Received: October 27, 2024. Revised: November 29, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Hybrid Strategies for CVRP Initial Solution: Leveraging Weighted Score Insertion with Grid Search and Multiple Insertion with Iterative Tournament

Abbas G. Hafedh[1]* **Hamid M. Hasan[1]**

[1]*Control and Systems Engineering Department, University of Technology, Baghdad, Iraq*
* Corresponding author's Email: cse.22.09@grad.uotechnology.edu.iq

Abstract: This paper presents two novel strategies for solving the Capacitated Vehicle Routing Problem (CVRP): Weighted Score Insertion with Grid Search (WSI-GS) and Multiple Insertion with Iterative Tournament Selection (MI-ITS). The novelty of WSI-GS lies in its weighted demand-distance heuristic which uses balancing customer demand and proximity to the depot for node insertion, refined by systematic parameter optimization using a grid search. MI-ITS novelty is integration of iterative tournament selection to enhance solution diversity by a balance between exploration and exploitation in node insertion. This improves adaptability to various problem settings. Despite improved savings-based algorithms like Clarke-Wright and Parker-Holmes often showing superior overall performance, proposed methods outperformed in several Augerat benchmark instances, reaching optimal or better-than optimal (WSI-GS: P-n22-k2; MI-ITS: B-n50-k7, B-n52-k7, P-n19-k2, P-n20-k2, P-n21-k2, and P-n22-k2). The complementary characteristics of WSI-GS and MI-ITS position them as viable candidates for future hybrid frameworks, providing improved robustness and exploration in CVRP solutions.

Keywords: Cheapest insertion, Demand-distance balance, Grid search, Single and multiple start route, Thread parallelization, Iterative improvement, Tournament selection, Heuristics rules insertion.

1. Introduction

There are many variants of Vehicle Routing Problems (VRP) according to the constraints considered, and CVRP (Capacitated VRP) is deemed the core of these variants because it handles the base presumed constraint which is the vehicle capacity employed for distribution from/to the depot(s) to/from a customer(s) [1]. So, there are many studies have poured light on this literature in recent years [2]. The aim of CVRP is to identify the best paths for delivery or collection routes across all the nodes of the problem being solved while taking care of homogeneous vehicles capacity to be assigned for each path. Because of its wide search space, CVRP is considered Non-polynomial-hard optimization problem [1]. So, its complexity arises as the number of nodes is increased, therefore there is no known single algorithm can be employed to solve all problems [3, 4]. There is a focus on CVRP literature as main part of operation researches nowadays especially as emerging of new technologies in image processing [5, 6] and IoT technologies [7-9] to utilize these techniques to strength CVRP problems.

There are six methods categories used to solve CVRPs [10]. Because of exact algorithms complexity and computation intensity, many heuristics methods have been developed to solve CVRPs [11]. The most important category is heuristics (pure and hybrid) because of its simplicity, and consistency in such way it can be considered as an entry to other method categories by generating initial solutions. Among heuristic category methods, constructive heuristics (like saving, nearest neighbour, cheapest insertion, and sweep algorithms) which build routes sequentially or in parallel way to extend routes by greedily adding unrouted customers approve their efficiency for rapid feasible initial solution generation in addition to be considered preferable choice for projects that require fast, high-quality solutions with little computational overhead because its balance with solution quality [12].

This study bases on Multiple Start Route Cheapest Insertion Heuristic (MSR-CIH) [13] concept to optimize it with grid search and iterative tournament selection. This paper's primary contributions can be summarized as follows:

1 Leveraging MSR-CIH with grid-search optimization to ensure balance between two heuristics metrics (demand and distance).
2 Hybridizing MSR-CIH with iterative tournament selection to cover most of the search space by enhancement in exploration and exploitation.
3 Compare the performance of the developed techniques with recent heuristics algorithms.
4 Analysis the complexity of developed algorithms to highlight the parts that is needed to be reviewed to optimize the results and computation time.

The rest of this paper is presented as follows. First, formulation and related works of CVRPs are demonstrated in section 2. Then, detailing the of proposed techniques in section 3 followed by performance and analysis of the results in section 4. At section 5, the conclusion and room for improvements are presented.

2. Formulating and related work

2.1 CVRP formulating

Despite that CVRP formulation has been widely addressed in previous works [14][10], we revisit formulation to ensure readability and consistency across the paper.

Objective Functions:

The minimization of total cost (overall paths distance, transportation time or any other interesting factors according to the customers' requirements) while ensuring visiting all customers (Eq. (1)).

Constraints:

CVRP are subjected to the following constraints:

1. Each customer is served only once by one vehicle (Eq. (2)).
2. Each vehicle can leave or visit only one customer at time (Eq. (3-a) and Eq. (3-b) respectively). This ensures all routes start from and end at the depot.
3. All vehicles leave the customers after visiting them except the depot (Eq. (4)). This ensure no vehicle stay away from the depot.

Table 1. Notation list

Symbol	Description
The indices and sets	
n	Customers Number
p	Vehicles number
V	Node set, where v_0 is the depot and $\{v_1, v_2 \dots, n\}$ are the customers
i, j	Subscripts of the customers, $i, j = 1,2, \dots, n$
k	Subscript of the vehicle $k = 1,2,..,p$; p number of vehicles
A	$A = \{(v_i, v_j): v_i, v_j \in V\}$; is arcs (paths) set linking nodes i *and* j
Parameters	
D_i	Customer i demand
d_{ij}	Distance between customers i, j
Q	Capacity of the vehicles
Decision Variable	
x_{ijk}	Decision binary value: $x_{ijk} = \begin{cases} 1 \ if\ there\ is\ an\ arc\ from\ i\ to\ j\ driven\ by\ vehicle\ k \\ 0\ otherwise \end{cases}$ where $x_{ijk} \in \{0,1\} \quad \forall k \in \{1, \dots, p\}, i, j \in \{1,...,n\}$ Note: There is no travel from a node to itself $x_{iik} = 0 \quad \forall k \in \{1, \dots, p\}, \forall i \in \{1,...,n\}$
y_{ik}	Decision binary variable: $y_{ik} = \begin{cases} 1 \ if$ vehicle k visits customer $i \\ 0\ otherwise \end{cases}$ where $y_{ik} \in \{0,1\} \quad \forall k \in \{1, \dots, p\}, i \in \{1, \dots, n\}$

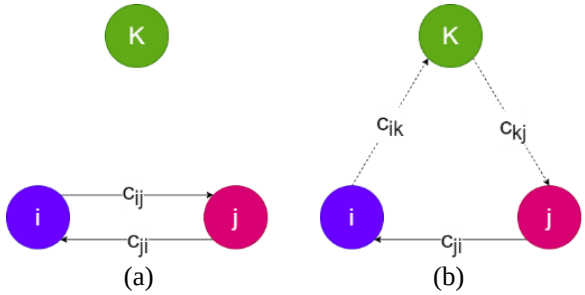

Figure. 1 Cheapest Insertion Example: (a) before insertion and (b) after insertion

4. The vehicles load shall not exceed the vehicle capacity (Eq. (5)).
5. The depot has demand equals to zero.

Table 1 list the notations used to solve CVRP according to the above definition.

So, the minimization of objective function can be expressed by the following equation:

$$Min \sum_{k=1}^{p} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ijk} x_{ijk} \quad (1)$$

Subject to the following constraints:

$$\sum_{k=1}^{p} y_{ik} = 1 \ \forall\, i \in \{1,..,n\} \quad (2)$$

$$\sum_{i=1}^{n} x_{i0k} = 1 \ \forall k \in \{1 \ldots p\} \quad (3\text{-a})$$

$$\sum_{j=1}^{n} x_{0jk} = 1 \ \forall k \ in \{1, \ldots, p\} \quad (3\text{-b})$$

$$\sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik}. \ \forall j \{1,..n\}; \ k \in \{1,..p\} \quad (4)$$

$$\sum_{i=1}^{n} \mathrm{D_i} y_{ik} \leq Q \ \forall k \in \{1, \ldots, p\} \quad (5)$$

2.2 Related work

The (Cheapest Insertion Heuristic) CIH is constructed initially by a subtour composed from a single customer connected to depot, then iteratively inserting other customers in the route in the position that cause least cost increase until no more customers can be added due to the capacity violation. This process is repeated until all customers are included in the feasible solution routes [15]. For example, consider Fig. 1 in which there is two-way-arc between two nodes (i, and j) (a) and a need to insert the k node in this arc, then Eq. (6) is used to calculate the cost increase resulting from such insertion [15]:

$$cost\_increase = c_{ik} + c_{kj} - c_{ij} \quad (6)$$

Where c_{xy} is the Euclidean distance between any node x and node y.

CIH has some drawbacks in spite of its effectiveness and simplicity. One of the main limitations is its greediness which may lead to suboptimal solutions particularly in complex instances with irregular customers geographic distribution, wide-varied demands and complex constraints. The heuristic result will depend mainly on the sequence of nodes insertion into the routes. Second matter, selection of starting routes has significant impact on the final solution. So, requires careful consideration [13]. Furthermore, despite CIH has good efficiency in small to middle-sized problems, its efficiency declines as the problem size increases because heuristic does not explore the whole solution space structure beyond the direct neighbourhood of current route then falling in potential local optima especially in large and complex instances [16].

