Article

AI-Powered Software Testing for Heterogeneous Fleet Vehicle Routing Optimization with Time Constraints

Zhongbo Liu [1,*], Guillermo Palacios-Navarro [2] and Raquel Lacuesta [1]

[1] Department of Computer Science and Systems Engineering, University of Zaragoza, Teruel, 44003, Spain
[2] Department of Electronic Engineering and Communications, University of Zaragoza, Teruel, 44003, Spain
* Correspondence: Zhongbo Liu, Department of Computer Science and Systems Engineering, University of Zaragoza, Teruel, 44003, Spain

Abstract: The Vehicle Routing Problem (VRP) is crucial in logistics, transportation, and distribution. Traditional VRP focuses on optimizing vehicle routes between a fixed starting point and multiple locations to minimize travel distance or time. However, these models perform inadequately in dynamic environments such as campus student path planning, which involve diverse movement patterns and time window constraints. This paper addresses campus student path planning as a Heterogeneous Fleet Vehicle Routing Problem with Time Windows (HFVRPTW) and introduces the Scooter-Aware Pathfinding with Time Windows (SAPTW) method. Students start from random points and navigate a grid-based campus to fixed destinations like dormitories and cafeterias, choosing either walking or using electric scooters available at specific locations. This study tackles key challenges including diverse movement modes, time windows for reaching destinations, automatic generation of campus maps, and random generation of student starting points and destinations. Additionally, ensuring AI-powered software testing, we developed the Grid-based Campus Map Randomized Generation (GMRG) method, a rule-based approach for creating grid maps with roads, obstacles, and specific buildings. This method provides a realistic and controlled environment for route planning tests and simulations, ensuring the robustness and reliability of the proposed solution in real-world applications. Our approach highlights the potential of integrating artificial intelligence with software testing to optimize complex routing problems with time constraints. Simulation results demonstrate that SAPTW significantly enhances student arrival efficiency, reducing average arrival time by approximately 3% to 44% compared to traditional methods.

Keywords: campus path planning; dynamic environments; time windows; Vehicle Routing Problem

Received: 13 May 2025
Revised: 22 May 2025
Accepted: 06 June 2025
Published: 11 June 2025

1. Introduction

Vehicle Routing Problem (VRP) aim at obtaining the lowest cost, time or distance. For this purpose, many VRP variants and solution methods have been developed, such as Vehicle Routing Problem with Time Window (VRPTW) and Heterogeneous Fleet Vehicle Routing Problem (HFVRP) [1-4]. Adaptive large-neighborhood search and artificial intelligence have significantly improved VRP solutions [5,6]. There is also growing interest in combining VRP with self-driving cars and real-time data analytics [7,8]. Small-scale problems for VRP can be solved using branch-and-bound algorithms, dynamic programming algorithms, and integer linear programming algorithms, Large-scale problems can be optimized using genetic algorithms, simulated annealing, tabu search, and ant colony optimization, and AI can help optimize routes [9-17]. Migration learning and adaptive algorithms enable knowledge transfer between VRP scenarios and real-time model tuning [18-20].

Despite significant advances in AI in the field of VRP, the diversity of data sources affects the performance of algorithms, and the deployment of learning models and algorithm integration is more complex to effectively adapt to real-time changes and dynamic conditions in VRP.

For this reason, we use a combination of artificial intelligence and software testing to provide effective path optimization solutions for multiple motion patterns and time windows.

The main contributions of this paper are as follows:

Provides the SAPTW approach, a new vehicle routing model that combines multiple student movement patterns (walking and riding motorized vehicles) and time window constraints, which handles complex movement patterns and time constraints in a dynamic campus environment.

Provides a grid and rule-based Campus Map Random Generation (GMRG) methodology, which creates gridded maps containing roads, obstacles, and specific buildings.

Extensive simulation experiments demonstrated the effectiveness of our method. Results showed that route planning considering scooter pickup significantly improved efficiency, reducing average travel time by approximately 3% to 44%.

2. Related Work

Vehicle Routing Problems (VRP) face challenges in dynamic and uncertain environments. Recent research has explored various machine learning (ML) and deep reinforcement learning (DRL) methods for dynamic path planning. This section focuses on deep learning-based approaches and traditional optimization algorithms.

