Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, No. 6, April : 2024

# MODELING FRAMEWORK FOR DEFERRED DIAL-A-RIDE PROBLEM WITH VEHICLE FENCING

**Dr. Ankita Karale,** Department of Computer Engineering Sandip Institute of Technology and Research Centre Nashik, India ankita.karale@sitrc.org
**Om Dahale,** Department of Computer Engineering Sandip Institute of Technology and Research Centre Nashik, India om.g.dahale@gmail.com
**Sanket Shivale,** Department of Computer Engineering Sandip Institute of Technology and Research Centre Nashik, India shivalesanket21@gmail.com
**Neeraj Adhav,** Department of Computer Engineering Sandip Institute of Technology and Research Centre Nashik, India adhavneeraja9500@gmail.com
**Prajakta Gavali,** Department of Computer Engineering Sandip Institute of Technology and Research Centre Nashik, India prajaktagavali6630@gmail.com

**Abstract**
This paper provides a framework for dial-a-ride applications, particularly deferred dial-a-ride applications. The paper formulates VRP with semi-soft time windows i.e the vehicle is allowed to wait if it arrives early but should strictly not cross the late limit, vehicle capacity constraints and also allows missing pickups for a penalty. The vehicles can have fencing constraints that restricts the vehicles to serve in only their selected area. This fencing is done by pre-computing the pickup and delivery locations that fall in the area for every vehicle, using raycasting method but you can use any other method as well such as winding number algorithm. Central to this framework is the utilization of raycasting techniques for fencing, enabling precise delineation of vehicle territories within irregular polygons. Raycasting, a computational method used in computer graphics and geometry, facilitates the identification of boundary intersections, thereby defining the boundaries within which vehicles can operate. This approach ensures accurate determination of the operational areas for vehicles, optimizing routing solutions to navigate efficiently within the specified territories and ensuring effective utilization of resources while enhancing overall transportation efficiency.

**Keywords**:
vehicle routing problem (VRP), hybrid genetic search (HGS), optimize routes, transportation, raycasting, fencing

## I. Introduction

The significance of this research extends beyond mere algorithmic refinement; it offers a pivotal advancement in the realm of transportation logistics. By addressing the complexities of the Vehicle Routing Problem (VRP) within irregular polygons, the framework caters to a critical need in modern transportation management. Its application spans a wide array of sectors, from urban transit systems to rural service providers, encompassing school buses, medical transport, and beyond.
This framework's ability to integrate various constraints such as semi-soft time windows, vehicle capacities, and penalty considerations for missed pickups reflects its adaptability to diverse operational contexts. Such versatility not only enhances operational efficiency but also translates into tangible benefits for both service providers and end-users. It promises to optimize resource utilization, reduce operational costs, and improve service reliability, thus significantly impacting the economic and social dimensions of transportation logistics. Furthermore, by fostering environmental sustainability through more efficient routing and reduced emissions, the framework aligns with broader societal goals of mitigating environmental impact. In essence, its significance lies in its capacity to revolutionize transportation logistics management, offering innovative solutions to complex operational challenges while fostering sustainability and enhancing overall quality of service.

## II. Problem Formulation:

The problem formulation comprises two distinct stages: initially, a preprocessing phase is undertaken to ascertain whether a given location falls within the operational range of a vehicle, achieved through raycasting techniques. Subsequently, the actual formulation of the Vehicle Routing Problem (VRP) is conducted based on the outcomes of the preprocessing step :-

**1. Preprocessing Step:** We will first find if pickup or drop off location is in range using Ray Casting. The outline of the algorithm follows like this:-
1. Start with a point *P* that you want to test for being inside the polygon.
2. Draw a horizontal ray from *P* towards the right, extending to infinity.
3. Count the number of times the ray intersects with the edges of the polygon.
4. If the number of intersections is odd, then the point is inside the polygon. If the number of intersections is even, then the point is outside the polygon.

```
def point_inside_polygon(point, polygon):
    num_intersections = 0
    for i in range(len(polygon)):
        p1 = polygon[i]
        p2 = polygon[(i + 1) % len(polygon)]  # Wrap around to the first point if we've reached the end
        if point[1] > min(p1[1], p2[1]) and point[1] <= max(p1[1], p2[1]):
            if point[0] <= max(p1[0], p2[0]):
                if p1[1] != p2[1]:
                    x_intersection = (point[1] - p1[1]) * (p2[0] - p1[0]) / (p2[1] - p1[1]) + p1[0]
                    if p1[0] == p2[0] or point[0] <= x_intersection:
                        num_intersections += 1
    return num_intersections % 2 == 1  # True if odd number of intersections
# Example usage:
polygon = [(1, 1), (1, 3), (3, 3), (3, 1)]
point = (2, 2)
print(point_inside_polygon(point, polygon))  # Output: True
```