Several improvements and modifications to CIH have been developed in the literature to address the above limitations to make it useful tool for solving complex and large-scale VRPs. One approach involved **randomness** in insertion process. This to introduce some stochasticity which enhances exploration particularly in complex instances [17]. Hybridizing CIH efficient initial solution generation with **metaheuristic** optimization algorithms gives significant improvement in terms of travelled distance, and number of vehicles used by utilizing advanced search capabilities of such algorithms especially in large-scale problems [18, 1]. Several techniques proposed dynamically **adapted versions** of insertion criteria based on the problem instances characteristics such as current route status, customers demand distribution and vehicle capacity to be responsive to complex environments [1]. Another improvement is involving **machine learning** supervised learning techniques to control the insertion process by training models on historical CVRPs [19]. Furthermore, integrating **reinforcement learning** to improve CIH performance by learning from responses delivered during the optimization process thereby adapting its insertion policy based on previous decisions. In advanced iterations, CIH becomes more effective at determining most promising regions that avoid local optima [20] A significant improvement is using of **parallel-processing** techniques so multiple insertion processes occur simultaneously to accelerate CIH and drastically reduce computation time of extensive exploration of the solution space [16].

In addition to previously discussed improvements, several other enhancements have been proposed to override CIH limitations and extend its capability in solving complex CVRP scenarios:

Table 2. Improvements techniques

Modification	Insight	Strengths	Weaknesses
Hybrid with Metaheuristics [13, 18]	Combines CIH with global search techniques.	Enhancing the search space exploration and escaping local optima.	Increase computational complexity and require careful tuning for parameters.
Adaptive CIH [15]	Dynamically adjust insertion criteria based on problem characteristics.	Improve robustness and raise flexibility.	Different scenarios require different adaption strategies thereby suitable strategy selection complexity
Learning-based CIH [17]	Using ML techniques to guide the insertion process based on historical data	Learns from past data to improve decision making.	Requires large datasets and extensive training with potential overfitting risk.
Parallel Processing [18]	Simultaneous insertion process utilizing parallel computing techniques.	Reduce computation dramatically, modify CIH viable for complex problems.	Implementation complexity and potential synchronization issues.
Clustering techniques [21]	Pre-processing by grouping customers.	Reduce problem complexity	Effectiveness depends on the clustering algorithm.
Multi-objective Optimization [15, 19]	Balance multiple objectives during insertion.	Solutions meet diverse real-world requirements	Balancing complexity and tuning weights appropriately.
Iterative Refinement [16]	Refine solution through multiple runs, improve route each time.	Allows progressive improvement of solution	Time-consuming and may require iterations to achieve significant improvements.
Randomized Insertion [17]	Introduce randomness to explore a wider search space	Enhance diversity and reduce the risk of getting stuck in local optima.	Potential inconsistent results so require multiple runs to ensure robustness.

Table 3. Summary of Single and Multiple Start Route Categories

	SSR-CIH	MSR-CIH
Initialize	Establishes a route with a singular starting node.	Initiates several routes by picking various starting nodes based on parameters such as index, distance, demand, randomness, weighted score, and proximity [13].
Starting Node flexibility	Restricted to the selection of a single node.	Exhibits greater flexibility by initiating several routes [13].
Risk of Inefficient routes	Increased risk resulting from restricted exploration of node combinations.	Reduced risk resulting from the diversity of initial nodes and selection methodologies, facilitating enhanced exploration of possible routes [15, 16].
Complexity	Reduced computational complexity.	Increased computational complexity [13].
Large instance performance	May exhibit suboptimal performance due to the exploration of a limited number of node combinations.	More suitable for larger instances, as exploration and results in more equitable routes [13].
Bias	Susceptible to bias.	Mitigates bias by examining various criteria for selecting initial nodes [13].

Clustering Techniques:

Grouping customers into smaller, more manageable units before applying CIH can dramatically improve heuristic performance. Clustering itself depends on various criteria such as customers distribution or customers' demands to simplify and strengthen the insertion process [21].

Multi-objective optimization:

In reality CVRP application, the objective function is not solely minimizing the distance, but to balance multiple objectives such as balancing vehicle loads or environmental impact. So multi-objective extension of CIH involves these additional criteria into the insertion process to meet the requirements of the problem by assigning dynamic weights during the insertion process [19].

Iterative Refinement:

Where the heuristic is applied multiple times, make adjustments to the insertion process or by re-evaluating the insertion order each time. This

approach iteratively improves the solution quality [16].

Table 2 summarizes these improvements(modifications). Among these improvements, hybridization with metaheuristics algorithms gained attention of most scholars in CVRP literature in recent two decades [2, 11] because of the promising results. Because of new emerging of new novel metaheuristic algorithms, there is a tendency to adapt such algorithms in solving CVRP complex problems because of the modern techniques employed in their phases such as guided searches and arithmetic crossover [22], splitting the swarm into two equal-sized groups to vary the search process and intensify sub-swarms [23] or employing three references in exploring search space [24] after considering converting some of their functionalities to be compatible with permutation-based problems like VRP.

3. Methodology

Traditional sequential insertion iteratively adds nodes to increasing routes based on short-sight metrics like least cost increase. This greedy strategy provides fast and simple solutions, but it is often caught in local optima because it lacks the global perspective to evaluate the early choice influence on solution quality. Furthermore, the heuristic nature of starting with a certain node based on their index, demand, or distance from the depot introduce a bias in the construction of the solution which significantly affects the resulting routes and lead to suboptimal solution, while other may produce more efficient solutions. To overcome these limitations, this paper introduced two MSR-CIH techniques. The key differences between single start route CIH (SSR-CIH) and MSR-CIH are summarized in Table 3.

3.1 Weighted sequential insertion with grid search optimization (WSI-GS)

WSI-GS is motivated by the principles of seed customer selection [16], highlighting the significance of judiciously choosing initial nodes to direct the insertion process. Additionally, the algorithm utilizes the advantages of multiple starting routes [16].

This technique is proposed to overcome the bias resulting from choosing starting nodes based on arbitrary metric such as proximity to depot, demand or customer index, so it depends mainly on assigning certain weights parameters for both demand and distance then calculating score as shown in Eq. (7). This calculation introduces a degree of balance optimization by reduce the travel distance while maximize vehicle capacity utilization. By letting users control weights, WSI-GS is more versatile than purely demand- or distance-based techniques.

$$score_i = dw * D_i + dist\_w * d_{0i} \quad (7)$$

$\forall i \in \{1, \dots n\}; dw$: demand weight; $dist\_w$:distance weight; D_i: customer i demand; d_{0i}: distance between the depot and customer i.

The general scheme of WSI-GS is shown in Appendix B, Algorithm 1.

3.1.1. WSI-GS implementation main steps

WSI-GS key steps can be summarized as follows:

1- Setting Up Grid Search:
   - Take the Cartesian product of all distance and demand weight Options.
   - Test weight settings with all combinations.
2- Choosing Start Node Scores:
   - For Every weight combination: Given distance and demand weights, calculate a weighted score for all nodes; Initialize several start routes with the highest weighted nodes.
3- Build Routes with Insertions:
   - Initialize a route for each start node.
   - Insert remaining nodes into routes sequentially, choosing the lowest cost increase while respecting vehicle capacity.
4- Calculate the best solution:
   - Comparing the cost of the obtained solution for each weight's combination.

3.1.2. WSI-GS complexity analysis

Outer While Loop:

WSI loops around all nodes until all customers are visited, taking O(n) time.

First Node selection (Weighted Score Calculation):

WSI calculates the weighted-score for all unvisited nodes during route initialization. Each route initialization takes O(n).

Node Insertion (Cheapest Insertion):

- To find the best node position in a route, the approach checks up to n positions and estimates insertion costs using the distance matrix. Then, inserting each node takes $O(n^2)$.

So, the complexity for the sequential insertion phase is $O(n^3)$ in the worst case.

GS complexity:

- GS explores different combinations of distance and demand weights. Because there are w_d

options for distance_weight and w_dem options for demand_weights, then the algorithm runs O($w\_d \times w\_dem$) times of WSI.

Since WSI has a complexity of O(n^3), then the overall complexity for WSI-GS technique will be O($w\_d \times w\_dem \times n^3$). So, the complexity depends on the amount of weights options.

