*, 15459–15471.
Karur, K., Sharma, N., Dharmatti, C., & Siegel, J. E. (2021). A survey of path planning algorithms for mobile robots. *Vehicles*, *3*(3), 448–468.
Leon, J. F., Li, Y., Martin, X. A., Calvet, L., Panadero, J., & Juan, A. A. (2023). A Hybrid Simulation and Reinforcement Learning Algorithm for Enhancing Efficiency in Warehouse Operations. *Algorithms*, *16*(9), 408.
Li, J., & Yang, S. X. (2023). Intelligent escape of robotic systems: A survey of methodologies, applications, and challenges. *Journal of Intelligent \& Robotic Systems*, *109*(3), 55.
Li, P., Wang, X., Gao, H., Xu, X., Iqbal, M., & Dahal, K. (2021). A dynamic and scalable user-centric route planning algorithm based on polychromatic sets theory. *IEEE Transactions on Intelligent Transportation Systems*, *23*(3), 2762–2772.
Li, Y., Wei, W., Gao, Y., Wang, D., & Fan, Z. (2020). PQ-RRT*: An improved path planning algorithm for mobile robots. *Expert Systems with Applications*, *152*, 113425. https://doi.org/10.1016/j.eswa.2020.113425
Nouri, A., Lachheb, A., & El Amraoui, L. (2024). Optimizing efficiency of Vehicle-to-Grid system with intelligent management and ANN-PSO algorithm for battery electric vehicles. *Electric Power Systems Research*, *226*, 109936.
Pyke, L. M., & Stark, C. R. (2021). Dynamic pathfinding for a swarm intelligence based UAV control model using particle swarm optimisation. *Frontiers in Applied Mathematics and Statistics*, *7*, 744955.
Rahmani, V., & Pelechano, N. (2022). Towards a human-like approach to path finding. *Computers \& Graphics*, *102*, 164–174.
Sanchez-Ibanez, J. R., Perez-del-Pulgar, C. J., & Garc\'\ia-Cerezo, A. (2021). Path planning for autonomous mobile robots: A review. *Sensors*, *21*(23), 7898.
Semiz, F., & Polat, F. (2021). Incremental multi-agent path finding. *Future Generation Computer Systems*, *116*, 220–233.
Shahar, T., Shekhar, S., Atzmon, D., Saffidine, A., Juba, B., & Stern, R. (2021). Safe multi-agent pathfinding with time uncertainty. *Journal of Artificial Intelligence Research*, *70*, 923–954.
Shit, R. C. (2020). Precise localization for achieving next-generation autonomous navigation: State-of-the-art, taxonomy and future prospects. *Computer Communications*, *160*, 351–374.
Tan, C. S., Mohd-Mokhtar, R., & Arshad, M. R. (2021). A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access*, *9*, 119310–119342.
Tang, L., Han, Y., Zalhaf, A. S., Zhou, S., Yang, P., Wang, C., & Huang, T. (2024). Resilience enhancement of active distribution networks under extreme disaster scenarios: A comprehensive overview of fault

location strategies. *Renewable and Sustainable Energy Reviews*, *189*, 113898.
Telikani, A., Tahmassebi, A., Banzhaf, W., & Gandomi, A. H. (2021). Evolutionary machine learning: A survey. *ACM Computing Surveys (CSUR)*, *54*(8), 1–35.
Tezaur, R., As'ad, F., & Farhat, C. (2022). Robust and globally efficient reduction of parametric, highly nonlinear computational models and real time online performance. *Computer Methods in Applied Mechanics and Engineering*, *399*, 115392.
Van Thanh, N., & Linh, T. T. M. (2022). Real-time Trajectory Planning for Autonomous Vehicles in Dynamic Traffic Environments: A Survey of Modern Algorithms and Predictive Techniques. *Journal of Intelligent Connectivity and Emerging Technologies*, *7*(12), 1–25.
Vasconcelos, J. V. R., Brandão, A. S., & Sarcinelli-Filho, M. (2020). Real-time path planning for strategic missions. *Applied Sciences*, *10*(21), 7773.