2.1. Deep Learning-Based Approaches

Deep Learning (DL) methods solve VRP problems by leveraging their ability to model complex relationships within data.

Reinforcement Learning (RL) is highly adaptable to dynamic environments by learning and improving from real-time feedback.

Hybrid methods that combine RL and DL have advantages when dealing with VRP problems in real time.

2.2. Traditional Optimization Algorithms

Traditional optimization algorithms are still crucial in VRPs. Okulewicz et al studied optimization algorithms for dynamic VRPs with real-time data integration [21]. Hutter et al proposed parametric algorithms for VRPs, however, traditional methods are not able to adapt to rapidly changing conditions in real-time [22].

Methods using artificial intelligence present significant advantages in terms of adaptability and performance in dynamic and stochastic environments. Shahbazian et al discussed deep learning methods for real-time VRP optimization [23].

3. Scooter-Aware Pathfinding with Time Windows Methodology

Our model is composed of the way students move on campus combined with the time window constraints of path planning, an abstraction that extends the traditional VRP model for dealing with complex movement patterns and time constraints. We address the problem of pathfinding in a grid-based environment where agents, such as students, can optionally acquire scooters to travel faster. The challenge involves optimizing the travel time considering the acquisition time of scooters and navigating obstacles.

3.1. Problem Formulation

The environment is represented as a grid of size N × N, where each cell can either be an obstacle, free space, or a scooter storage location. The agents start at a given position s and aim to reach a specific goal position g. Agents can either walk or ride a scooter, with

walking and riding speeds defined as v_w and v_s respectively. The objective is to find the shortest path considering both travel and scooter acquisition times.

3.2. Our Saptw Algorithm for Hfvrptw

To solve the heterogeneous fleet vehicle routing problem with time windows (HFVRPTW), We propose a scooter-aware pathfinding with time windows (SAPTW) algorithm that considers the time to acquire a scooter. The algorithm maintains a priority queue of states, each state represented as a tuple (f, p, h), where f is the estimated total cost from the start to the goal through the given state, p is the current position, and h is a boolean indicating whether the agent has a scooter.

3.3. Algorithm Detailed Steps

Algorithm 1 demonstrates the computation of SAPTW, and the following are its detailed steps:

Algorithm 1 Scooter-Aware Pathfinding with Time Windows Algorithm

---

Require: s (start), g (goal)
Ensure: Path from s to g
Initialize $O \leftarrow \{(0, s, \text{False})\}$ {Open set}
Initialize $C \leftarrow \emptyset$ {Came from}
Initialize $g_{score} \leftarrow \{(s, \text{False}) \mapsto 0\}$
Initialize $f_{score} \leftarrow \{(s, \text{False}) \mapsto d(s, g)\}$ {Estimated cost function}
while not $O = \emptyset$ **do**
  $(f, p, h) \leftarrow$ pop min(O) {State with lowest f}
  if p=g **then**
    return reconstruct path$(C, (p, h))$
  end if
  for each n in neighbors (p) **do**
    $h' \leftarrow h \vee$ is_scooter_storage(n) {Update scooter status}
    t_g $\leftarrow g_{score}[(p, h)] + 1$ {Tentative g_{score}}
    $v \leftarrow \begin{cases} v_s & \text{if } h' \\ v_w & \text{otherwise} \end{cases}$ {Travel speed}
    if h = False **then**
      $c \leftarrow \min_{s' \in S} \left(\frac{d(n,s')}{v_w} + \frac{d(s',g)}{v_s} \right)$ {Cost to nearest scooter storage}
    else
      $c \leftarrow \frac{d(n,g)}{v}$ {Direct cost to goal}
    end if
    if $(n, h') \notin g_{score} \vee$ t_g $< g_{score}[(n, h')]$ **Then**
      $g_{score}[(n, h')] \leftarrow$ t_g
      $f_{score}[(n, h')] \leftarrow g_{score}[(n, h')] + c$
      push$(O, (f_{score}[(n, h')], n, h'))$
      $C[(n, h')] \leftarrow (p, h)$
    end if
  end for
end while
return $\emptyset$

---

4. Grid-Based Campus Map Randomized Generation Methodology

Traditional methods are difficult to effectively address the problem of optimizing student paths in grid-based campus environments, especially when considering multiple movement patterns, obstacles, and the need to efficiently plan routes. this paper introduces the Grid-based Campus Map Randomized Generation (GMRG) method, to generate campus maps with various basic elements (e.g., destinations, obstacles, scooter storage areas, and student activity areas) and ensure that these elements do not overlap within buffer distances.