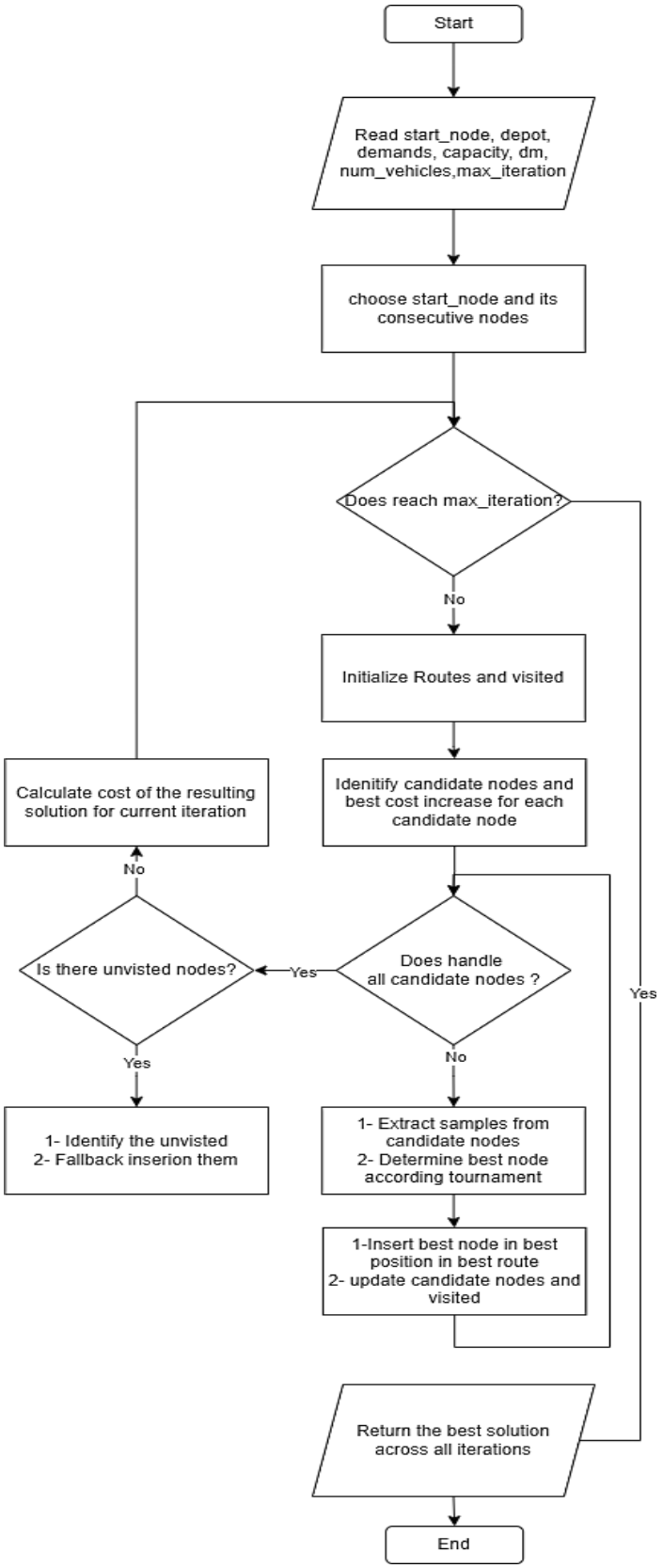

Figure. 2 Worker Method Flow Chart for MI-ITS

3.2 Multiple insertion with iterative tournament selection (MI-ITS)

MI-ITS is based on importance of iterative refinement in developing progressively optimized routes and multiple start routes concept [14, 13]. It prioritizes diversity, as highlighted by [15, 16] which improves global search capabilities and maintains a balanced exploration-exploitation trade-off essential for addressing complex vehicle routing problems.

Three approaches comprise MI-ITS. The main (calling) method calls the worker (core) method with the start node parameter, worker calls the tournament selection method with candidate nodes. MI-ITS main purpose is to reduce the potential bias that may be resulted from starting with certain nodes by looping through all nodes as starting node and passes that start node to worker method and evaluates the resulted routes cost then chooses the lowest cost solution. The core innovation of this algorithm is leveraging multiple start route strategy with two enhancement techniques: tournament and iteration. The tournament introduces stochastic element that enhances exploration thus avoiding local optima which is a common problem in deterministic insertion algorithms that follow purely greedy insertion approaches, and exploitation (improving known good solutions) while iteration improves the finding better routes chance over time.

Fig. 2 shows worker method implementation. The MI-ITS pseudo code is in Appendix B, Algorithm 2.

3.2.1. MI-ITS implementation main steps

MI-ITS key steps can be summarized as follows:

1- Iterative selection for start nodes:
Repeat for each start node to call the worker function. Each start node starts several routes with successive nodes. So, the start_node will be the starting node for first vehicle ($k = 1$), while Eq. (8) will calculate the consecutive nodes.

$$C_k = \begin{cases} S + k - n + 1 \text{ if } (S + k) \geq n \\ (S + k)\ mod\ \ n, Otherwise \end{cases} \quad (8)$$

$\forall k \in \{2 \dots p\}$; C_k: start node for vehicle k; S: starting node; n: number of customers; p: vehicles number.

2- Tournament Selection process:
Evaluate candidate nodes and choose the best using k-tournament selection per iteration. Place the chosen node with minimal cost increase.

3- Solution selection for each iteration:
Record each iteration's cost and solution. After iterations, choose the optimal solution with fewest routes and lowest cost.

4- Return the best solution:
Return all-iteration best solution.

3.2.2. MI-ITS complexity analysis

Outer Loop:

Algorithm run for $max\_iteration$ iterations. Each iteration reinitializes the routes and does node insertion using tournament selection. So, the complexity of the outer loop is O($max\_iteration$).

Tournament Selection:

The method estimates the cost of adding each unvisited node to all routes and placements. So, tournament evaluates each node n times. The sample usually has fewer candidates than n, therefore the tournament runs O(k).

Node Insertion:

The worker complexity is O($k \times n^2 \times$ max $\_iteration$) for each insertion since it calculates the cost increase of putting a node at all feasible positions in all routes.

Total complexity:

The worker is called n times Thus, the total complexity will be O($k \times n^3 \times$ max $\_iteration$).

4. Experimental analysis

This section introduces the numerical results of experimenting of the proposed methods on 74 Augerat instances- whose key traits are in Table 4- existing in the literature [25]. Because of high efficiency of saving method in constructing initial solutions as demonstrated by article [13] in related work section, the proposed strategies are compared with two state-of-art methods -(Improved Clarke-Wright (ICW) and Improved Parker-Holmes (IPH))- [10] to verify their effectiveness. Most important features of PC used in experimental application: Intel(R) Core (TM) i7 x64-based CPU; Physical Memory (RAM):16.0 GB; Programming language: Python. All instances depots are symmetric, centered, or non-centered. Eq. (9) calculates the ratio of total demands to vehicle capacity.

$$RDC = \sum_{i \in n} D_i \,/(Q * P) \quad (9)$$

Where RDC :Ratio of demands to capacity; Q : Vehicle capacity; P : Number of vehicles; D_i : customer ith demand; n: customers number.

Table 4. Main Attributes of Augerat Benchmark Instances

Set / Feature	Set (A)	Set B	Set P
Locations and demand [25]	Randomly generated customer locations and demands	More realistic scenarios such as where customers are located in certain urban or business areas with more realistic demands	Some instances are adapted from datasets with different vehicle capacities and demands.
Variety of problem sizes [25]	Range from small to medium	Medium sizes	Range widely in size, thereby covering wide CVRP spectrum.
Clustering nature of customer location [25]	Nodes are uniformly or non-uniformly dispersed	Customers are geography clustered	Mix of clustered and non-clustered customers

Table 5. Comparison of Techniques in Multiple Start Route Insertion

	MSR-Farthest	MSR-Highest Demand	MSR-Random	MSR-Nearest	MSR-WSI-GS
Initiating Node Selection	Gives precedence to nodes that are most distant.	Based on nodes demand	Randomly select nodes to start routes	Prioritizes the closest nodes to the depot for route initiation.	Weighs distance and demand, then using grid search.
Exploration	Prioritizes distant nodes initially, potentially lead to neglecting load balancing at the outset.	Focuses on high-demand nodes, may overlook proximity.	Randomness enhances exploration but may result in suboptimal route formation.	Emphasizes proximity, yet may result in an ineffective demand equilibrium.	quittable strategy takes into account both distance and demand, optimizing each element.
Bias	Lead to suboptimal routes if the distant nodes have higher demands	Could lead to less efficient routes because high demand takes precedence over distance.	While it minimizes bias, random selection can result in varying solutions.	It may promote shorter distances while missing a more important demand distribution	Minimize bias due to balanced selection.
Complexity	Moderately complex.	Moderately complex.	Lower complexity because of randomness.	Minimal complexity, as it depends on proximity for node selection.	Increased complexity due to grid search optimization.
Large instances performance	Appropriate for scenarios where remote nodes substantially influence routing expenses.	Exhibits optimal performance when demand is a pivotal element in route balancing.	Inconsistent performance	Most appropriate for situations when closeness to the depot is of paramount importance.	Perform well consistently, especially in large instances with high-varied demand and distance nodes.
Route optimization	Concentrates on remote nodes, potentially lowering total route length.	Regulates load demand, although may increase cost.	Random performance yields inconsistent efficiency.	prefers shorter starting routes but may result in an uneven load distribution.	offers the ideal ratio of reducing distance to balancing demand.

Eqs. (10) and (11) calculate the problem standard deviation (square root of variance).

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(D_i - \mu)^2} \quad (10)$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} Di \quad (11)$$

Where σ: Problem standard deviation; n: customers number; μ: demand mean; D_i:ith Customer demand;

Recalling that there are two categories of insertion procedures in terms of start routes: single start route (SSR) and multiple start route (MSR) categories[13].

At the outset of a SSR algorithm, a single route is initialized, with a single node selected using a predetermined heuristic rule (such as its index or closeness to the depot, for example). Each route in a MSR algorithm starts with a different node according to a variety of criteria, such as:

- the distance from the depot (nearest or farthest)
- prioritizing nodes based on customer demand
- choosing start nodes randomly
- Weighted score insertion: by combing both nodes proximity and demand.

Furthermore, by utilizing parallelization scheme, parallel improved insertion- with consecutive nodes (PII-CN) which is used to send multiple start nodes as argument in parallel threads to the worker method which by turn initialize multiple start routes by choosing consecutive nodes to the passed start node as shown in step 2 of the section 3.1.1. This implementation is used to avoid bias by starting with a certain node.

Table 5 summarize the main differences between some MSR adapted variants [16, 13].

Euclidean distances—rounded integer node distances—are used in optimal solutions for the three primary CVRP datasets [26]. Appendix A Table 1 demonstrates the solutions obtained and optimal value for all benchmark instances. The gap between an algorithm's solution and the literature's optimal solution measures its performance[13]. So, solution percentage gap (or solution quality) is calculated according to Eq. (12) below.

$$gap = \frac{(sol_{obt} - sol_{opt})}{sol_{opt}} \times 100\% \quad (12)$$

Where sol_{obt} is obtained solution for a certain instance; sol_{opt} the best-known solution.