**Vertical Range Check:** We ensure that the given point's y-coordinate falls within the vertical range defined by the current edge of the polygon. This ensures that the point is potentially intersecting with the edge vertically.
**Horizontal Range Check:** Further filtering is applied to consider only those edges where the given point's x-coordinate lies within the horizontal range defined by the current edge. This ensures that the point is potentially intersecting with the edge horizontally.
**Non-Horizontal Edge Check:** To avoid division by zero in subsequent calculations, we verify that the current edge is not horizontal.
**Intersection Calculation:** Using the equation of a line, we calculate the x-coordinate of the intersection point between the edge and a horizontal line passing through the given point. This helps determine if the given point intersects the current edge.
**Left-Side Check:** We check if the given point lies to the left of the intersection point. If it does, it indicates an intersection with the edge, ensuring that we count valid intersections.
These conditions collectively help determine whether the given point lies inside or outside the polygon, based on the number of intersections it makes with the edges of the polygon.

**2. VRP Formulation:**
Below are the notations used for the VRP formulation. Reference of Equation from [ [9] P. Toth and D. Vigo]

| Terms | Description |
|---|---|
| **Node Sets** | |
| $P$ | Set of pickup nodes (ie. P= {1,2,........,n}) |
| $D$ | Set of delivery nodes (ie. D= {n+1,........,2n}) |
| $N$ | Overall set of nodes, N = P ∪ D |
| **Transportation Specifications** | |
| $d_i$ | Units to be transported from node i to node n + i |
| $l_i$ | Load change at node i, $l_i = d_i$ for pickup nodes, $l_{i+n} = -d_i$ for delivery nodes |
| **Vehicle Assignment** | |
| $K$ | Set of vehicle categories |
| **Network Representation** | |
| $G$ | Network for each vehicle k, $G = (V + A)$ |
| $V$ | Nodes in N along with origin and destination depots for vehicle k |
| $A$ | Subset of feasible arcs within V x V |
| **Vehicle Departure** | |
| $a_{o(k)} = b_{o(k)}$ | Departure time of vehicle k from its origin depot, $a_{o(k)} = b_{o(k)}$ |
| **Time Constraints** | |
| $[a_i, b_i]$ | Predefined time windows for visiting node i, with service time $s_i$ commencing within this window |
| $s_i$ | Service time at node i |
| $T_{ik}$ | Initiation time of service by vehicle k at node i within V |
| $L_{ik}$ | Cargo load of vehicle k after completion of service at node i within V |

**The objective function is given as**:

1. $min(Longest + Penalty)$

Where, $Penalty = \sum_{i\in P} p_i t_{i.i+n}\, PenaltyCofficient$ , $Longest = max\{\sum_{(i,j)\in A} t_{ij} x_{ijk} \;\forall k \in K\}$ .

Where $PenaltyCoefficient$ is large enough that it try not to get penalized. This is important because we don't want that getting penalized for missing all pickups be better than doing them.

2. $\sum_{k\in K}\sum_{j\in N\cup\{d(k)\}} x_{ijk} \le 1 \;\forall i \in P$

Each pickup location shall have exactly 1 vehicle dis-parting and reaching anywhere pickup or delivery or destination for that vehicle but not origin. To allow missing pickups we will make it less than or equal to 1.

3. $\sum_{j\in N} x_{ijk} - \sum_{j\in N} x_{j.i+n.k} = 0 \;\forall k \in K, i \in P$

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, No. 6, April : 2024

The equation says that the vehicle that picks i is the vehicle that delivers it to i+n where there may be any number of intermediate nodes.

4. $\sum_{j\in P\cup\{d(k)\}} x_{o(k)jk} = 1 \;\forall k \in K$
Each vehicle shall depart from origin and go somewhere.

5. $\sum_{i\in N\cup\{o(k)\}} x_{ijk} - \sum_{i\in N\cup\{d(k)\}} x_{ijk} = 0 \;\; \forall k \in K, j \in N$
For every vehicle that comes to j from anywhere but d(k), must also depart it and vice verca.

