International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(7s) 2026 pp. 559-574
https://doi.org/10.70917/ijcisim-2026-3116
Article

Data-Driven Optimization of Garbage Collection Point Locations for Efficient Urban Solid Waste Management

U Balakrishna[1], Suresha R[2*], Jayanth Kumar T[2], A Prakash[3], Thupili Rama Mohan Reddy[4] and Thangaraj M[5]

[1]Department of Mathematics, SITAMS, Chittoor - 517127 Andhra Pradesh, India; prasantibalu@gmail.com
[*2]Department of Data Analytics and Mathematical Sciences, Faculty of Engineering and Technology, JAIN (Deemed-to-be University), Ramanagara-562112, Karnataka, India; suresharamareddy@gmail.com; jayanth.maths@gmail.com
[3]Department of Mathematics, Vemu Institute of Technology, P kothakota, Chittoor -517112, Andhra Pradesh, India; alikapati.prakash@gmail.com
[4]Department of Basic Sciences, Narayana Engineering College, Nellore-524003, Andhra Pradesh, India; rammohanreddy803@gmail.com
[5]Department of Mathematics, School of Advanced Sciences, Vellore Institute of Technology, Chennai 600127, Tamil Nadu, India; mthangaraj51@gmail.com
Corresponding Author: Suresha R (suresharamareddy@gmail.com)

Abstract: Rapid urbanization and growing population density have made municipal solid waste management increasingly challenging, particularly in designing efficient and cost-effective garbage collection systems. Among the key planning decisions, the placement of Garbage Collection Points (GCPs) plays a vital role, as it influences the walking distance for residents, vehicle travel distance, fuel consumption, operational expenses, and the overall cleanliness of urban areas. However, while considerable attention has been given to optimizing waste collection routes, relatively little research has focused on determining the optimal locations of GCPs in real urban environments. To address this issue, this study formulates the GCP placement problem as a Minimum Vertex Cover (MVC) problem on an urban road network. This formulation identifies the minimum number of collection points required to provide complete coverage of all demand locations while maintaining service accessibility. To solve the problem effectively, a Hybrid Genetic Algorithm (HGA) is proposed, combining the global search capability of genetic algorithms with problem-specific local improvement techniques to enhance solution quality and convergence. The effectiveness of the proposed approach is evaluated through computational experiments on benchmark datasets and further demonstrated using a real-world case study from the Yadgir district. The results indicate that the proposed method consistently achieves complete coverage using fewer collection points, reduces vehicle travel requirements, and produces higher-quality solutions than conventional greedy and heuristic methods. These findings demonstrate that the proposed framework can serve as a practical decision-support tool for municipal authorities by enabling scalable, data-driven, and economically efficient planning of urban waste collection infrastructure. Since the approach relies on readily available geographic information, it can be adapted for waste management planning in medium-sized cities across different regions.
Keywords: Garbage Collection Points; Solid Waste Management; Minimum Vertex Cover; Urban Road Networks; Genetic Algorithm; Hybrid Genetic Algorithm; Decision Support Systems; Sustainable Urban Planning

1. Introduction

The rapid growth of cities in recent decades has made municipal solid waste (MSW) management one of the most pressing urban challenges. As urban areas expand and populations rise, the volume of waste generated increases proportionally, leading to pollution of land, air, and water and creating serious risks to public health (Aleluia and Ferrão, 2016). Historically,

many regions relied on simple and unsustainable disposal methods such as open dumping or uncontrolled burning, which provided little protection for the environment (Hamer, 2023). Over time, stricter regulations and technological advances have encouraged the adoption of improved practices such as source separation, recycling, composting, and energy recovery. Despite these advancements, one persistent challenge remains: the efficient collection and transportation of waste before it causes harm to people or the environment. These two activities—collection and transport, often account for the largest share of waste management expenses, sometimes exceeding half of the total operational budget (Guerrini et al., 2017). For cities with constrained budgets, particularly in developing countries, these high costs limit the resources available for other essential public services. Improving the efficiency of collection systems is therefore not merely a matter of convenience but an economic and public health necessity, as it reduces costs, minimizes environmental impacts, and helps mitigate health hazards.

As a result, research in waste management has increasingly focused on improving the efficiency of waste collection systems, with much of the work centered on optimizing vehicle routes. Route planning studies aim to minimize total travel distance, reduce fuel consumption, and shorten service times by generating efficient paths for collection vehicles (Beliën et al., 2014; Das et al., 2019). However, most of these studies operate under the assumption that the locations of garbage collection points (GCPs)—the designated sites where waste is stored prior to collection—are already optimally placed. Poorly located GCPs can significantly reduce the benefits of optimized routing, resulting in longer travel distances, unbalanced workloads, delayed services, and, in some cases, overflowing or unattended waste. Such inefficiencies are particularly pronounced in densely populated or irregularly planned cities (Chattopadhyay et al., 2009; Doğan and Süleyman, 2021). These issues highlight that optimizing GCP placement is just as crucial as optimizing collection routes, yet this problem has been relatively underexplored in both research and practical applications.

In recent years, graph-theoretic methods have gained prominence for modeling and improving urban waste collection systems. One particularly effective approach is the Minimum Vertex Cover (MVC) problem, which identifies the smallest set of nodes required to cover all connections within a network (Ganian, 2017). Applied to urban road networks, MVC can determine the minimal number of GCPs necessary to ensure that every street segment is served, thereby preventing redundant or inefficient placements. Extensions of this framework, such as weighted vertex cover and twin cover models, allow the incorporation of real-world factors including road size, population density, and service frequency (Ganian, 2018; Jovanovic and Tuba, 2013). Alongside these theoretical models, advances in metaheuristic optimization have demonstrated the effectiveness of algorithms such as Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Variable Neighbourhood Search (VNS) in solving large-scale, complex problems (Banharnsakun, 2023). These methods have been applied successfully to facility location, vehicle routing, and scheduling, often producing near-optimal solutions within practical constraints (Papalitsas and Andronikos, 2007; Bautista and Pereira, 2007). In the context of waste management, however, the application of such techniques has largely focused on routing and allocation problems, with less emphasis on dynamic, mathematically minimal placement of GCPs guided by MVC principles.

Waste management challenges extend beyond logistics, reflecting broader social, economic, and operational dimensions. The quantity and composition of waste are influenced by consumption patterns, technological trends, and governance systems (Brown, 2015; Berthier, 2003). Operationally, inefficiencies in routing, limited fleet capacity, occupational health risks, and low community engagement continue to pose significant obstacles (Akhtar et al., 2015; Singh et al., 2014; Jaunich et al., 2016). Various computational approaches, including integer programming (Das et al., 2019), graph-based scheduling (Saukenova et al., 2022), and GIS-supported allocation models (Erfani et al., 2023), have enhanced planning and optimization. Nonetheless, the use of MVC-based models for determining the optimal placement of GCPs remains underexplored, leaving substantial opportunities for methodological innovation.