Appendix A Tables 2 calculates the solution quality for all implementations. The overall performance of a particular algorithm within a dataset can be determined by the mean percentage gap for that algorithm over all instances as shown in Eq. (13).

$$mean_{set} = \frac{\sum_{ins \in \{set\}} gap_{ins}}{set\_num} \quad (13)$$

$\forall set \in \{A, B, P\}$; ins is instance within a set; gap_{ins} algorithm gap for ins;set_num: number of instances in a certain set.

Table 6 calculates the mean percentage gap for all implementations on all sets.

Table 6. All Implementations' Mean Percentage gap (%) on all sets

Implementation	Set A	Set B	Set P
SSR	36.37	37.77	24.78
MSR-Farthest	46.42	60.28	27.39
MSR-Highest Demand	27.33	27.47	21.43
MSR-WSI-GS	20.35	17.1	13.99
MSR-Random	29.21	28.96	21.59
MSR-Nearest	35.88	44.67	21.21
PII-CN	15.38	11.37	9.263
MI-ITS(k=3)	11.37	5.943	6.967
MI-ITS(k=5)	11.8	6.3044	6.815
ICW	3.41	2.68	4.397
IPH	2.76	2.61	2.49

On **Set A**, **MSR methods** outperform SSR because of elimination of start node bias. A significant feature that improves WSI-GS performance is the cartesian product of demand and distance weights strengths. WSI-GS. The algorithm has the lowest mean gap across MSR variations at 20.35% because it adapts better to non-clustered, dispersed locales. SSR and MSR-Farthest struggle without clustering or spatial trends. The 36.37% gap reveals SSR's greedy, rigid nature which struggles with diversified input. MI-ITS (k=3 and k=5) outperforms PII-CN with 11.37% and 11.8% mean gaps. Due to lack of clustering, dynamic and randomized datasets benefit from iterative node insertion search.

With fewer nodes than Set A but larger demands, **Set B** requires precise starting positions, therefore the weighted score and highest demand algorithms prioritize starting nodes by demand and distance. Its adaptive grid search approach helps WSI-GS achieve 17.1% mean gap. Geographically restricted clusters benefit from weight scoring that balances distance and demand. MI-ITS (k=3 and k=5) exceeds PII-CN (11.37% gap) with mean gaps of 5.943% and 6.304%. and achieves optimal solution in instances (B-n50-k7, and B-n52-k7). In clustered contexts, precise node insertion enhances solution quality, making iterative tournament selection useful. Due to its leaning toward geographically adjacent nodes, MSR-nearest performs poorest on Set B (44.67% mean gap) in complicated, clustered layouts.

Set P, merits by the following attributes: Diverse demands, complex depot location, capacity constraint tightness and random nodes selection. So, WSI-GS handles mixed scenarios well with a 13.99% mean gap and achieves better than optimal solution in instance (P-n22-k2). Its grid search optimization balances clustered and non-clustered nodes. With mean gaps of 6.967% and 6.815%, MI-ITS (k=3 and k=5) likewise perform well and achieves better than optimal solution on instances (P-n19-k2, P-n20-k2,

P-n21-k2 and P-n22-k2). Iterations manages Set P instances' heterogeneity. MSR Random find better than optimal solution in instance P-n21-k2 but the main issue with randomization is inconsistency.

PII-CN outperforms all MSR variations (including WSI-GS) and SSR, with mean gaps of 15.38% on Set A, 11.37% on Set B, and 9.263% on Set P. Because start node bias is eliminated. MI-ITS perform better than PII-CN because of start node bias elimination in addition to iterative tournament selection mechanism which leads to robust performance. PII-CN uses parallelization to test many starting nodes at once and select the optimal solution. While MI-ITS prioritizes tournament selection for iterative node insertion. This method uses controlled randomization to find more promising solutions. Increasing tournament size (k) influences exploration-exploitation balance. A bigger tournament size (k = 5) lowers unpredictability and emphasizes near-greedy solutions, whereas a smaller size (k = 3) stimulates exploration but may miss locally optimal solutions. However, because of iterative and tournament, we avoid using thread pooling in MI-ITS because it may cause thread management overhead and synchronization difficulties, reducing computing efficiency.

Improved savings algorithms perform best overall, with mean gaps of 2.49% to 4.397% across all sets. They prove their initial solution generating dominance and academic literature benchmark status. WSI-GS and MI-ITS perform well on all datasets with considerable strengths particularly in complicated structures or randomized inputs, sometimes providing optimal or better-than-optimal solutions as shown in Figs. 3 to 5. They improve CVRP initial solution generation with creative methods, although ICW and IPH outperform them. They can be deemed as scholarly contributions because of the following principal rationales

1. Insertion algorithms are more adaptable:
   - Insertion-based techniques like MI-ITS enable dynamic route construction, improving adaptability to non-uniform demand distributions, especially in complex instances like Set P.
   - MI-ITS enables deeper exploration and better-than-optimal solutions when the savings-based heuristic fails to locate the global optimum.
   - WSI-GS mitigates node selection bias: WSI-GS uses weighted scoring functions optimized with grid search to avoid bias in starting routes with any node. It balances proximity and demand better than standard insertion methods, producing competitive results and sometimes outperforming ideal solutions.
2. Further exploration than Greedy Saving: These techniques provide alternate approaches for initializing and constructing routes that mitigate the bias associated with savings-based heuristics. This illustrates their promise for further investigation and enhancement for some cases.
3. Hybridization Opportunities: Mixing these techniques with savings-based methods combines their strengths—flexibility and efficiency.