6. $\sum_{i\in D\cup\{o(k)\}} x_{i,d(k),k} = 1 \;\; \forall k \in K$
Finally, the vehicle should reach to it's final destination d(k). Constraints (4)-(6) are crafted to delineate a multi commodity flow framework, guaranteeing the initiation of each vehicle's journey from its designated origin depot and concluding at its assigned destination depot.

7. $x_{ijk}\left(T_{ik} + s_i + t_{ijk} - T_{jk}\right) \leq 0 \;\; \forall k \in K, (i,j) \in A$
If vehicle k goes from i to j then the time of starting service at j should be greater than or equal to the sum of time it takes to reach from i to j, the service time $s_i$ and the time the service started at i given by $T_{ik}$.

8. $a_i \leq T_{ik} \leq b_i \;\; \forall k \in K, i \in V$
The time the service started at i by vehicle k given by $T_{ik}$ shall fall within the time window of i given by $a_i$ and $b_i$. Please note that the time to start the service and the time the vehicle reaches at location can be different and thus the vehicle is allowed to wait at the location.

9. $T_{ik} + t_{i,n+i,k} \leq T_{n+i,k} \;\; \forall k \in K, i \in P$
Time for a vehicle to start the service at n+1 shall be greater than or equal to sum of time the service started at i and the time it takes to reach i+n. Each request is subjected to constraints (9), mandating the vehicle's adherence to the sequence of visiting the pickup node before the delivery node.

10. $x_{ijk}\left(L_{ik} + l_j - L_{jk}\right) = 0 \;\; \forall k \in K, (i,j) \in A$
If vehicle k goes from i to j then the load of vehicle at j after service shall be exactly the sum of load of vehicle at k and the loading or unloading load at j.

11. $l_i \leq L_{ik} \leq C_k \;\; \forall k \in K, i \in P$
The load of vehicle at k at i after servicing shall always be less than or equal to it's total capacity and greater than or equal to loading requirement at i.

12. $0 \leq L_{n+i,k} \leq C_k - l_i \;\; \forall k \in K, n+i \in D$
The load at i+n shall for vehicle k after delivery shall always be non negative and under capacity.

13. $L_{o(k),k} = 0 \;\; \forall k \in K$
We set the initial load of every vehicle at depot to 0.

14. $x_{ijk} \geq 0 \;\; \forall k \in K, (i,j) \in A$

15. $x_{ijk} binary \;\; \forall k \in K, (i,j) \in A$
And finally to allow only locations that are in range for vehicle k, we will do

16. $Range_{ik} - \sum x_{jik} \geq 0 \quad \forall k \in K, i \in N$

Or alternatively, you can just have different sets for pickups and delivery locations according to the vehicle k.

## III. Solution:

We've employed OR-tools to meticulously model the equations for our Vehicle Routing Problem (VRP), aiming for an exact solution. OR-tools offer a robust framework tailored for tackling complex optimization challenges like VRP with precision and efficiency. However, we acknowledge the diversity of problem-solving approaches. While OR-tools serve as our primary engine, we encourage exploration of alternative methodologies. Whether through different optimization libraries or custom algorithms, feel empowered to explore and discover the solution approach that aligns best with your specific VRP scenario.

Stochastic methods, including heuristics and meta heuristics, offer powerful tools for solving complex equations when exact solutions are difficult to find. Techniques like genetic algorithms, simulated annealing, or particle swarm optimization can efficiently explore solution spaces, searching for optimal or near-optimal values of VRP by iteratively refining potential solutions based on predefined criteria or fitness functions. The current state-of-the-art solution is the Hybrid Genetic Search [7].

The Hybrid Genetic Search Algorithm (HGS) offers a sophisticated approach to solving the Vehicle Routing Problem (VRP) by integrating genetic algorithms with local search techniques. Beginning with population initialization, HGS generates a diverse set of potential solutions, encompassing both feasible and infeasible routes. Selection of parent solutions for crossover facilitates the exchange of genetic material, producing offspring with potentially enhanced characteristics. The crossover operations, such as Single-Point, Two-Point, or Uniform Crossover, diversify the solution space, aiming for improved quality and diversity. Subsequently, the newly generated offspring undergo a rigorous local search procedure, refining solution quality by considering soft constraints like time windows and vehicle capacities. Penalty weights dynamically adjust during local search, ensuring a balance between feasibility and solution quality.