Empirical studies have demonstrated that reducing the number of GCPs while maintaining full coverage can yield significant cost savings and environmental benefits. For instance, (Gallego et al., 2022) employed mixed-integer programming to optimize bin placement, achieving reductions in both setup and operational costs. Similarly, (Cárdenas-León et al., 2024) utilized GIS-based clustering and optimization to minimize the number of collection points without compromising service accessibility. Multi-criteria decision-making frameworks have also been applied to integrate economic, environmental, and operational considerations when planning GCP locations (Goulart et al., 2021; Alshraideh and Al-Jbori, 2022). However, most of these approaches treat the problem as a location-allocation issue rather than a network coverage problem, limiting their ability to achieve truly minimal solutions as defined by MVC principles.

Despite advancements in routing optimization and facility location models, there remains a lack of computational frameworks that combine MVC formulations with adaptive optimization techniques to identify the minimal set of GCP locations. Existing methods often rely on static placement strategies, do not guarantee minimality, and lack flexibility to adapt to evolving urban network structures. Addressing this gap, the present study proposes a Hybrid Genetic Algorithm (HGA) for optimal GCP allocation within an MVC framework. In this approach, the urban road network is modeled as a graph, with vertices representing potential GCP sites and edges corresponding to street segments requiring service. The HGA leverages the global search capabilities of GA alongside a problem-specific local optimization phase, enabling the identification of minimal vertex covers that ensure complete coverage without redundancy. This hybrid approach mitigates premature convergence, dynamically adapts to network variations, and efficiently produces high-quality solutions, offering a practical and scalable tool for sustainable urban waste management.

The proposed approach is evaluated through a case study in Yadgir district, Karnataka, India. Results demonstrate significant reductions in both the number of GCPs, and total collection distance compared to conventional allocation models. By directly linking graph-theoretic optimization with real-world waste management objectives, this study provides a scalable, algorithm-driven decision-support framework that promotes cost reduction, environmental sustainability, and operational efficiency—particularly in rapidly urbanizing and resource-constrained regions.

2. Problem Statement and Mathematical Model

The MVC Problem is a well-established optimization problem in graph theory, particularly useful in urban infrastructure planning. In the context of waste management, it can be applied to determine the optimal placement of GCPs. Given an undirected graph G = (V , E), where V, the node set represents intersections/potential GCP locations and E, the edge set represents roads connecting these points, the objective is to identify the smallest possible subset of GCPs $C \subseteq V$ such that every edge in the graph is "covered"—meaning at least one of its endpoints is a selected GCP. This ensures that all road segments are serviced by at least one nearby collection point. The goal is to minimize the total number of GCPs required while maintaining complete coverage of the urban road network, ultimately improving operational efficiency, reducing costs, and enhancing environmental and public health outcomes.

The following notations are used in the model:

Notations		Description
V	:	Set of intersections or potential GCP locations (vertices)
$E \subseteq V \times V$	:	Set of roads connecting GCP locations (edges)
$C \subseteq V$	:	A subset of vertices

$$x_i \quad : \quad \begin{cases} 1, \text{ if a garbage collection point} \\ \quad \text{is placed at intersection} \\ 0, \quad \text{Otherwise} \end{cases}$$

The zero-one integer linear programming formulation is given as follows

$$Minimize\ Z = \sum_{i \in V} x_i \tag{1}$$

Subjected to

$$x_i + x_j \geq 1 \ \ \forall \ (i, j) \in E \tag{2}$$

$$x_i \in \{0,1\} \ \forall i \in V \tag{3}$$

Eq. 1 represents the objective function that minimizes the total number of GCPs. The constraint [2] guarantees that for every road connecting two intersections, at least one endpoint has a GCP. Finally, a binary decision variable x_i takes 1 if a GCP is placed at location i, and takes 0 if no GCP is placed at the location i.

3. Proposed Algorithm

This section presents a Hybrid Genetic Algorithm (HGA) developed to generate high-quality, near-optimal solutions for the placement of garbage collection points (GCPs). The performance of the HGA is evaluated against three alternative methods: the Harmony Search Algorithm (HSA), the Greedy Approximation Algorithm (GAA), and the standard Genetic Algorithm (GA).

3.1 Harmony Search Algorithm (HSA)

The Harmony Search Algorithm is a population-based metaheuristic inspired by the improvisation process of musicians. In the proposed graph-based model, each node represents a potential GCP, and edges correspond to street segments requiring service. The objective of the algorithm is to identify the smallest subset of nodes such that every edge is incident to at least one selected GCP, thereby achieving complete network coverage with minimal infrastructure.

Each candidate solution is encoded as a binary vector, where a value of '1' at position i indicates that vertex i is included in the cover. The algorithm begins by generating an initial Harmony Memory (HM), consisting of a randomly created set of these vectors. To ensure feasibility, each solution is processed using a repair function that examines all edges and, if an edge is uncovered, adds one of its endpoints—preferably the vertex with the higher degree—to the cover. The quality of a solution is evaluated based on the total number of selected vertices, with lower values representing superior solutions. During each iteration, a new solution is constructed by either selecting values from the existing HM with a probability defined by the Harmony Memory Considering Rate (HMCR) or by assigning random values. Selected components may be further adjusted (i.e., flipping their bit values) according to the Pitch Adjusting Rate (PAR). This mechanism maintains a balance between exploiting known high-quality solutions and exploring new possibilities, which is essential for the algorithm's efficiency. Once a new solution is generated, it is repaired to ensure coverage and then evaluated. If it performs better than the worst solution in the HM, it replaces that solution. The iterative process continues until the algorithm reaches a predefined maximum number of improvisation iterations. For this study, the Harmony Memory Size (HMS) was set to 10, the HMCR to 0.9, and the PAR to 0.3. The algorithm was executed for up to 100 iterations (max_iter) to ensure convergence. The complete procedure is formally summarized in Algorithm 1.

Algorithm 01. Harmony Search Algorithm for the Vertex Cover Problem

Input: Graph G = (V, E), Harmony Memory Size HMS,
Harmony Memory Considering Rate HMCR,
Pitch Adjusting Rate PAR, Maximum Iterations max_iter

1. Initialize Harmony Memory (HM) with HMS random solutions
   - For each solution:
     a. Generate random binary vector of length |V|
     b. Apply repair function to ensure all edges are covered
     c. Store the solution and its fitness in HM
2. Repeat for max_iter iterations:
   a. Initialize new solution vector
   b. For each position i in the solution:
      - With probability HMCR:
        • Select a value from HM at position i
        • With probability PAR:
          Flip the selected bit (0 to 1 or 1 to 0)
      - Else:
        • Assign random bit (0 or 1)
   c. Apply repair function to new solution
   d. Evaluate fitness (i.e., size of the vertex cover)
   e. If better than the worst in HM, replace it
3. Return the best solution from HM

This algorithmic framework is particularly well-suited for sparse urban graphs, where the topology allows efficient convergence toward compact and feasible vertex covers. By adapting its search strategy through harmonically inspired principles, the HSA demonstrates a strong capability to generate near-optimal solutions for complex spatial problems such as GCP placement. Its integration with geospatial data from sources like OpenStreetMap allows practical deployment in urban planning scenarios, helping reduce infrastructural costs and improve waste management logistics while maintaining service coverage across the entire road network.