We start by defining the grid size and constants used throughout the generation process. These constants include the number of destinations (N_d), obstacles (N_o), scooter storage regions (N_S), and student generation areas (N_g), as well as proximity buffers and size constraints for the various regions. Algorithm 3 below demonstrates the computational process of GMRG.

To ensure no regions overlap, we use a function $\mathcal{F}_{overlapping}$ to check if a given region R overlaps with any regions in the set $\mathcal{R}$ within a specified buffer distance δ.Main input parameters include:

R: The region to be checked, represented as $(x_{min}, y_{min}, x_{max}, y_{max})$.

$\mathcal{R}$: A set of regions, Each element represented as $(x'_{min}, y'_{min}, x'_{max}, y'_{max})$.

Algorithm 2 demonstrates how $\mathcal{F}_{\text{overlapping}}$, works, and the following are its detailed steps:

Algorithm 2 Check Overlapping Region with Buffer

Require: $R = (x_{\min}, y_{\min}, x_{\max}, y_{\max})$, $\mathcal{R}$, δ
Ensure: True if R overlaps with any region in $\mathcal{R}$ within buffer δ, False otherwise
1: **for all** R′ in R **do**
2: $(x'_{min}, y'_{min}, x'_{max}, y'_{max}) \leftarrow R'$
3: **if** $(x_{min} - \delta < x'_{max} < x_{max})$ **and** $(y_{min} - \delta < y'_{max} < y_{max} + \delta)$ **then**
4: **return** True
5: **end if**
6:end for
7:return False

The overall algorithm for the GMRG algorithm (as shown in Algorithm 3) aims to generate a grid-based campus map comprising destinations (D), obstacles (O), scooter storage regions (S), student generation areas (G), and student positions (P). Initialization begins with empty sets for these elements.

Algorithm 3 GMRG Algorithm

Require: $\mathbf{G}$, $\mathbf{N_d}$, $\mathbf{N_o}$, $\mathbf{N_s}$, $\mathbf{N_g}$, $\mathbf{N_p}$, $\boldsymbol{\delta}$, $\mathbf{a}_{\min}$, $\mathbf{a}_{\max}$
Ensure: Map configuration
1: **I**nitialize $\mathcal{D}\leftarrow[]$,$\mathcal{O}\leftarrow[]$, $\mathcal{S}\leftarrow[]$, $\mathcal{G}\leftarrow[]$,$\mathcal{P}\leftarrow[]$
2: **while** $|\mathcal{D}| < N_d$ **do**
3: Generate (x,y)
4: **if** not $\mathcal{F}_{overlapping}$ ((x - δ, y - δ, x + δ, y + δ), $\mathcal{D}$) **then**
5: $\mathcal{D}\leftarrow\mathcal{D}\cup\{(x,y)\}$
6: **end if**
7: end while
8: **while** $|\mathcal{O}| < N_o$ **do**
9: Generate R
10: **if** not $\mathcal{F}_{overlapping}$ (R,$\mathcal{O}$) and not $\mathcal{F}_{overlapping}$ (R, $\mathcal{D}$) **then**