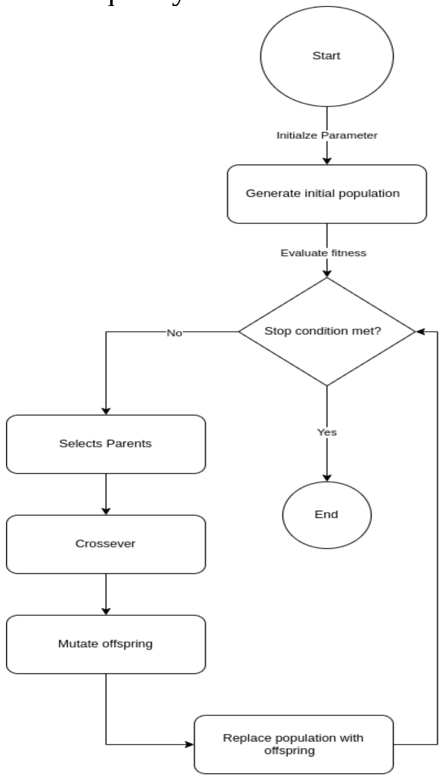

**Fig 1. Hybrid Genetic Search (HGS)**

Population management mechanisms maintain optimal population size and diversity, removing solutions that contribute the least to overall quality and making room for new offspring. Through iterative processes, HGS continuously refines solutions, converging towards high-quality solutions that meet VRP objectives. The algorithm terminates based on predefined criteria, such as reaching a maximum number of iterations or achieving satisfactory solution quality. By systematically navigating the complexities of VRP optimization, the Hybrid Genetic Search Algorithm delivers high-quality routing solutions within reasonable computational effort, making it a valuable tool for logistical planning and optimization.

## IV. Results:

The generation of random territory can be done by following function:

```
 # python code
def generate_random_polygon(center, num_vertices, radius, xrange, yrange):
    angles = np.sort(np.random.rand(num_vertices) * 2 * np.pi)
    distances = np.random.rand(num_vertices) * radius

    x_coords = np.clip(center[0] + distances * np.cos(angles), x_range[0], x_range[1])
    y_coords = np.clip(center[1] + distances * np.sin(angles), y_range[0], y_range[1])

    # Ensure the polygon is closed
    x_coords = np.append(x_coords, x_coords[0])
    y_coords = np.append(y_coords, y_coords[0])

    return x_coords, y_coords
```

This function generates a random polygon based on specific parameters. In polar coordinates, the angle (θ) and distance (r) from the origin (center point) define each vertex. Sine and cosine functions allow us to convert these polar coordinates to Cartesian coordinates (x, y), which represent the vertex positions in a two-dimensional plane.
Specifically, the cosine function (np.cos()) calculates the x-coordinate of each vertex based on the angle (θ) and distance (r), while the sine function (np.sin()) calculates the y-coordinate. This conversion ensures that the vertices are correctly positioned relative to the center point, forming a polygon with the desired shape and orientation.
Below is the driver code.

```
# Generate vehicle territories
territories = []
x_margin = 200
y_margin = 200
for x, y in origins_locations:

    center_point = (x, y)
    num_vertices = np.random.randint(3, 11)  # You can adjust the number of vertices here
    radius = 800
    convex = False  # Change to True for convex polygon

    x_coords, y_coords = generate_random_polygon(
        center_point, num_vertices, radius, x_range, y_range, convex
      )
    # Ensure the original point is inside the polygon
    while not is_point_inside_polygon(center_point, list(zip(x_coords, y_coords))):
```

```
    x_coords, y_coords = generate_random_polygon(
        center_point, num_vertices, radius, x_range, y_range, convex
    )
territories.append(list(zip(x_coords, y_coords)))
```