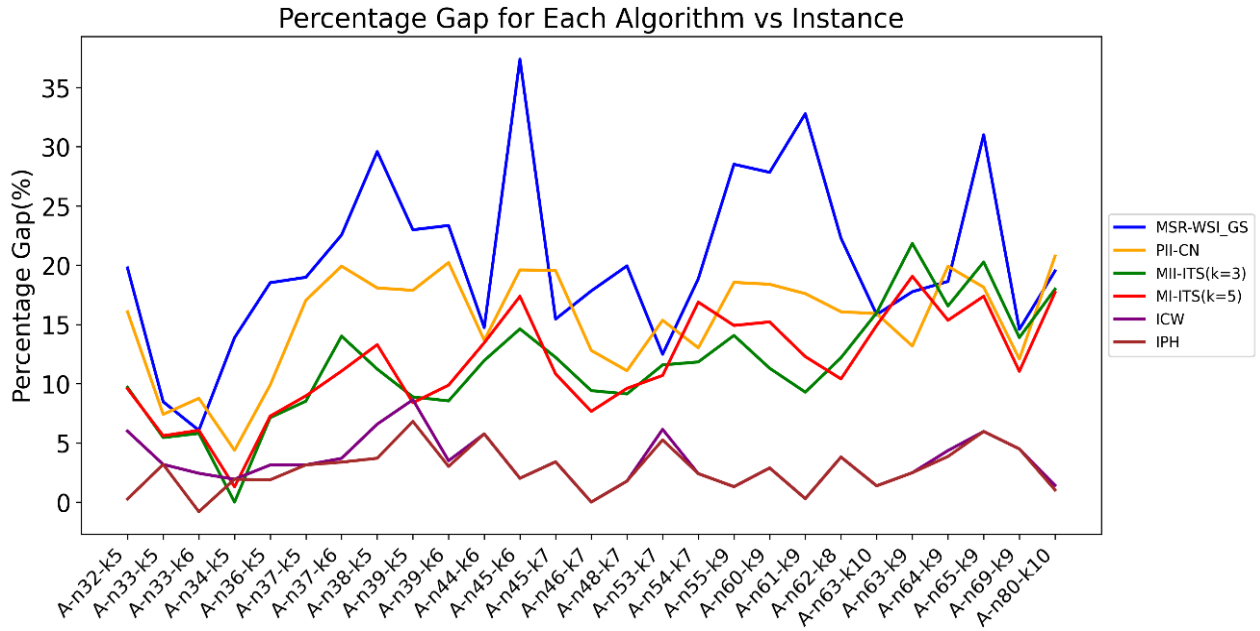

Figure. 3 Elite Implementations gap on Set A

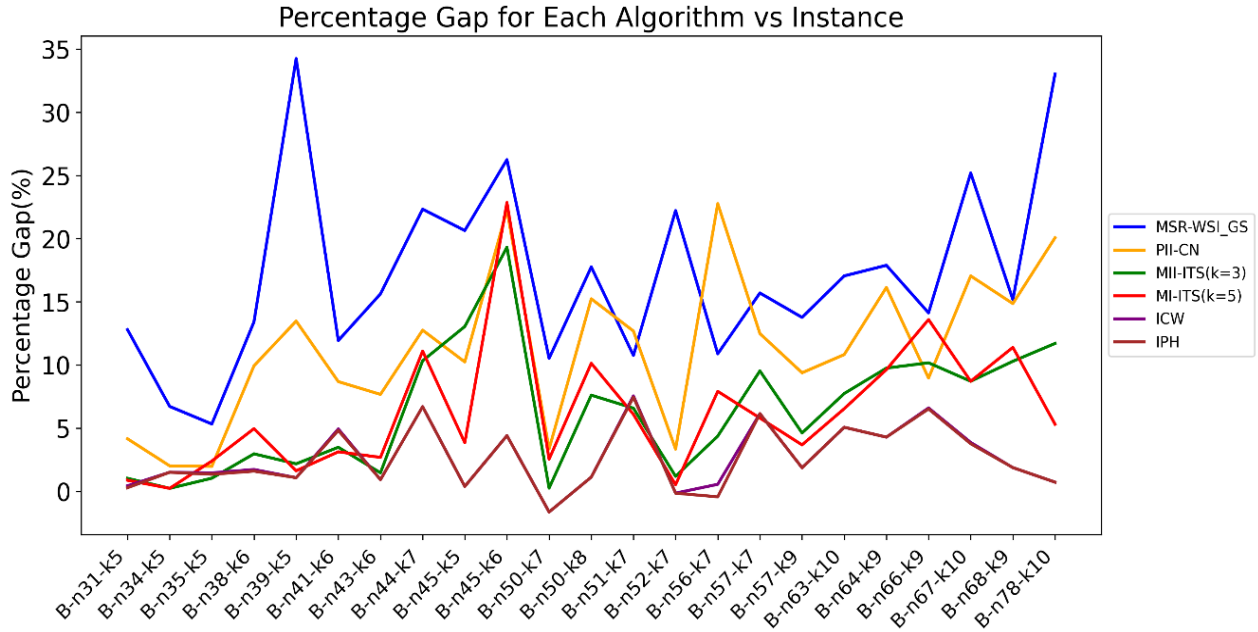

Figure. 4 Elite Implementations gap on Set B

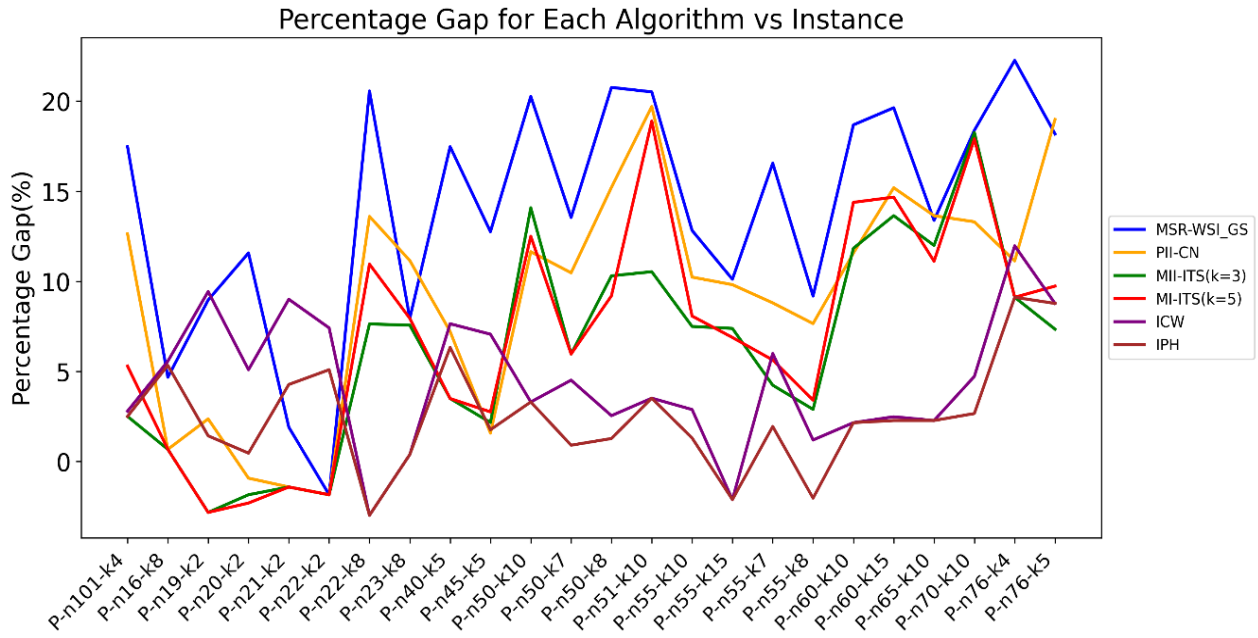

Figure. 5 Elite Implementations gap on Set P

5. Conclusion, limitations and room for improvement

5.1 Conclusion

This paper presents two novel algorithms, WSI-GS and MI-ITS, as innovative methods to tackle the challenges associated with generating high-quality initial solutions and remain relevant in contexts that require variety and flexibility. Both strategies fall under the category of multiple start routes MSR, designed to address the constraints of conventional insertion algorithms.

The **WSI-GS** aimed at enhancing the route initialization process by utilizing a weighted score mechanism. The algorithm is driven by the principles of seed customer selection, highlighting the significance of meticulously choosing initial nodes to direct the insertion process. Through the integration of distance and demand priorities into a weighted score, along with the application of grid search optimization, WSI-GS adeptly harmonizes these

elements to produce high-quality initial solutions. Additionally, it utilizes the advantages of various starting routes as other MSR variants. It improves the examination of possible routes while reducing biases of SSR techniques, leading to a more thorough and fair analysis of the solution space. WSI-GS implementation shows superiority over several variants of MSR in addition to traditional SSR implementation.

The **MI-ITS** presents a methodical framework aimed at improving solution quality via Iterative tournament selection, which provides multiple benefits. This method enhances node selection by evaluating a varied array of candidate nodes during each iteration, thereby reducing the likelihood of early convergence to less optimal solutions. It improves the exploration phase by iteratively selecting nodes through a tournament mechanism, effectively balancing the exploration- exploitation trade-off. This approach is essential for addressing complex vehicle routing problems, as it ensures a more solution space comprehensive coverage during the search process. The variation in node selection enhances the quality of the routes constructed and raises the probability of identifying near-optimal or globally optimal solutions. Furthermore, MI-ITS utilizes the principle of multiple start routes by starting routes from various starting nodes. MI-ITS implementation shows superiority as compared to other several MSR variants including WSI-GS. It also outperforms PII-CN algorithm which merits with start node bias elimination.

The numerical analysis indicates both techniques do not consistently surpass ICW and IPH; however, both methods exhibit considerable potential, especially in scenarios necessitating dynamic, flexible, and high-quality initial solutions. MI-ITS shown exceptional efficacy in managing non-uniformly dispersed nodes and demand fluctuations, whereas WSI-GS provided consistent and dependable solutions with reduced computational expenses. The findings demonstrate the scholarly value of both strategies in cases where savings algorithms may not perform optimally.

MI-ITS performs best of the two methods. The following situations favor it:

1. Cases where Demand Variability is High:
   MI-ITS excels in managing network demand imbalances by considering multiple starting nodes and dynamically inserting customers based on tournament choices.
2. Uneven Node Distributions:
   Savings-based algorithms may struggle when nodes are geographically dispersed and route merging is difficult. MI-ITS's iterative tournament selection can balance geographical proximity and demand limits to find optimal insertion places. benefit: Tournament selection iteratively considers local improvements, avoiding greedy errors in savings algorithms.
3. Diversification in complex routing networks:
   By initializes numerous routes using different starting nodes and then selects candidates. MSR and tournament selection increase the likelihood of finding near to or better than optimal solutions by extensive exploration.
4. When high computational resources available:
   The tournament-based selection can increase computational load, yet MI-ITS performs well with enough power. So, leveraging it with parallelization can improve solution quality.
5. Situations Needing Dynamic Route Construction:
   MI-ITS builds routes repeatedly, considering each node's insertion dynamically, unlike methods that build entire route based on available heuristic rule. This flexible strategy improves adaptability and accommodates last-minute modifications or constraint.
6. Prioritizing Solution Quality over Speed:
   MI-ITS iteratively explores various candidate nodes to find high-quality solutions at the cost of computing performance. This makes it excellent for situations when superior solutions are crucial. This is benefiting logistics applications where even little changes results in large cost reductions.
7. Large-Scale Problems with numerous vehicles:
   Multiple vehicle deployments work well with MI-ITS because it assigns distinct start nodes to initialize routes. Tournament-based insertion maximizes vehicle use without overloading roads. So, better performance can be ensured by the ability to provide balance between vehicles numbers and workload, when traditional savings algorithms may face capacity restrictions.

Despite its elevated computational expense, the parallelization feature offsets this drawback, giving MI-ITS an excellent contender for extensive, complicated VRP instances where the attainment of superior solutions is essential. These algorithms have promising VRP potential advancement, according to our findings.

5.2 Limitations

Despite the encouraging outcomes, both MI-ITS and WSI-GS are constrained by specific limitations:

1. **Computation overhead:**
   By iterative tournament selection, especially for large situations.

2. **WSI-GS parameter tuning sensitivity:**
   Relying on carefully selected grid search weights, which may limit its scalability.
3. **Performance shortage in certain instances**:
   Both proposals outperform savings-based techniques in some cases. This highlights the bias caused by node initialization processes.
4. **Absence of Parallelization in MI-ITS:**
   Without parallelization, MI-ITS is limited. The iterative tournament selection process makes it difficult to parallelize without reducing algorithm efficiency for large Problems instances.

5.3 Future work and room for improvements

1. **Considering MI-ITS effective parallelization:**
   With maintain its iterative selection characteristics [19]. This may entail hybrid parallel models or asynchronous processing methods to address iterations challenges.
2. **Dynamic parameter modification:**
   Adaptive methods to adjust tournament size dynamically [26] within MI-ITS could reduce computational overhead while maintaining solution quality.
3. **Improving scalability:**
   Exploring methods to diminish the computational complexity of both MI-ITS and WSI-GS may enhance the scalability for bigger instances.
4. **Hybrid methodologies:**
   Combining MI-ITS and WSI-GS with savings-based algorithms may produce hybrid methodologies that optimize the balance between exploration and exploitation. Furthermore, utilizing metaheuristic approaches such as Simulated Annealing, Tabu Search or other state-of-the-art swarm intelligence algorithms [27, 28] can enhance the algorithms' capacity to avoid local optima [2].
5. **Enhanced weight optimization in WSI-GS:**
   Refining the **scoring** system by employing machine learning or adaptive optimization has the potential performance improvement [29, 30].
6. **Evaluating on more extensive and practical scenarios:**
   Further **investigation** should prioritize comprehensive evaluations on larger and real-world datasets, including urban logistics and e-commerce delivery networks, to determine the practical applicability of these algorithms.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author Contributions

Conceptualization, 2; methodology, 1; software, 1; validation, 1, and 2; formal analysis, 1; investigation, 1; resources, 1; data curation, 1; writing—original draft preparation, 1; writing—review and editing, 1; visualization, 1; supervision, 2; project administration, 2;
1: Author 1, 2: Author2

References

[1] Z. H. Ahmed, A. S. Hameed, M. L. Mutar, and H. Haron, "An enhanced Ant Colony system algorithm based on subpaths for solving the capacitated vehicle routing problem", *Symmetry*, Vol. 15, No. 11, p. 2020, 2023.