3.2 Greedy Approximation Algorithm (GAA)

In this study, the GAA serves as a foundational heuristic approach for addressing the VCP within the context of MSW management. The vertex cover formulation models the urban road network as an undirected graph, where each node corresponds to a GCPs and each edge denotes a road segment requiring coverage. Given the NP-hard nature of the problem, obtaining exact solutions is computationally infeasible for large graphs. Therefore, an efficient greedy strategy is employed to generate near-optimal vertex covers with minimal computational overhead. The core logic of the algorithm is based on iteratively selecting vertices that contribute the most to edge coverage. In each step of the algorithm, the vertex with the highest degree—meaning the one connected to the largest number of uncovered edges—is selected and added to the cover set. Choosing this vertex helps reduce the maximum number of edges at once. After a vertex is included, all its incident edges are removed from the graph, and the process is repeated until no edges remain uncovered. This degree-based rule speeds up

convergence and keeps the solution feasible at every stage. While it does not always yield the exact optimal cover, the method consistently produces small and valid solutions, making it well-suited for cases where quick results and limited computational effort are important. The procedure is outlined in Algorithm 2 below:

Algorithm 02. Greedy Approximation for Vertex Cover Using Degree Heuristic

Input: An undirected graph G = (V, E)

Output: An approximate vertex cover C

1. Initialize C ← ∅
2. Let U ← set of all edges in E
3. While U is not empty do:
   a. Initialize a dictionary D to count degrees
   b. For each edge (u, v) in U:
      i. D[u] ← D[u] + 1
      ii. D[v] ← D[v] + 1
   c. Select vertex x with maximum D[x]
   d. Add x to C
   e. Remove all edges from U that are incident to x
4. Return C

This greedy strategy proves particularly effective in practical contexts where speed and simplicity are critical, such as the rapid establishment of infrastructure in newly developed urban areas or during emergency planning scenarios. Although it does not guarantee the identification of the absolute minimal vertex cover, the approach reliably generates compact and feasible solutions with a low computational cost. Consequently, it offers a practical and scalable option for municipal authorities aiming to implement efficient and cost-effective waste management systems.

3.3 Genetic Algorithm (GA)

To address the limitations of deterministic heuristics in solving the Vertex Cover Problem, a GA is employed as a metaheuristic search strategy. Rooted in the principles of natural evolution, the GA provides a robust mechanism for navigating large and complex solution spaces. In the context of this study, each potential solution is encoded as a binary string of length equal to the number of vertices in the graph. A value of '1' at the ith position in the string indicates the inclusion of the corresponding vertex, i.e., GCP in the vertex cover, while a '0' signifies exclusion. The algorithm begins by generating an initial population of such binary-encoded individuals randomly. To ensure feasibility from the outset, a repair procedure is applied to everyone, adding a vertex (typically the one with a higher degree) whenever an uncovered edge is identified. This guarantees that each candidate solution satisfies the constraint of covering all edges. Fitness evaluation is a critical component of the GA, wherein everyone is assessed based on the number of vertices selected in the cover. To discourage invalid solutions, a penalty is added to the fitness score for each uncovered edge, thereby guiding the search toward feasible and compact covers. Parents are selected for reproduction using tournament selection, which favors fitter individuals while maintaining diversity. Offspring are generated through crossover—typically uniform or single-point—followed by bitwise mutation, where each bit in the offspring has a small probability of flipping. Since mutation can introduce infeasible solutions, a post-mutation repair phase is applied to restore coverage. Elitism is incorporated by carrying over the best-performing individual from the current generation to the next, ensuring that the quality of solutions does not degrade. The process is repeated over a predetermined number of generations, after which the best solution

encountered is returned as the approximate vertex cover. The GA with a population of 30 and 50 generations was applied. Uniform crossover, bit-flip mutation (0.1 rate), and tournament selection (size 3) guided evolution, while elitism preserved the best solution. A repair operator and a penalty-based fitness function ensured feasibility and minimized the vertex cover size. The algorithm follows the logic outlined in Algorithm 3 below:

Algorithm 03. Genetic Algorithm for Approximating Vertex Cover

Input: Graph G = (V, E), population size P, mutation rate μ, number of generations G

Output: Approximate vertex cover

1.Randomly initialize a population of P binary vectors, each representing a candidate vertex cover.

2.For everyone in the population, apply a repair function to ensure all edges in E are covered.

3.Repeat the following steps for a fixed number of generations:

a. Evaluate the fitness of everyone as:

fitness = |cover| + penalty × (number of uncovered edges)

b. Select parents using tournament selection.

c. Apply crossover (e.g., uniform or single-point) to produce offspring.

d. Mutate each offspring by flipping each bit with probability μ.

e. Repair mutated offspring to ensure edge coverage.

f. Form the new population by selecting the best P individuals from the combined parent and offspring pools, preserving the best solution found so far (elitism).

4.After the final generation, return the best valid individual as the approximate vertex cover.

The GA's capacity to balance exploration and exploitation makes it particularly suitable for medium-scale urban networks, such as city wards or industrial sectors, where a compromise between computational cost and solution quality is essential. Its iterative, adaptive nature allows it to uncover high-quality vertex covers that deterministic methods may overlook, thereby supporting more efficient waste management planning.

3.4 Hybrid Genetic Algorithm (HGA)

To overcome the limitations of standard heuristics and enhance solution quality for the Vertex Cover Problem, this study proposes a HGA that combines the global search capabilities of traditional GAs with problem-specific heuristics, such as greedy initialization, local search optimization, and adaptive mutation control. The goal remains to identify a minimum subset of vertices i.e. GCPs such that every edge in the undirected graph (representing roads) is incident to at least one selected vertex. Owing to the NP-hard nature of the problem, exact algorithms are computationally infeasible for large graphs, thereby justifying the need for a hybrid metaheuristic approach. The HGA encodes each solution as a binary string of length equal to the number of vertices, where a '1' indicates inclusion of a vertex in the cover. Unlike conventional GAs that begin with entirely random populations, the HGA initializes half of the population using a greedy heuristic, which iteratively selects the highest-degree vertex until all edges are covered. This strategy injects high-quality individuals into the initial population, accelerating convergence. The remaining individuals are generated randomly and subsequently repaired to ensure edge coverage by adding one endpoint of any uncovered edge—typically the one with higher degree. Once initialized, all individuals undergo a local search procedure designed to remove redundant vertices while preserving the validity of the cover. This step significantly reduces cover size without compromising feasibility. Fitness evaluation is based primarily on the number of selected vertices, with penalties imposed for any uncovered edges.