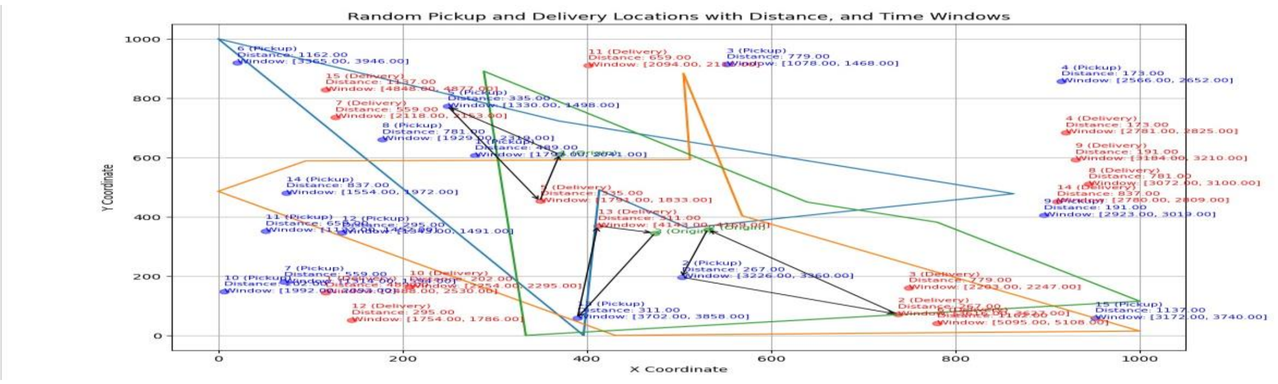

**Fig. 2 Result 1: Route assignment with 3 depots and 15 pickup/destination points**

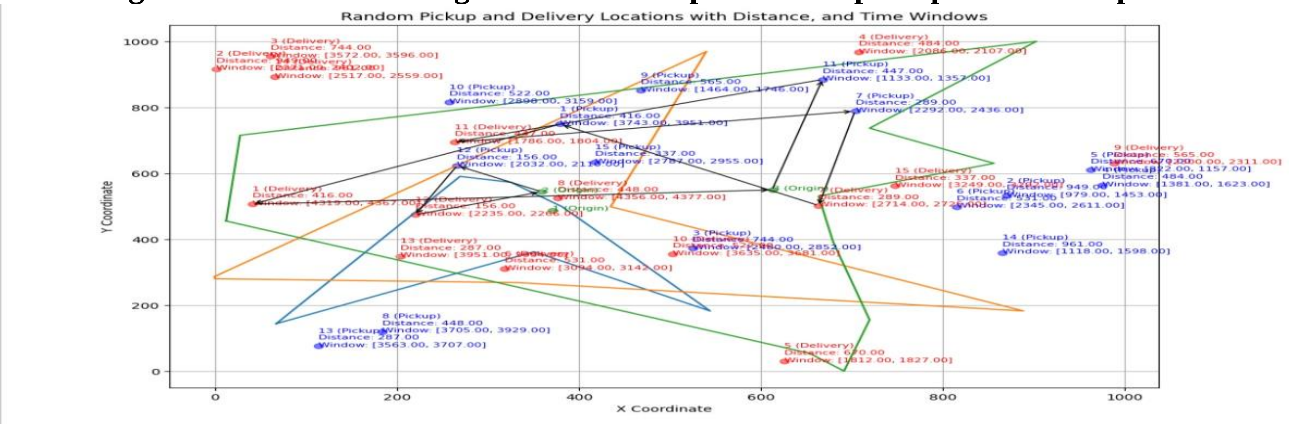

**Fig. 3 Result 2: Route assignment with 3 depots and 15 pickup/destination points**

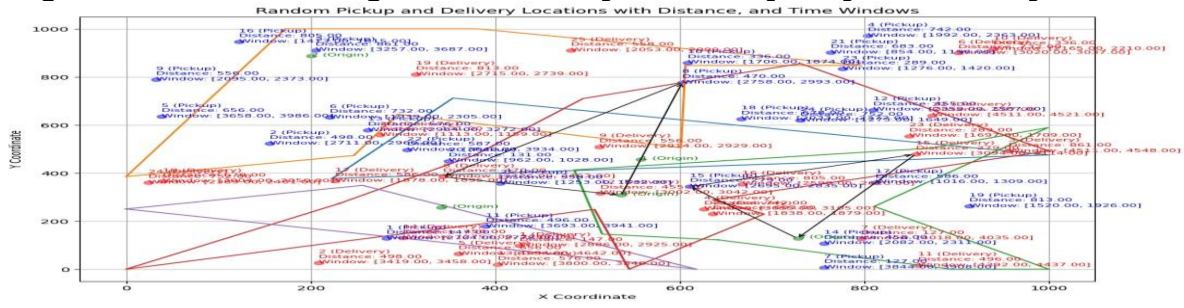

**Fig. 4 Result 3: Route assignment with 5 depots and 25 pickup/destination points**

The green points represent the depots for vehicle $k \in K$, blue points represent the pickup point in $P$ and the corresponding delivery point in D. The black arc is the assigned route, and the rest arcs represent the fences. The distance given in the pickup and delivery labels is the time it takes between to travel from pickup point to the corresponding delivery point. The time windows starts from 0.