11: $\mathcal{O} \leftarrow \mathcal{O} \cup \{R\}$
12: **end if**
13: end while
14: **while** $|\mathcal{S}| < \mathbf{N_s}$ **do**
15: Generate R
16: **if** not $\mathcal{F}_{overlapping}$ (R, $\mathcal{S}$) and not $\mathcal{F}_{overlapping}$ (R, $\mathcal{O}$) and not $\mathcal{F}_{overlapping}$(R, $\mathcal{D}$) **then**
17: $\mathcal{S} \leftarrow \mathcal{S} \cup \{R\}$
18: **end if**
19: end while
20: **while** $|\mathcal{G}| < \mathrm{N_g}$ **do**
21: Generate R
22: **if** not $\mathcal{F}_{overlapping}$ (R, $\mathcal{G}$) and not $\mathcal{F}_{overlapping}$ (R, $\mathcal{O}$) and not $\mathcal{F}_{overlapping}$ (R, $\mathcal{D}$) and not $\mathcal{F}_{overlapping}$ (R, $\mathcal{S}$) then
23: $\mathcal{G} \leftarrow \mathcal{G} \cup \{R\}$
24: end if
25: end while
26: **while** $|\mathcal{P}| < \mathrm{N_p}$ **do**
27: Generate (x,y)
28: **if** not $\mathcal{F}_{\mathrm{obstacle}}$((x, y), $\mathcal{O}$) **then**
29: $\mathcal{P} \leftarrow \mathcal{P} \cup \{(x,y)\}$
30: **end if**
31: end while
32: **S**ave $\mathcal{D}$, $\mathcal{O}$, $\mathcal{S}$, $\mathcal{G}$, $\mathcal{P}$ to map configuration

---

5. Experiments

This paper comprehensively evaluates the path planning performance of each algorithm in complex campus environments through experiments and evaluation metrics, and verifies the effectiveness of the proposed method.

5.1. Baselines

In the context of Vehicle Routing Problem (VRP), several baseline methods include: Uniform Cost Search (UCS), Depth-First Search (DFS), Breadth-First Search (BFS), Dijkstra's Algorithm, Bidirectional Search, and A* Algorithm.

5.2. Overall Experiments

In the overall experiments (see below Table 1), our primary goal is to validate the effectiveness and performance of the proposed method across different scenarios, especially considering diverse modes of transportation and the impact of time windows on path planning efficiency.

Table 1. Performance Comparison of Different Path Planning Methods on Various Maps.

Map Parameters	Method	Total Path Distance (meters)	Average Arrival Time (seconds)	Average Speed meter/second	Percentage of Students with Scooter(%)
Grid Size: 500 Obstacles: 15	UCS	217.4385	295.171	1.09	18.0
	Dijkstra	225.0817	283.2081	1.26	20.0
	Bidirectional A*	242.1616	293.3622	1.09	18.0

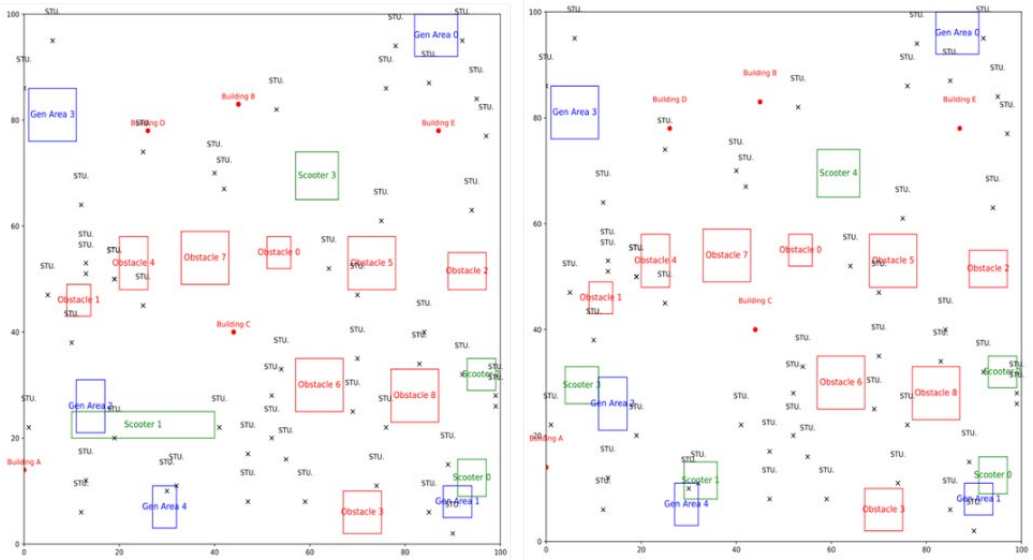
Scooter Storage: 5 Generation Areas: 10 Grid Size: 100 Obstacles: 5 Scooter Storage: 5 Generation Areas: 5 Grid Size: 50 Obstacles: 5 Scooter Storage: 5 Generation Areas: 5	SAPTW (Ours)	308.2508	274.7574	1.64	42.0
	UCS	101.6957	184.1253	0.72	16.0
	Dijkstra	109.0261	198.8643	0.76	10.0
	Bidirectional A*	114.8945	201.382	0.72	16.0
	SAPTW (Ours)	138.6231	112.7899	1.53	100.0
	UCS	57.6759	92.1349	1.09	26.0
	Dijkstra	61.7015	79.9816	1.32	38.0
	Bidirectional A*	57.7847	90.049	1.13	32.0
	SAPTW (Ours)	67.6651	44.132	2.01	98.0