The evolutionary cycle begins by selecting parents through tournament selection, followed by crossover and bitwise mutation. Mutation rates are dynamically adjusted: increased when no improvement is observed over consecutive generations to encourage exploration and decreased when better solutions are found to favour exploitation. After mutation, each offspring is repaired and optimized through local search to ensure both feasibility and compactness. An elitism mechanism retains the best solution from the current generation, ensuring that progress is not lost. This integrated framework offers significant advantages over other algorithms studied. Compared to the Greedy approach, HGA yields smaller vertex covers by escaping local optima. It outperforms the GA using greedy seeding and local refinement, which the standard GA lacks. When compared with the HSA, HGA demonstrates superior adaptability and convergence speed due to its dynamic mutation and combined exploitation-exploration strategy. Experimental results confirm that HGA consistently produces high-quality, feasible solutions across both sparse and dense graph instances, making it the most effective method evaluated in this study. In the proposed HGA, the population size was set to 50 and the algorithm was executed for 50 generations. The initial population was generated using a combination of greedy-based and random solutions to maintain a balance between solution quality and diversity. Tournament selection with a pool size of three was employed for parent selection, and uniform crossover was applied to generate offspring. An elitism strategy preserved the best individual in each generation. The mutation rate was initialized at 0.1 and adaptively adjusted within the range [0.01, 0.3] based on population improvement to prevent premature convergence. The complete procedure is outlined in Algorithm 4 as follows:

Algorithm 04. Hybrid Genetic Algorithm for the Vertex Cover Problem

Input: Undirected graph G(V, E), population size P, number of generations G_max

Output: Approximate minimum vertex cover

1. Initialize population:
   - Half individuals using greedy heuristic
   - Half individuals using random bitstrings
   - For everyone:
     → Repair to ensure vertex cover
     → Apply local search to minimize solution size
2. Evaluate fitness of all individuals:
   - Fitness = cover size + (penalty × number of uncovered edges)
3. Identify the best individual (elitism)
4. For generation = 1 to G_max do:
   a. Initialize new population with the best individual (elitism)
   b. While new population size < P:
      i. Select two parents via tournament selection
      ii. Perform crossover to create two offspring
      iii. Apply mutation to each offspring
      iv. Repair each offspring to ensure feasibility
      v. Apply local search on each offspring
      vi. Add both offspring to new population
   c. Replace old population with new population

d. Evaluate fitness of all individuals

→ Update best solution if a better one is found

e. Adapt mutation rate:

→ If best solution hasn't improved: increase mutation rate

→ Else: decrease mutation rate

5. Return the best individual as the approximate minimum vertex cover

In summary, the proposed HGA effectively combines the strengths of constructive heuristics and evolutionary search. By integrating greedy initialization, adaptive mutation, and local refinement, it consistently produces superior solutions with efficient use of computational resources. Its robustness across varying network scales and densities establishes it as a powerful tool for urban solid waste management and similar large-scale optimization problems.

4. Comparative Study

To comprehensively assess the effectiveness of the proposed Hybrid Genetic Algorithm (HGA), a comparative analysis was conducted using ten benchmark test cases sourced from existing literature. The performance of the HGA was evaluated against several state-of-the-art approaches, including the Harmony Search Algorithm (HSA), the Greedy Approximation Algorithm (GAA), the standard Genetic Algorithm (GA), and other heuristic techniques, as summarized in Table 1. The computational results demonstrate that, across all ten test instances, the proposed HGA consistently produced solutions that were either superior to or at least on par with those obtained by the benchmark methods. This consistent performance highlights the algorithm's effectiveness in leveraging the global search capabilities of GAs alongside the local refinement strengths of heuristic operators. By integrating these complementary strategies, the HGA effectively mitigates the risk of premature convergence and improves its ability to escape local optima, thereby enhancing both robustness and solution quality. Furthermore, the algorithm's adaptability across diverse problem settings indicates its practical potential for designing efficient and scalable garbage collection networks in real-world urban environments. Overall, these findings confirm that the HGA provides more reliable, high-quality solutions and greater flexibility compared to the baseline heuristics examined in this study.

To further evaluate the performance of the proposed HGA, Table 2 presents a comparison of four vertex cover algorithms—HSA, GAA, GA, and the proposed HGA—on 31 real-world garbage collection networks from Karnataka, available at: (https://tinyurl.com/5a6r9ax7). In these networks, nodes represent GCP's, and edges represent the connecting roads. The goal was to find the smallest set of GCPs that covers all edges. The results show that HGA consistently gave the best solutions, especially in medium and large networks with more than 200 edges, where it clearly outperformed the other methods. The GA was the fastest, often finishing in milliseconds, but usually produced the worst solutions. GA gave a good balance, with smaller covers than GAA and reasonable runtimes. HSA performed the weakest, being both slower and less accurate as the network size grew. The strength of HGA comes from combining a wide search with local improvements, which helped it explore better solutions, though at the cost of longer runtimes. This means HGA is best suited for offline planning where accuracy matters most, while the Greedy algorithm is better when speed is critical. Overall, the study shows that the choice of algorithm should depend on the situation, balancing solution quality with computation time.

Further, to statistically evaluate the performance of the four algorithms, the Friedman test was applied to both the Best-Found Solutions (BFS) and CPU Runtime across identical problem instances. The test revealed significant performance differences for both BFS ($\chi^2 = 87.1020$, $p < 0.05$) and Runtime ($\chi^2 = 93.0000$, $p < 0.05$). Following this, Wilcoxon signed-rank tests were conducted to compare the proposed HGA with HSA, GAA, and GA. The results showed that HGA significantly outperforms the other algorithms in terms of solution quality

(all $p < 0.05$). However, this improved solution quality comes at the cost of increased computational time, with HGA exhibiting significantly higher runtimes compared to the alternatives (all $p < 0.05$). These findings indicate that while HGA is more effective in finding high-quality solutions, it requires more computational effort to do so.

Table 1. Comparative performance of MVC solution methods

Instance	|N|	|E|	AVCA	AA	ApA	BKS	HAS	GAA	GA	HGA
Fig.1 (Thenepalle and Singamsetty, 2018)	7	8	6	3	4	-	3	4	3	3
Fig.2 (Thenepalle and Singamsetty, 2018)	8	9	-	-	-	-	4	4	4	4
Fig.3 (Thenepalle and Singamsetty, 2018)	11	19	-	-	-	-	7	7	7	7
Fig.4 (Thenepalle and Singamsetty, 2018)	9	8	7	5	5	-	5	5	4	4
Fig.5 (Thenepalle and Singamsetty,2018)	17	16	14	8	9	-	8	8	8	7
Fig.6 (Chen et al., 2016)	25	46	-	-	-	15	17	15	16	15
Fig.7 (Zhuang and Chen, 2019)	4	5	-	-	-	2	2	2	2	2
Fig.8 (Wang et al., 2019)	5	5	-	-	-	3	3	3	3	3
Fig.9 (Khattab et al., 2019)	6	6	-	-	-	2	2	2	2	2
Fig.10 (Hochbaum, et al., 1997)	6	10	-	-	-	4	4	4	4	4

|N| = Nodes, |E| = Edges, AVCA = Approximation Vertex-Cover Algorithm, AA = Alom's Algorithm, ApA = Approximation Algorithm, BKS = Best Known Solution, GAA = Greedy Approximation Algorithm, GA = Genetic Algorithm, HAS = Harmony Search Algorithm, HGA = Proposed Hybrid Genetic Algorithm.