## V. Conclusion

In conclusion, this paper presents a comprehensive framework for addressing the complexities of dial-a-ride applications, particularly focusing on deferred scenarios. By integrating semi-soft time windows, vehicle capacity constraints, and penalty considerations for missed pickups, the framework offers a versatile solution adaptable to diverse operational contexts. The utilization of raycasting techniques for fencing ensures precise delineation of vehicle territories within irregular polygons, optimizing routing solutions and resource utilization. Through the employment of OR-tools and the

Hybrid Genetic Search Algorithm (HGS), the framework delivers high-quality routing solutions, enhancing transportation efficiency and service reliability.Overall, this framework represents a significant advancement in transportation logistics, offering innovative solutions to complex operational challenges while fostering sustainability and improving overall quality of service. This framework bridges the gap between theoretical optimization models and practical operational challenges, thereby advancing the transportation sector.

**Acknowledgment:**
1. We extend our heartfelt appreciation to **Prof. (Dr.) Ankita V. Karale,** Head of the Department of Computer Science at Sandip Institute of Technology and Research Centre, Nashik, for her invaluable support and guidance.
2. Our profound gratitude goes to our esteemed Guide, **Prof. Akhilesh Sharma,** whose expertise and direction were instrumental in navigating the intricacies of this project and conducting the experimental research.

## References

[1]A. Mor and M.G. Speranza, “Vehicle routing problems over time: a survey”, 2022.
[2]Boumpa, E.; Tsoukas, V.; Chioktour, V.; Kalafati, M.; Spathoulas, G.; Kakarountas, A.; Trivellas, P.; Reklitis, P.; Malindretos, G. A Review of the Vehicle Routing Problem and the Current Routing Services in Smart Cities. Analytics (2023) , 2, 1-16. https://doi.org/10.3390/analytics2010001
[3]Dana Marsetiya Utama, Shanty Kusuma Dewi, Abdul Wahid & Imam Santoso | Duc Pham (Reviewing editor) (2020) The vehicle routing problem for perishable goods: A systematic review, Cogent Engineering, https://doi.org/10.1080/23311916.2020.1816148
[4]Kris Braekers, Katrien Ramaekers, Inneke Van Nieuwenhuyse, The vehicle routing problem: State of the art classification and review, Computers & Industrial Engineering, Volume 99, (2016), ISSN 0360-8352, https://doi.org/10.1016/j.cie.2015.12.007.
[5]Marta Torgal, Teresa Galvão Dias, Tânia Fontes, A multi objective approach for DRT service using tabu search, Transportation Research Procedia, Volume 52, 2021, ISSN 2352-1465, https://doi.org/10.1016/j.trpro.2021.01.092
[6]M. R. H. Maia, A. Plastino and U. S. Souza, “An improved hybrid genetic search with data mining for the CVRP”, 2024, https://doi.org/10.1002/net.22213
[7]N A. Wouda , L. Lan and W. Kool, “PyVRP: A High-Performance VRP Solver Package”, 2024, https://doi.org/10.1287/ijoc.2023.0055
[8]N. Errami, E. Queiroga, R. Sadykov, E. Uchoa. "VRPSolverEasy: a Python library for the exact solution of a rich vehicle routing problem", 2023
[9]P. Toth and D. Vigo, “The Vehicle Routing Problem” SIAM, 2002, https://doi.org/10.1137/1.9780898718515
[10]Tan, S.-Y.; Yeh, W.-C. “The Vehicle Routing Problem: State-of-the-Art Classification and Review”. Appl. Sci. 2021, 11, 10295.1. Introduction https://doi.org/10.3390/app112110295
[11]Tan, S.-Y.; Yeh, W.-C. “The Vehicle Routing Problem: State-of-the-Art Classification and Review”. Appl. Sci. 2021, 11, 10295.1. Introduction https://doi.org/10.3390/app112110295
[12]T. Vidal, “Hybrid Genetic Search for the CVRP: Open-Source Implementation and SWAP* Neighborhood”, 2022, https://doi.org/10.1016/j.cor.2021.105643
[13]T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, “A hybrid genetic algorithm for multi depot and periodic vehicle routing problems”, (2012), https://doi.org/10.1287/opre.1120.1048