The experimental setup consists of simulating a 500 × 500 grid campus environment with roads, obstacles, and specific buildings (e.g., dormitories, classrooms, scooter storage). By configuring multiple student groups of different sizes (e.g., 50, 100, 200) and randomly assigning start and end points to cover different campus areas. The grid size, obstacles, storage areas, Generation Areas, and Number of students can be dynamically varied, and the experiment reflects the complexity of the real world.

Regarding transportation modes, students can either walk or use electric scooters, each with predefined speeds (1.5"m/s" for walking, 6"m/s" for scooters). To increase complexity, students must arrive at their destinations within specified time windows, e.g., with an upper limit of 30 minutes. Evaluation metrics include average arrival times, path lengths, and scooter utilization, providing comprehensive insights into the method's performance under diverse transportation modes and time window restrictions. We additionally present the optimization results of different methods for 5 destinations when the grid size is 500. The results from Tables 1 show that our method SAPTW excels in speed, with the highest efficiency, making it ideal for rapid pathfinding.

5.3. Visualization of Generated Maps

In this paper, we have developed a rule-based method for automatic map generation and conducted a large number of simulations. The results validate the effectiveness of our algorithm in handling HFVRPTW, where the average arrival time of students is reduced by 3% to 44% during the simulations in Figure 1. With the use of scooters, the average time is reduced by 10.04%, and these visualizations support practical applications in terms of related optimization and overall efficiency while demonstrating the optimal path.



(a)Map before optimizing scooter storage areas. (b)Map after optimizing scooter storage areas.

Figure 1. Comparison of Campus Maps before and after Optimizing Scooter Storage Areas.

6. Conclusion and Future Work

6.1. Summary

This paper introduces a novel solution for optimizing paths in campus environments, considering multiple movement modes and time windows, particularly focusing on campus student path planning within the framework of the Heterogeneous Fleet Vehicle Routing Problem with Time Windows (HFVRPTW). The SAPTW method is proposed to optimize routes for students navigating grid-based campuses, allowing them to choose between walking and using electric scooters available at designated locations. The study demonstrates that integrating scooter availability significantly improves path efficiency, reducing average student travel times by approximately 3% to 44% compared to conventional methods.

Moreover, the paper underscores the potential of combining artificial intelligence with software testing to tackle complex routing problems under time constraints. It highlights the relevance of adaptive and real-time path-planning solutions in modern campus logistics and transportation management. Given the increasing complexity and diversity of student needs on campuses, traditional static path planning approaches often fall short. The SAPTW method offers a scalable and robust alternative capable of addressing the dynamic and heterogeneous nature of campus transportation systems.

6.2. Future Work

Future work should focus on the following aspects: Firstly, developing more robust algorithms to effectively handle diverse and sparse data is crucial.

Additionally, improving the adaptability of these models to dynamic and real-time changing environments is essential.

Furthermore, a comprehensive integration of these advanced algorithms into existing systems, along with thorough testing and validation, is imperative to ensure their stability and reliability.

Finally, exploring interdisciplinary approaches that combine insights from software engineering and artificial intelligence can provide innovative solutions.

References

1. G. D. Konstantakopoulos, S. P. Gayialis and E. P. Kechagias, "Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification," *Oper. Res. Int. J.*, vol. 22, pp. 2033–2062, 2022, doi: 10.1007/s12351-020-00600-7.