[2] A. K. Garside, R. Ahmad, and M. N. Bin Muhtazaruddin, "A recent review of solution approaches for green vehicle routing problem and its variants", *Operations Research Perspectives*, pp. 100303-100303, 2024,

[3] B. R. Vangipurapu and R. Govada, "A construction heuristic for finding an initial solution to a very large-scale capacitated vehicle routing problem", *RAIRO - Operations Research*, Vol. 55, No. 4, pp. 2265-2283, 2021.

[4] P. Garrido and C. Castro, "A Flexible and Adaptive Hyper-heuristic Approach for (Dynamic) Capacitated Vehicle Routing Problems", *Fundamenta Informaticae*, Vol. 119, No. 1, pp. 29-60, 2012.

[5] H. M. Hasan, "Image Based Vehicle Traffic Measurement", *Engineering and Technology Journal*, Vol. 32, No. 11, pp. 2722-2733, 2014.

[6] M. A. Uthaib and M. S. Croock, "Multiclassification of license plate based on deep convolution neural networks", *International Journal of Electrical and Computer Engineering*, Vol. 11, No. 6, pp. 5266-5276, 2021.

[7] A. J. Humaid, H. M. Hasan, and F. A. Raheem, "Development of Model Predictive Controller for Congestion Control Problem", *Iraqi Journal of Computers, Communication, Control & Systems Engineering*, Vol. 14, No. 3, 2014.

[8] H. Ameer and H. Hasan, "Enhanced MQTT Protocol by Smart Gateway*", Iraqi Journal of Computer Communication Control and System Engineering*, pp. 53-67, 2020.

[9] G. Drewil and R. Al-Bahadili, "Environmental Pollution Monitoring System Based on IoT",

Iraqi Journal of Computer Communication Control and System Engineering, pp. 1-10, 2022.

[10] A. G. Hafedh and H. M. Hasan, "Design and Implementation of an Improved Parallel-Accelerated Solution for Urban CVRP. Baghdad as a Case Study", *International Journal of Intelligent Engineering and Systems*, Vol. 17, No. 5, pp. 934-951, 2024, doi: 10.22266/ijies2024.1031.70.

[11] M. K. D. D. Sandaruwan, D. M. Samarathunga, and W. B. Daundasekara, "An improved two-phased heuristic algorithm for the capacitated vehicle routing problem and a case study", *Ceylon Journal of Science*, Vol. 49, No. 4, p. 477, 2020.

[12] A. H. Ismail, "Domino algorithm: a novel constructive heuristic for traveling salesman problem", *IOP Conference Series Materials Science and Engineering*, Vol. 528, No. 1, p. 012043, 2019.

[13] S. M. Avdoshin and E. N. Beresneva, "Constructive heuristics for Capacitated Vehicle Routing Problem: a comparative study", In: *Proc. of the Institute for System Programming of the RAS*, Vol. 31, No. 3, pp. 145-156, 2019.

[14] S.-Y. Tan and W.-C. Yeh, "The Vehicle Routing Problem: State-of-the-Art Classification and Review", *Applied Sciences*, Vol. 11, No. 21, p. 10295, 2021.

[15] M. Alssager, Z. A. Othman, and M. Ayob, "Cheapest Insertion Constructive Heuristic based on Two Combination Seed Customer Criterion for the Capacitated Vehicle Routing Problem", *International Journal on Advanced Science Engineering and Information Technology*, Vol. 7, No. 1, p. 207, 2017.

[16] E. Yuliza, F. M. Puspita, and S. S. Supadi, "Heuristic approach for robust counterpart open capacitated vehicle routing problem with time windows", *Science and Technology Indonesia*, Vol. 6, No. 2, pp. 53-57, 2021.

[17] M. I. Takano and M. S. Nagano, "Evaluating the performance of constructive heuristics for the blocking flow shop scheduling problem with setup times", *International Journal of Industrial Engineering Computations*, Vol. 10, No. 1, pp. 37-50, 2019.

[18] B. Keçeci, F. Aliparmak, and I. Kara, "A Mathematical Formulation and Heuristic Approach for The Heterogeneous Fixed Fleet Vehicle Routing Problem with Simultaneous Pickup And Delivery", *Journal of Industrial and Management Optimization*, Vol. 17, No. 3, pp. 1069-1100, 2021.

[19] X. Liu, Y.-L. Chen, L. Y. Por, and C. S. Ku, "A Systematic Literature Review of Vehicle Routing Problems with Time Windows", *Sustainability*, Vol. 15, No. 15, p. 12004, 2023.

[20] O. Udomkasemsub, B. Sirinaovakul, and T. Achalakul, "PHH: Policy-Based Hyper-Heuristic with Reinforcement Learning", *IEEE Access*, Vol. 11, pp. 52026-52049, 2023.

[21] T. D. C. Le, D. D. Nguyen, J. Oláh, and M. Pakurár, "Clustering Algorithm for A Vehicle Routing Problem with Time Windows", *Transport*, Vol. 37, No. 1, pp. 17-27, 2022.

[22] P. D. Kusuma and M. Kallista, "Swarm Space Hopping Algorithm: A Swarm-based Stochastic Optimizer Enriched with Half Space Hopping Search", *International Journal of Intelligent Engineering and Systems*, Vol. 17, No. 2, pp. 670-682, 2024, doi: 10.22266/ijies2024.0430.54.

[23] P. D. Kusuma and A. Dinimaharawati, "Swarm Bipolar Algorithm: A Metaheuristic Based on Polarization of Two Equal Size Sub Swarms", *International Journal of Intelligent Engineering and Systems*, Vol. 17, No. 2, pp. 377-389, 2024, doi: 10.22266/ijies2024.0430.31.

[24] P. D. Kusuma and A. Dinimaharawati, "Extended Stochastic Coati Optimizer", *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 3, pp. 482-494, 2023, doi: 10.22266/ijies2023.0630.38.

[25] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the Capacitated Vehicle Routing Problem", *European Journal of Operational Research*, Vol. 257, No. 3, pp. 845-858, 2016

[26] T. Pichpibul and R. Kawtummachai, "An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem", *ScienceAsia*, Vol. 38, No. 3, pp. 307-318, 2012.

[27] P. D. Kusuma and M. Kallista, "Migration-Crossover Algorithm: A Swarm-based Metaheuristic Enriched with Crossover Technique and Unbalanced Neighbourhood Search", *International Journal of Intelligent Engineering and Systems*, Vol. 17, No. 1, pp. 698-710, 2024, doi: 10.22266/ijies2024.0229.59.

[28] P. D. Kusuma and A. L. Prasasti, "Guided Pelican Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 6, pp. 179-190, 2022, doi: 10.22266/ijies2022.1231.18.

[29] P. Czuba and D. Pierzchała, "Machine Learning methods for solving Vehicle Routing Problems", In: *Proc. of the 36th International Business*

Information Management Association (IBIMA), Granada, Spain, pp. 4-5, 2020.

[30] R. Asín-Achá, O. Goldschmidt, D. S. Hochbaum, and I. I. Huerta, "Fast Algorithms for the Capacitated Vehicle Routing Problem using Machine Learning Selection of Algorithm's Parameters", In: *Proc. of International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K - Proceedings, Science and Technology Publications*, Lda, pp. 29-39, 2022.