Table 2. Comparative results of HSA, GAA, GA, and HGA applied to spatial garbage collection networks

Test Case	Nodes /GCP's	Edges /Roads	Best Found Solution (BFS)							
			HSA		GAA		GA		HGA	
			BFS	CPU Runtime (In Sec.)	BFS	CPU Runtime (In Sec.)	BFS	CPU Runtime (In Sec.)	BFS	CPU Runtime (In Sec.)
1	7	6	3	0.0106	3	0.00016	3	0.0262	3	0.1938
2	42	47	25	0.0443	21	0.00051	21	0.1165	21	3.2216

3	55	69	36	0.0583	29	0.00081	31	0.1671	27	1.8429
4	57	72	40	0.0589	29	0.00075	32	0.3232	27	2.5361
5	58	69	38	0.0586	29	0.00093	33	0.1811	29	2.4969
6	59	72	38	0.0637	33	0.00053	32	0.1618	28	2.9806
7	65	83	45	0.0818	34	0.00063	35	0.2072	31	2.4913
8	95	118	64	0.1012	49	0.0018	55	0.3084	47	6.1783
9	104	122	72	0.1042	49	0.00154	57	0.3301	49	6.772
10	112	134	77	0.1143	57	0.00271	65	0.3879	54	6.7777
11	139	174	96	0.1431	70	0.00214	81	0.73	66	12.2593
12	154	181	101	0.2822	77	0.0027	89	0.554	73	14.1248
13	157	164	103	0.2707	73	0.00493	86	0.8431	68	12.3474
14	163	223	120	0.3028	87	0.00352	107	0.7142	83	17.4522
15	290	372	214	0.3989	151	0.00863	188	1.8036	143	49.2972
16	293	379	211	0.3001	151	0.00904	186	1.9532	143	52.6519
17	298	427	227	0.3345	163	0.01092	202	1.8531	157	56.2002
18	305	425	233	0.3924	169	0.0112	209	3.4622	160	62.2041
19	320	442	245	0.6193	173	0.01177	218	2.0613	166	62.9341
20	356	517	276	0.4097	200	0.01849	249	2.576	190	89.4777
21	363	512	273	0.4181	202	0.0164	248	3.2243	191	86.7039
22	392	468	285	0.3807	196	0.02507	254	2.3535	187	82.1368
23	475	606	355	0.4726	243	0.02241	319	4.7101	229	129.831
24	489	623	365	0.5011	248	0.02878	325	3.611	237	131.722
25	491	649	368	0.5256	254	0.04495	334	4.1749	247	147.797
26	617	765	461	0.891	318	0.03759	413	6.2253	301	217.808
27	651	908	496	0.7094	358	0.04805	467	8.1844	347	273.235
28	778	1044	598	1.5733	414	0.06366	533	10.685	398	403.533
29	917	1317	722	1.0719	510	0.099	664	14.3729	495	561.51
30	950	1248	733	1.0004	499	0.16694	663	15.2026	472	533.485
31	953	1212	725	1.002	494	0.1531	658	12.9801	479	519.84

Table 3. Geospatial Coordinates of Garbage Collection points (GCP's) Network in Yadgir District

GCP	Latitude	Longitude	GCP	Latitude	Longitude	GCP	Latitude	Longitude	GCP	Latitude	Longitude
0	16.7466	77.1377	51	16.7496	77.133	101	16.7444	77.1326	151	16.7462	77.139
1	16.7467	77.1368	52	16.7499	77.1327	102	16.7444	77.1317	152	16.747	77.139
2	16.7512	77.1374	53	16.7506	77.1334	103	16.7447	77.1328	153	16.7479	77.1367
3	16.7515	77.1371	54	16.7507	77.137	104	16.7447	77.1337	154	16.7482	77.1363
4	16.751	77.1372	55	16.7494	77.1355	105	16.7453	77.1337	155	16.7514	77.1322
5	16.7516	77.1378	56	16.7498	77.1342	106	16.7453	77.1328	156	16.7526	77.1326
6	16.7471	77.1377	57	16.7492	77.1386	107	16.7453	77.1318	157	16.7507	77.1349
7	16.7471	77.1368	58	16.7488	77.1386	108	16.7449	77.132	158	16.7516	77.1332
8	16.747	77.1386	59	16.7492	77.138	109	16.7461	77.1321	159	16.7517	77.1334
9	16.7485	77.1378	60	16.7528	77.138	110	16.746	77.1338	160	16.7506	77.1339
10	16.7468	77.1349	61	16.753	77.1383	111	16.7459	77.131	161	16.7506	77.1337

11	16.746	77.1376	62	16.7532	77.1378	112	16.7466	77.1292	162	16.7505	77.1295
12	16.7536	77.1308	63	16.7535	77.1317	113	16.7468	77.1307	163	16.7498	77.13
13	16.7535	77.131	64	16.753	77.1315	114	16.7467	77.1317	164	16.7519	77.13
14	16.7525	77.1313	65	16.7535	77.1373	115	16.7471	77.1311	165	16.752	77.1296
15	16.7519	77.1312	66	16.7504	77.1361	116	16.7488	77.1334	166	16.7523	77.1303
16	16.7526	77.132	67	16.7502	77.1353	117	16.7472	77.1302	167	16.7521	77.1282
17	16.7525	77.1311	68	16.7484	77.1365	118	16.7472	77.131	168	16.752	77.1283
18	16.7531	77.1291	69	16.7498	77.1349	119	16.7482	77.1301	169	16.7524	77.1292
19	16.753	77.1295	70	16.75	77.1344	120	16.7481	77.1329	170	16.7526	77.1289
20	16.7533	77.1291	71	16.7501	77.1355	121	16.7477	77.1295	171	16.7492	77.1379
21	16.7528	77.129	72	16.7488	77.1378	122	16.7487	77.1308	172	16.7531	77.1275
22	16.7524	77.1286	73	16.749	77.1378	123	16.7475	77.1294	173	16.7531	77.1274
23	16.7478	77.1356	74	16.7483	77.1365	124	16.747	77.1288	174	16.7493	77.1304
24	16.7478	77.1351	75	16.751	77.131	125	16.749	77.1268	175	16.7523	77.1361
25	16.7523	77.1309	76	16.7513	77.1311	126	16.7472	77.1298	176	16.7526	77.1358
26	16.7527	77.1313	77	16.7506	77.131	127	16.7514	77.1305	177	16.7531	77.135
27	16.752	77.1374	78	16.7515	77.1301	128	16.7497	77.1301	178	16.7491	77.1379
28	16.751	77.1377	79	16.7515	77.1302	129	16.7494	77.1295	179	16.7501	77.1344
29	16.7468	77.1386	80	16.7518	77.1304	130	16.7498	77.1299	180	16.7498	77.1342
30	16.7448	77.1366	81	16.7513	77.1289	131	16.749	77.1287	181	16.7488	77.1334
31	16.7454	77.1381	82	16.7446	77.1388	132	16.7492	77.1303	182	16.7502	77.1357
32	16.7488	77.1334	83	16.747	77.1387	133	16.75	77.1322	183	16.7486	77.1308
33	16.7528	77.1329	84	16.7514	77.1308	134	16.75	77.1318	184	16.7487	77.1308
34	16.7515	77.1319	85	16.7424	77.1336	135	16.7514	77.1319	185	16.7488	77.1308
35	16.7516	77.1315	86	16.7426	77.1331	136	16.7532	77.1309	186	16.7488	77.1309
36	16.7492	77.1328	87	16.7438	77.1329	137	16.7535	77.1297	187	16.7519	77.1283
37	16.7492	77.1328	88	16.7432	77.1325	138	16.7489	77.128	188	16.7488	77.1333
38	16.7474	77.1314	89	16.7437	77.1318	139	16.7491	77.1283	189	16.7488	77.1334
39	16.7487	77.1309	90	16.7437	77.1325	140	16.7491	77.1278	190	16.7488	77.1334
40	16.7506	77.1318	91	16.7437	77.1337	141	16.7493	77.127	191	16.7489	77.1334
41	16.7509	77.1313	92	16.7441	77.1311	142	16.7493	77.1281	192	16.7489	77.1334
42	16.7511	77.1319	93	16.7438	77.1327	143	16.7495	77.1279	193	16.7531	77.1273
43	16.7502	77.1324	94	16.744	77.1326	144	16.7502	77.128	194	16.7532	77.1274
44	16.7521	77.1307	95	16.7445	77.1328	145	16.7499	77.1276	195	16.75	77.1298
45	16.7525	77.1305	96	16.744	77.1321	146	16.7501	77.1273	196	16.7499	77.1289
46	16.7533	77.1296	97	16.7442	77.1326	147	16.7507	77.1276	197	16.7499	77.1294
47	16.7517	77.1373	98	16.7448	77.1313	148	16.751	77.1279	198	16.7521	77.1262
48	16.7522	77.1362	99	16.7454	77.1306	149	16.7534	77.1288	199	16.7469	77.1301
49	16.7477	77.1334	100	16.7442	77.1321	150	16.7535	77.1285	200	16.7478	77.131
50	16.7444	77.1348									