2. G. Laporte, "Fifty years of vehicle routing," *Transp. Sci.*, vol. 43, no. 4, pp. 408–416, 2009, doi: http://doi.org/10.1287/trsc.1090.0301.
3. M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987, doi: 10.1287/opre.35.2.254.
4. B. L. Golden, A. A. Assad, L. Levy and F. Gheysens, "The fleet size and mix vehicle routing problem," *Comput. Oper. Res.*, vol. 11, no. 1, pp. 49–66, 1984, doi: 10.1016/0305-0548(84)90007-8.
5. M. Nazari, A. Oroojlooy, L. V. Snyder and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 9839–9849, 2018.
6. V. Ghilas, E. Demir and T. Van Woensel, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines," *Comput. Oper. Res.*, vol. 72, pp. 12–30, 2016, doi: 10.1016/j.cor.2016.01.018.
7. L. Abbatecola, M. P. Fanti and W. Ukovich, "A review of new approaches for Dynamic Vehicle Routing Problem," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Fort Worth, TX, USA, 2016, pp. 361–366, doi: 10.1109/COASE.2016.7743429.
8. Q. Lu, T. Tettamanti, D. Hörcher and I. Varga, "The impact of autonomous vehicles on urban traffic network capacity: an experimental analysis by microscopic traffic simulation," *Transp. Lett.*, vol. 12, no. 8, pp. 540–549, 2019, doi: 10.1080/19427867.2019.1662561.
9. P. Toth and D. Vigo, "Branch-and-bound algorithms for the capacitated VRP," in *The Vehicle Routing Problem*, Philadelphia, PA, USA: SIAM, 2002, pp. 29–51, doi: 10.1137/1.9780898718515.ch2.
10. R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *J. ACM*, vol. 9, no. 1, pp. 61–63, 1956, doi: 10.1145/321105.321111.
11. J.-F. Cordeau and G. Laporte, "A branch-and-cut algorithm for the dial-a-ride problem," *Oper. Res.*, vol. 51, no. 5, pp. 831–844, 2002, doi: 10.1287/opre.1060.0283.
12. J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Cambridge, MA, USA: MIT Press, 1992. ISBN: 9780262581110.
13. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983, doi: 10.1126/science.220.4598.671.
14. F. Glover, "Tabu search—Part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989, doi: 10.1287/ijoc.1.3.190.
15. M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern., Part B (Cybern.)*, vol. 26, no. 1, pp. 29–41, Feb. 1996, doi: 10.1109/3477.484436.
16. H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina and L. Song, "Learning combinatorial optimization algorithms over graphs," *arXiv preprint* arXiv:1704.01665, 2017, doi: 10.48550/arXiv.1704.01665.
17. W. Kool, H. van Hoof and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint* arXiv:1803.08475, 2018, doi: 10.48550/arXiv.1803.08475.
18. I. Bello, H. Pham, Q. V. Le, M. Norouzi and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint* arXiv:1611.09940, 2016, doi: 10.48550/arXiv.1611.09940.
19. H. Mao, M. Alizadeh, I. Menache and S. Kandula, "Resource management with deep reinforcement learning," *arXiv preprint* arXiv:1605.06676, 2016, doi: 10.1145/3005745.3005750.
20. S. Panigrahi, A. Nanda and T. Swarnkar, "A survey on transfer learning," in *Intelligent and Cloud Computing*, D. Mishra, R. Buyya, P. Mohapatra and S. Patnaik, Eds., Singapore: Springer, 2021, vol. 194, *Smart Innov. Syst. Technol.*, pp. 871–883, doi: 10.1007/978-981-15-5971-6_83.
21. M. Okulewicz and J. Mańdziuk, "Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem," in *Artificial Intelligence and Soft Computing. ICAISC 2013*, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh, and J.M. Zurada, Eds., vol. 7895, *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2013, pp. 423–434, doi: 10.1007/978-3-642-38610-7_50.
22. F. Hutter, H. H. Hoos, K. Leyton-Brown and T. Stützle, "ParamILS: An automatic algorithm configuration framework," *J. Artif. Intell. Res.*, vol. 36, pp. 267–306, 2009, doi: 10.1613/jair.2861.
23. R. Shahbazian, L. D. P. Pugliese, F. Guerriero and G. Macrina, "Integrating machine learning into vehicle routing problem: Methods and applications," *IEEE Access*, vol. 12, pp. 93087–93115, 2024, doi: 10.1109/ACCESS.2024.3422479.