Appendix A

Table 1. All implementations Results on All sets

Seq	Instance	unhandled	SSR	msr_farthest	msr_highest_demand	msr_weighted_GS	msr_random	msr_nearest	PII-CN	MI-ITS(k=3)	MI-ITS(k=5)	ICW	IPH	optimal_val
1	A-n32-k5	2067	1127	1205	1049	939	929	1073	910	857	868	831	786	784
2	A-n33-k5	1847	983	1020	915	717	833	1014	710	697	698	682	682	661
3	A-n33-k6	1516	1161	1179	1089	787	893	898	807	784	788	760	736	742
4	A-n34-k5	1905	1047	914	851	886	1020	959	812	782	788	793	793	778
5	A-n36-k5	1981	1073	1075	1110	947	1049	1160	878	853	842	824	814	799
6	A-n37-k5	1578	1045	889	716	796	750	878	783	715	720	690	690	669
7	A-n37-k6	2036	1247	1314	1171	1163	1219	1199	1138	1094	1073	984	981	949
8	A-n38-k5	2232	1051	1254	867	946	1127	982	862	814	848	778	757	730
9	A-n39-k5	2311	1078	1142	1003	1011	1057	1113	969	919	897	893	878	822
10	A-n39-k6	2366	1133	1213	949	1025	1201	1309	999	910	891	860	856	831
11	A-n44-k6	2583	1249	1381	1060	1075	1181	1183	1065	1065	1048	991	991	937
12	A-n45-k6	2793	1315	1517	1147	1297	1250	1434	1129	1101	1106	963	963	944
13	A-n45-k7	2633	1443	1470	1523	1323	1462	1460	1370	1290	1292	1185	1185	1146
14	A-n46-k7	2671	1283	1413	1213	1077	1141	1320	1031	974	1002	914	914	914
15	A-n48-k7	2861	1406	1582	1534	1287	1386	1496	1192	1168	1183	1092	1092	1073
16	A-n53-k7	3167	1488	1626	1467	1136	1208	1300	1165	1057	1114	1072	1063	1010
17	A-n54-k7	3463	1503	1699	1472	1387	1590	1453	1319	1321	1326	1195	1195	1167
18	A-n55-k9	2928	1393	1559	1353	1379	1256	1265	1272	1245	1215	1087	1087	1073
19	A-n60-k9	3741	1738	1961	1647	1731	1707	2082	1603	1516	1546	1393	1393	1354
20	A-n61-k9	3294	1473	1655	1299	1373	1484	1403	1216	1159	1171	1037	1037	1034
21	A-n62-k8	3450	1664	1889	1660	1575	1649	1793	1495	1442	1471	1337	1337	1288
22	A-n63-k10	3673	1763	2024	1733	1522	1746	1729	1523	1474	1513	1332	1332	1314
23	A-n63-k9	4125	2026	2349	2093	1903	2104	2253	1829	1903	1929	1656	1656	1616
24	A-n64-k9	3628	1687	1862	1898	1662	1747	1936	1680	1621	1599	1462	1455	1401
25	A-n65-k9	3646	1605	1706	1477	1538	1575	1593	1387	1417	1370	1244	1244	1174
26	A-n69-k9	4162	1638	1612	1386	1328	1507	1624	1299	1291	1305	1211	1211	1159
27	A-n80-k10	5133	2316	2608	2223	2107	2308	2290	2129	2089	2080	1788	1781	1763
28	B-n31-k5	1438	818	1100	756	758	930	868	700	680	679	675	674	672
29	B-n34-k5	1814	1001	1130	842	841	800	1003	804	790	789	800	800	788
30	B-n35-k5	2672	1209	1273	1265	1006	1094	1311	974	977	980	969	968	955
31	B-n38-k6	2058	1124	1184	1045	913	957	1038	885	832	858	819	818	805
32	B-n39-k5	1849	1012	1125	738	737	672	1123	623	550	557	555	555	549
33	B-n41-k6	2472	1074	1278	1019	928	1001	1176	901	860	855	870	869	829
34	B-n43-k6	2218	985	991	873	858	900	1068	799	754	756	749	749	742
35	B-n44-k7	2502	1145	1436	1174	1112	1111	1231	1025	997	1002	970	970	909
36	B-n45-k5	2594	1136	1097	965	906	1146	1056	828	813	831	754	754	751
37	B-n45-k6	1773	992	1094	966	856	1011	958	829	784	806	708	708	678
38	B-n50-k7	2746	1067	1398	1033	819	982	1143	766	741	752	729	729	741
39	B-n50-k8	2778	1704	1776	1621	1545	1727	1553	1512	1428	1440	1327	1327	1312

Table 1. All implementations Results on All sets

40	B-n51-k7	3003	1391	1462	1237	1143	1160	1311	1163	1099	1109	1110	1109	1032
41	B-n52-k7	2941	1075	1217	1025	913	813	1229	772	758	747	746	746	747
42	B-n56-k7	2807	947	1421	1103	784	909	1124	868	746	750	711	704	707
43	B-n57-k7	4198	1606	1872	1655	1334	1344	1846	1297	1235	1227	1224	1224	1153
44	B-n57-k9	3468	1922	2174	1910	1818	2012	1978	1748	1664	1655	1628	1628	1598
45	B-n63-k10	4076	2243	2385	1818	1751	2073	2058	1658	1603	1598	1572	1572	1496
46	B-n64-k9	2889	1197	1653	1060	1015	1232	1368	1000	936	931	898	898	861
47	B-n66-k9	3278	1652	1894	1578	1502	1636	1847	1434	1476	1474	1403	1402	1316
48	B-n67-k10	3468	1564	1932	1239	1292	1692	1694	1208	1127	1084	1072	1071	1032
49	B-n68-k9	4108	1741	1969	1574	1465	1532	1691	1461	1383	1399	1296	1296	1272
50	B-n78-k10	4265	1642	2144	1557	1624	1925	1900	1466	1355	1363	1230	1230	1221
51	P-n101-k4	2107	935	796	896	800	858	854	767	702	707	700	698	681
52	P-n16-k8	634	461	550	453	471	535	466	453	453	453	475	474	450
53	P-n19-k2	496	229	248	217	231	216	248	217	206	206	232	215	212
54	P-n20-k2	490	253	257	230	241	239	263	214	211	211	227	217	216
55	P-n21-k2	466	253	259	278	215	208	250	208	208	208	230	220	211
56	P-n22-k2	488	255	282	272	212	268	251	212	212	212	232	227	216
57	P-n22-k8	757	636	780	651	727	703	720	685	655	666	585	585	603
58	P-n23-k8	721	728	628	587	571	629	625	588	571	568	531	531	529
59	P-n40-k5	1097	601	555	585	538	601	559	491	472	476	493	487	458
60	P-n45-k5	1293	754	580	666	575	635	607	518	521	528	546	519	510
61	P-n50-k10	1450	954	948	853	837	846	881	777	788	804	719	719	696
62	P-n50-k7	1362	670	713	627	629	658	698	612	587	591	579	559	554
63	P-n50-k8	1339	747	821	770	762	813	752	727	700	698	647	639	631
64	P-n51-k10	1469	922	925	961	893	971	877	887	863	845	767	767	741
65	P-n55-k10	1561	833	930	863	783	895	853	765	760	745	714	703	694
66	P-n55-k15	1617	1226	1294	1258	1089	1181	1109	1086	1064	1061	968	968	989
67	P-n55-k7	1446	727	734	687	662	642	679	618	592	595	602	579	568
68	P-n55-k8	1391	740	741	699	642	770	678	633	608	608	595	576	588
69	P-n60-k10	1741	884	1047	950	883	939	923	830	864	856	760	760	744
70	P-n60-k15	1870	1237	1428	1137	1158	1192	1219	1115	1106	1108	992	990	968
71	P-n65-k10	1866	995	1054	1031	898	954	973	900	906	884	810	810	792
72	P-n70-k10	1983	1037	1068	1056	979	1066	1018	937	960	966	866	849	827
73	P-n76-k4	1988	781	754	761	725	748	850	659	653	621	664	647	593
74	P-n76-k5	2070	889	806	803	741	812	807	746	676	689	682	682	627

Table 2. Percentage gap for all implementations on all sets

Seq	Instance	unhandled_gap	SSR_gap	msr_farthest_gap	msr_highest_demand_gap	msr_weighted_grid_gap	msr_random_gap	msr_nearest_gap	PII-CN_gap	MI-ITS(k=3)_gap	MI-ITS(k=5)_gap	ICW_gap	IPH_gap
1	A-n32-k5	163.65	43.75	53.7	33.8	19.77	18.49	36.86	16.07	9.31	10.71	5.99	0.26
2	A-n33-k5	179.43	48.71	54.31	38.43	8.47	26.02	53.4	7.41	5.45	5.6	3.18	3.18
3	A-n33-k6	104.31	56.47	58.89	46.77	6.06	20.35	21.02	8.76	5.66	6.2	2.43	-0.81
4	A-n34-k5	144.86	34.58	17.48	9.38	13.88	31.11	23.26	4.37	0.51	1.29	1.93	1.93
5	A-n36-k5	147.93	34.29	34.54	38.92	18.52	31.29	45.18	9.89	6.76	5.38	3.13	1.88
6	A-n37-k5	135.87	56.2	32.88	7.03	18.98	12.11	31.24	17.04	6.88	7.62	3.14	3.14
7	A-n37-k6	114.54	31.4	38.46	23.39	22.55	28.45	26.34	19.92	15.28	13.07	3.69	3.37
8	A-n38-k5	205.75	43.97	71.78	18.77	29.59	54.38	34.52	18.08	11.51	16.16	6.58	3.7