Table 4. Edge Set Connecting GCP's

(0, 1), (1, 10), (1, 11), (2, 3), (2, 4), (2, 5), (3, 47), (3, 66), (4, 54), (4, 28), (5, 27), (5, 28), (6, 7), (6, 8), (6, 9), (7, 23), (7, 153), (8, 58), (8, 29), (9, 72), (9, 153), (10, 24), (10, 49), (10, 50), (11, 29), (11, 30), (11, 31), (12, 13), (13, 63), (13, 136), (14, 15), (14, 16), (14, 17), (15, 25), (15, 76), (16, 33), (16, 34), (16, 35), (17, 25), (17, 26), (18, 19), (18, 20), (18, 21), (18, 22), (19, 46), (19, 45), (20, 149), (21, 166), (21, 170), (22, 167), (22, 170), (23, 24), (23, 154), (24, 32), (25, 44), (25, 45), (26, 46), (26, 64), (27, 47), (27, 60), (28, 171), (29, 82), (29, 83), (32, 189), (32, 190), (33, 156), (33, 157), (34, 135), (34, 35), (35, 41), (36, 37), (36, 38), (36, 39), (36, 188), (37, 51), (37, 191), (37, 186), (38, 115), (38, 200), (39, 183), (39, 186), (40, 41), (40, 42), (40, 43), (41, 75), (42, 43), (42, 135), (43, 52), (43, 133), (44, 80), (44, 127), (45, 166), (46, 137), (47, 48), (48, 65), (48, 175), (48, 70), (49, 109), (49, 120), (51, 52), (51, 53), (52, 133), (53, 158), (55, 56), (56, 70), (56, 180), (56, 181), (57, 58), (57, 59), (58, 72), (59, 178), (60, 61), (60, 62), (61, 62), (62, 65), (63, 64), (64, 136), (66, 67), (66, 68), (67, 69), (67, 71), (68, 74), (68, 189), (68, 171), (69, 70), (69, 71), (70, 179), (71, 182), (72, 73), (73, 74), (73, 178), (74, 154), (75, 76), (75, 77), (76, 84), (77, 134), (78, 79), (78, 80), (78, 81), (79, 162), (79, 127), (80, 164), (81, 162), (81, 168), (83, 152), (84, 127), (84, 128), (85, 86), (86, 87), (86, 88), (87, 91), (87, 93), (87, 95), (88, 89), (88, 90), (89, 92), (89, 93), (92, 98), (92, 99), (93, 94), (94, 96), (94, 97), (95, 101), (95, 103), (97, 100), (97, 101), (98, 103), (98, 107), (99, 111), (99, 112), (101, 102), (103, 106), (104, 105), (105, 110), (105, 106), (106, 109), (107, 108), (107, 109), (111, 113), (111, 114), (112, 124), (112, 126), (113, 117), (113, 118), (114, 115), (114, 116), (114, 120), (115, 118), (116, 188), (116, 190), (118, 119), (119, 121), (119, 122), (119, 183), (120, 190), (121, 123), (122, 184), (122, 183), (123, 124), (123, 126), (124, 125), (125, 141), (125, 198), (126, 199), (128, 174), (128, 163), (129, 130), (129, 131), (129, 132), (130, 163), (130, 132), (130, 195), (132, 184), (132, 174), (133, 134), (135, 155), (136, 137), (138, 139), (138, 140), (138, 141), (139, 142), (139, 144), (140, 142), (140, 143), (141, 145), (141, 146), (142, 143), (143, 145), (144, 145), (144, 147), (145, 146), (146,

147), (147, 148), (148, 187), (148, 197), (149, 150), (151, 152), (153, 154), (155, 156), (157, 175), (157, 179), (158, 159), (159, 160), (159, 161), (162, 163), (164, 165), (164, 166), (167, 168), (167, 172), (168, 187), (169, 170), (171, 178), (172, 173), (172, 194), (173, 193), (173, 187), (174, 185), (175, 176), (176, 177), (179, 180), (180, 192), (181, 192), (181, 189), (184, 185), (185, 186), (187, 195), (188, 191), (191, 192), (193, 194), (195, 197), (196, 197)

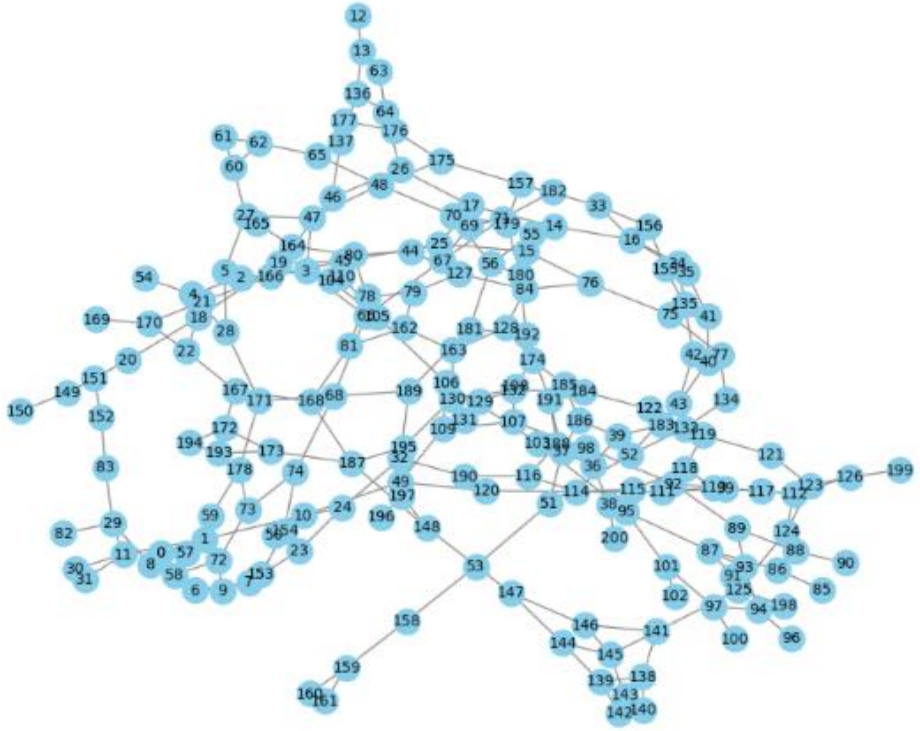

Figure 1. Graphical Network of GCPs.

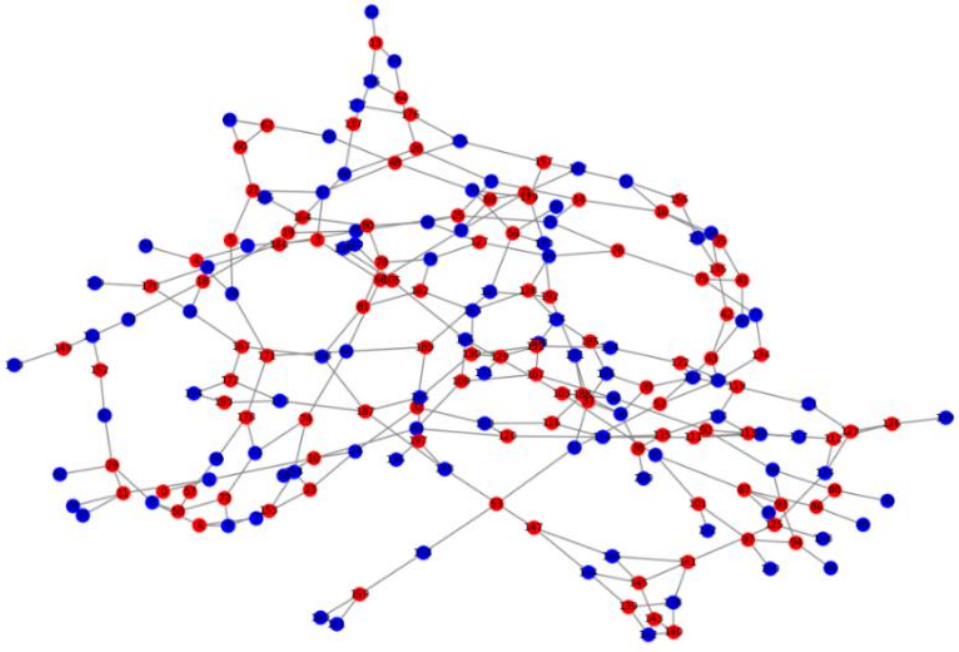

Figure 2. Network with Optimal GCP's (Marked in Red).

Table 5. Summary of best-found solutions using HAS, GAA, GA and HGA

|N|	|E|	Best Found Solution							
		HSA		GAA		GA		HGA	
		Vertex cover size	CPU Runtime (In Sec.)	Vertex cover size	CPU Runtime (In Sec.)	Vertex cover size	CPU Runtime (In Sec.)	Vertex cover size	CPU Runtime (In Sec.)
201	271	149	0.2275	109	0.00457	133	3.3345	104	25.4678

5. Optimization of Garbage Collection Points: A Yadgir Case Study

This case study is based on a geospatial city network in Yadgir District, Karnataka, covering a 1,000-meter radius around 16.7446°N and 77.1297°E, with 201 GCPs represented as nodes and 271 road segments represented as edges, forming a realistic collection network. Tables 3 and 4 list the GCP coordinates and road connectivity details, while Figures 1 and 2 show the complete network and the optimized GCP layout. The goal was to find the smallest set of GCPs that could cover all roads, helping reduce duplication and operating costs. Four algorithms were tested: HSA, GAA, GA, and the proposed HGA. As shown in Table 5, HGA gave the best result, selecting only 104 GCPs—fewer than GAA (109), GA (133), and HSA (149)—though it took longer to run. HGA's strength is that it combines broad search with local improvements, allowing it to avoid poor solutions and give a balanced design. These results show how combining spatial data with advanced algorithms can make urban infrastructure planning more effective and scalable. By applying an HGA to the vertex cover problem, the approach offers a scalable and adaptable framework for efficient infrastructure planning. As cities face increasing pressure from rapid growth and limited resources, such data-driven methods present a sustainable path forward for informed and effective municipal decision-making.

6. Conclusions

This study presents a comprehensive comparison of heuristic and metaheuristic approaches for optimizing garbage collection networks using the Minimum Vertex Cover framework, with particular emphasis on geospatial applications in Karnataka. Four optimization methods—HSA, GAA, GA, and the proposed Hybrid Genetic Algorithm (HGA)—were evaluated using both benchmark datasets and real-world road networks. Among the methods considered, HGA consistently identified solutions requiring the fewest Garbage Collection Points (GCPs) while ensuring complete network coverage. The superiority of the proposed approach was further validated through Friedman and Wilcoxon statistical tests, which confirmed that the improvements achieved by HGA were statistically significant in terms of both solution quality and computational efficiency. The real-world case study conducted in the Yadgir district also demonstrated the practical effectiveness of the proposed algorithm, where it generated more efficient GCP layouts than the competing methods. The findings suggest that integrating evolutionary search with problem-specific local improvement strategies offers a robust approach for solving complex urban infrastructure optimization problems. By reducing the number of required collection points without compromising service coverage, the proposed framework has the potential to lower operational costs and improve the efficiency of municipal waste collection systems. Overall, HGA provides a practical, scalable, and data-driven decision-support tool that can assist urban planners and municipal authorities in developing more sustainable solid waste management systems, particularly in rapidly growing cities and developing regions.

Acknowledgements
All investigators have reviewed and approved the manuscript, fulfilling authorship standards, and certify that the work is truthful.

Data Availability

All supporting data are included in the manuscript, and the test cases are available on the associated GitHub repository.

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Competing Interests

The authors declare that they have no competing interests.

References:

1. Akhtar, M., Hannan, M. A., Basri, H., & Scavino, E.(2015): Solid waste generation and collection efficiencies: Issues and challenges. Jurnal Teknologi, 75(11). https://doi.org/10.11113/jt.v75.5331
2. Aleluia, J., & Ferrão, P.(2016): Characterization of urban waste management practices in developing Asian countries: A new analytical framework based on waste characteristics and urban dimension. Waste Management, 58, 301–318. https://doi.org/10.1016/j.wasman.2016.05.008
3. Alshraideh, H. A., & Al-Jbori, A. M.(2022): Integrated multi-criteria decision-making model for sustainable location of waste collection nodes considering population density, waste generation rates, road connectivity, and service frequency. Journal of Environmental Management and Planning, 8(2), 112–128.
4. Banharnsakun, A. (2023). A new approach for solving the minimum vertex cover problem using artificial bee colony algorithm. Decision Analytics Journal, 6, 100175.
5. Bautista, J., & Pereira, J.(2007): Ant algorithms for a time and space constrained assembly line balancing problem. European journal of operational research, 177(3), 2016-2032.
6. Beliën, J., De Boeck, L., & Van Ackere, J.(2014): Municipal solid waste collection and management problems: A literature review. Transportation Science, 49(1), 1–17. https://doi.org/10.1287/trsc.2013.0475
7. Berthier, H. C.(2003): Garbage, work and society. Resources, Conservation and Recycling, 39(3), 193-210. https://doi.org/10.1016/S0921-3449(03)00027-2
8. Brown, D. P.(2015): Garbage: How population, landmass, and development interact with culture in the production of waste. Resources, Conservation and Recycling, 98, 41-54. https://doi.org/10.1016/j.resconrec.2015.02.012
9. Cárdenas-León, I., Koeva, M., Nourian, P., & Davey, C. (2024): Urban digital twin-based solution using geospatial information for solid waste management. Sustainable Cities and Society, 115, 105798. https://doi.org/10.1016/j.scs.2024.105798
10. Chattopadhyay, S., Dutta, A., & Ray, S.(2009): Municipal solid waste management in Kolkata, India. Waste Management, 29(1), 470–478. https://doi.org/10.1016/j.wasman.2008.01.023
11. Chen, J., Kou, L., & Cui, X.(2016): An approximation algorithm for the minimum vertex cover problem. Procedia engineering, 137, 180-185. https://doi.org/10.1016/j.proeng.2016.01.248
12. Das, S., Bhattacharya, B. K., & Bhattacharya, A. (2019): An efficient approach for waste collection route optimization using GIS. International Journal of Environment and Waste Management, 23(1), 56–76. https://doi.org/10.1504/IJEWM.2019.097276
13. Das, S., Lee, S. H., Kumar, P., Kim, K. H., Lee, S. S., & Bhattacharya, S. S.(2019): Solid waste management: Scope and the challenge of sustainability. Journal of cleaner production, 228, 658-678. https://doi.org/10.1016/j.jclepro.2019.04.323
14. Doğan, B., & Süleyman, K. (2021): Optimization of urban solid waste collection using spatial analysis and GIS. Sustainability, 13(4), 1893. https://doi.org/10.3390/su13041893
15. Erfani, T., Diaz, F., & Ahmad, A (2023): Optimizing solid waste collection using integer linear programming and GIS: A case study. Algorithms, 16(4), 174. https://doi.org/10.3390/a16040174.
16. Gallego, L. A., López-Lezama, J. M., & Carmona, O. G. (2022): A mixed-integer linear programming model for simultaneous optimal reconfiguration and optimal placement of capacitor banks in distribution networks. IEEe Access, 10, 52655-52673. doi: 10.1109/ACCESS.2022.3175189
17. Ganian, R. (2017): Twin-cover: Beyond vertex cover in parameterized complexity. Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17), 3917–3923,10.1007/978-3-642-28050-4_21
18. Ganian, R.(2018): Advances in structural parameterizations for vertex cover variants. Discrete Applied Mathematics, 234, 34–43. https://doi.org/10.1016/j.dam.2017.08.015

19. Goulart, M. P., Silva, F. C., & Costa, R. S.(2021): Multi-criteria optimization of garbage collection point placement based on socioeconomic, spatial, and operational parameters. European Journal of Operational Research, 292(3), 947–960.
20. Guerrini, A., Carvalho, P., Romano, G., Marques, R. C., & Leardini, C.(2017): Assessing efficiency drivers in municipal solid waste collection services through a non-parametric method. Journal of cleaner production, 147, 431-441. https://doi.org/10.1016/j.jclepro.2017.01.079
21. Hamer, G. (2003): Solid waste treatment and disposal: effects on public health and environmental safety. Biotechnology advances, 22(1-2), 71-79. https://doi.org/10.1016/j.biotechadv.2003.08.007.
22. Hochbaum, D. S.(1997): Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. Approximation algorithms for NP-hard problems, 94-143.
23. Jaunich, M. K., Levis, J. W., DeCarolis, J. F., Gaston, E. V., Barlaz, M. A., Bartelt-Hunt, S. L., ... & Jaikumar, R.(2016): Characterization of municipal solid waste collection operations. Resources, Conservation and Recycling, 114, 92-102. https://doi.org/10.1016/j.resconrec.2016.07.012
24. Jovanovic, R., & Tuba, M.(2013): An ant colony optimization algorithm for the minimum vertex cover problem. Computing and Informatics, 32(2), 225–245.
25. Khattab, H., Sharieh, A., & Mahafzah, B. A. (2019). Most valuable player algorithm for solving minimum vertex cover problem. International Journal of Advanced Computer Science and Applications, 10(8).
26. Papalitsas, C., & Andronikos, T.(2023): A generalized variable neighborhood search algorithm for the TSP with time windows in waste collection. Technologies, 11(3), 61. https://doi.org/10.3390/technologies11030061.
27. Saukenova, I., Oliskevych, M., Taranic, I., Toktamyssova, A., Aliakbarkyzy, D., & Pelo, R.(2022): Optimization of schedules for early garbage collection and disposal in the megapolis. Eastern-European Journal of Enterprise Technologies, 1(3), 115 .https://doi.org/10.15587/1729-4061.2022.251082
28. Singh, G. K., Gupta, K., & Chaudhary, S.(2014): Solid waste management: its sources, collection, transportation and recycling. International Journal of Environmental Science and Development, 5(4), 347.DOI: 10.7763/IJESD.2014.V5.507
29. Thenepalle, J. K., & Singamsetty, P. (2018): An articulation point-based approximation algorithm for minimum vertex cover problem. In Advances in Algebra and Analysis: International Conference on Advances in Mathematical Sciences, Vellore, India, December 2017-Volume I (pp. 281-289). Springer International Publishing.
30. Wang, L., Hu, S., Li, M., & Zhou, J.(2019): An exact algorithm for minimum vertex cover problem. Mathematics, 7(7), 603.
31. Zhuang, S., & Chen, D. (2019): A novel algorithm for the vertex cover problem based on minimal elements of discernibility matrix. International Journal of Machine Learning and Cybernetics, 10, 3467-3474. https://doi.org/10.1007/s13042-019-00933-6