Table 2. Percentage gap for all implementations on all sets

9	A-n39-k5	181.14	31.14	38.93	22.02	22.99	28.59	35.4	17.88	11.8	9.12	8.64	6.81
10	A-n39-k6	184.72	36.34	45.97	14.2	23.35	44.52	57.52	20.22	9.51	7.22	3.49	3.01
11	A-n44-k6	175.67	33.3	47.39	13.13	14.73	26.04	26.25	13.66	13.66	11.85	5.76	5.76
12	A-n45-k6	195.87	39.3	60.7	21.5	37.39	32.42	51.91	19.6	16.63	17.16	2.01	2.01
13	A-n45-k7	129.76	25.92	28.27	32.9	15.45	27.57	27.4	19.55	12.57	12.74	3.4	3.4
14	A-n46-k7	192.23	40.37	54.6	32.71	17.83	24.84	44.42	12.8	6.56	9.63	0	0
15	A-n48-k7	166.64	31.03	47.44	42.96	19.94	29.17	39.42	11.09	8.85	10.25	1.77	1.77
16	A-n53-k7	213.56	47.33	60.99	45.25	12.48	19.6	28.71	15.35	4.65	10.3	6.14	5.25
17	A-n54-k7	196.74	28.79	45.59	26.14	18.85	36.25	24.51	13.02	13.2	13.62	2.4	2.4
18	A-n55-k9	172.88	29.82	45.29	26.1	28.52	17.05	17.89	18.55	16.03	13.23	1.3	1.3
19	A-n60-k9	176.29	28.36	44.83	21.64	27.84	26.07	53.77	18.39	11.96	14.18	2.88	2.88
20	A-n61-k9	218.57	42.46	60.06	25.63	32.79	43.52	35.69	17.6	12.09	13.25	0.29	0.29
21	A-n62-k8	167.86	29.19	46.66	28.88	22.28	28.03	39.21	16.07	11.96	14.21	3.8	3.8
22	A-n63-k10	179.53	34.17	54.03	31.89	15.83	32.88	31.58	15.91	12.18	15.14	1.37	1.37
23	A-n63-k9	155.26	25.37	45.36	29.52	17.76	30.2	39.42	13.18	17.76	19.37	2.48	2.48
24	A-n64-k9	158.96	20.41	32.91	35.47	18.63	24.7	38.19	19.91	15.7	14.13	4.35	3.85
25	A-n65-k9	210.56	36.71	45.32	25.81	31.01	34.16	35.69	18.14	20.7	16.7	5.96	5.96
26	A-n69-k9	259.1	41.33	39.09	19.59	14.58	30.03	40.12	12.08	11.39	12.6	4.49	4.49
27	A-n80-k10	191.15	31.37	47.93	26.09	19.51	30.91	29.89	20.76	18.49	17.98	1.42	1.02
28	B-n31-k5	113.99	21.73	63.69	12.5	12.8	38.39	29.17	4.17	1.19	1.04	0.45	0.3
29	B-n34-k5	130.2	27.03	43.4	6.85	6.73	1.52	27.28	2.03	0.25	0.13	1.52	1.52
30	B-n35-k5	179.79	26.6	33.3	32.46	5.34	14.55	37.28	1.99	2.3	2.62	1.47	1.36
31	B-n38-k6	155.65	39.63	47.08	29.81	13.42	18.88	28.94	9.94	3.35	6.58	1.74	1.61
32	B-n39-k5	236.79	84.34	104.9	34.43	34.24	22.4	104.6	13.48	0.18	1.46	1.09	1.09
33	B-n41-k6	198.19	29.55	54.16	22.92	11.94	20.75	41.86	8.69	3.74	3.14	4.95	4.83
34	B-n43-k6	198.92	32.75	33.56	17.65	15.63	21.29	43.94	7.68	1.62	1.89	0.94	0.94
35	B-n44-k7	175.25	25.96	57.98	29.15	22.33	22.22	35.42	12.76	9.68	10.23	6.71	6.71
36	B-n45-k5	245.41	51.26	46.07	28.5	20.64	52.6	40.61	10.25	8.26	10.65	0.4	0.4
37	B-n45-k6	161.5	46.31	61.36	42.48	26.25	49.12	41.3	22.27	15.63	18.88	4.42	4.42
38	B-n50-k7	270.58	43.99	88.66	39.41	10.53	32.52	54.25	3.37	0	1.48	-1.6	-1.62
39	B-n50-k8	111.74	29.88	35.37	23.55	17.76	31.63	18.37	15.24	8.84	9.76	1.14	1.14
40	B-n51-k7	190.99	34.79	41.67	19.86	10.76	12.4	27.03	12.69	6.49	7.46	7.56	7.46
41	B-n52-k7	293.71	43.91	62.92	37.22	22.22	8.84	64.52	3.35	1.47	0	-0.1	-0.13
42	B-n56-k7	297.03	33.95	101	56.01	10.89	28.57	58.98	22.77	5.52	6.08	0.57	-0.42
43	B-n57-k7	264.09	39.29	62.36	43.54	15.7	16.57	60.1	12.49	7.11	6.42	6.16	6.16
44	B-n57-k9	117.02	20.28	36.05	19.52	13.77	25.91	23.78	9.39	4.13	3.57	1.88	1.88
45	B-n63-k10	172.46	49.93	59.43	21.52	17.05	38.57	37.57	10.83	7.15	6.82	5.08	5.08
46	B-n64-k9	235.54	39.02	91.99	23.11	17.89	43.09	58.89	16.14	8.71	8.13	4.3	4.3
47	B-n66-k9	149.09	25.53	43.92	19.91	14.13	24.32	40.35	8.97	12.16	12.01	6.61	6.53
48	B-n67-k10	236.05	51.55	87.21	20.06	25.19	63.95	64.15	17.05	9.21	5.04	3.88	3.78
49	B-n68-k9	222.96	36.87	54.8	23.74	15.17	20.44	32.94	14.86	8.73	9.98	1.89	1.89
50	B-n78-k10	249.3	34.48	75.59	27.52	33.01	57.66	55.61	20.07	10.97	11.63	0.74	0.74
51	P-n101-k4	209.4	37.3	16.89	31.57	17.47	25.99	25.4	12.63	3.08	3.82	2.79	2.5
52	P-n16-k8	40.89	2.44	22.22	0.67	4.67	18.89	3.56	0.67	0.67	0.67	5.56	5.33
53	P-n19-k2	133.96	8.02	16.98	2.36	8.96	1.89	16.98	2.36	-2.83	-2.83	9.43	1.42
54	P-n20-k2	126.85	17.13	18.98	6.48	11.57	10.65	21.76	-0.93	-2.31	-2.31	5.09	0.46
55	P-n21-k2	120.85	19.91	22.75	31.75	1.9	-1.42	18.48	-1.42	-1.42	-1.42	9	4.27
56	P-n22-k2	125.93	18.06	30.56	25.93	-1.85	24.07	16.2	-1.85	-1.85	-1.85	7.41	5.09
57	P-n22-k8	25.54	5.47	29.35	7.96	20.56	16.58	19.4	13.6	8.62	10.45	-3	-2.99
58	P-n23-k8	36.29	37.62	18.71	10.96	7.94	18.9	18.15	11.15	7.94	7.37	0.38	0.38
59	P-n40-k5	139.52	31.22	21.18	27.73	17.47	31.22	22.05	7.21	3.06	3.93	7.64	6.33
60	P-n45-k5	153.53	47.84	13.73	30.59	12.75	24.51	19.02	1.57	2.16	3.53	7.06	1.76

Table 2. Percentage gap for all implementations on all sets

61	P-n50-k10	108.33	37.07	36.21	22.56	20.26	21.55	26.58	11.64	13.22	15.52	3.3	3.3
62	P-n50-k7	145.85	20.94	28.7	13.18	13.54	18.77	25.99	10.47	5.96	6.68	4.51	0.9
63	P-n50-k8	112.2	18.38	30.11	22.03	20.76	28.84	19.18	15.21	10.94	10.62	2.54	1.27
64	P-n51-k10	98.25	24.43	24.83	29.69	20.51	31.04	18.35	19.7	16.46	14.04	3.51	3.51
65	P-n55-k10	124.93	20.03	34.01	24.35	12.82	28.96	22.91	10.23	9.51	7.35	2.88	1.3
66	P-n55-k15	63.5	23.96	30.84	27.2	10.11	19.41	12.13	9.81	7.58	7.28	-2.1	-2.12
67	P-n55-k7	154.58	27.99	29.23	20.95	16.55	13.03	19.54	8.8	4.23	4.75	5.99	1.94
68	P-n55-k8	136.56	25.85	26.02	18.88	9.18	30.95	15.31	7.65	3.4	3.4	1.19	-2.04
69	P-n60-k10	134.01	18.82	40.73	27.69	18.68	26.21	24.06	11.56	16.13	15.05	2.15	2.15
70	P-n60-k15	93.18	27.79	47.52	17.46	19.63	23.14	25.93	15.19	14.26	14.46	2.48	2.27
71	P-n65-k10	135.61	25.63	33.08	30.18	13.38	20.45	22.85	13.64	14.39	11.62	2.27	2.27
72	P-n70-k10	139.78	25.39	29.14	27.69	18.38	28.9	23.1	13.3	16.08	16.81	4.72	2.66
73	P-n76-k4	235.24	31.7	27.15	28.33	22.26	26.14	43.34	11.13	10.12	4.72	12	9.11
74	P-n76-k5	230.14	41.79	28.55	28.07	18.18	29.51	28.71	18.98	7.81	9.89	8.77	8.77

Appendix B

Common Parameter list

depot	Depot's index	capacity	Vehicle Maximum capacity
demands	Customers' demand list	dm	Distance matrix among nodes

Algorithm 1: WSI-GS

Method: WSI

Input: (depot, demands, capacity, dm, distance_weight, demand_weight)
distance_weight: Distance from depot weight in the weighted score
demand_weight: Demand weight in the weighted score

Output:
routes: routes spanning all nodes without exceeding capacity

Algorithm

1. **Variable Initialization:**
   - num_customers = demands' length; routes = empty list to store routes (solution); visited = Set {depot} to track node visits; num_vehicles = ceil(sum(demands) / capacity)
2. **Calculate weighted scores for Nodes**
   - For each node I in [from 1 to (num_customers_1)]:
     Score(i) = dm[depot][i] * distance_weight +demand_weight * demands[i]
   - Choose (num_vehicles) nodes with highest scores in chosen_nodes
3. **Initialize Routes with Highest_score Nodes**
   - For each node i in chosen_nodes
     Add i to visited.
     Create a route [depot,i,depot] and add it to routes.
4. **Insert Nodes Sequentially**
   - While the number of visited_nodes < num_customers:
     Set best_cost_increase = infinity.
     Set best_node, best_position, and best_route = None.
5. **Find best Node for insertion:**
   - For each unvisited node j (from 1 to (num_customers-1)):
     For each r in routes:
     If adding j to the route doesn't exceed the vehicle capacity:

For each (pos) possible insertion position in r:
Calculate cost_increase:
dm[pos-1][j]+dm[j][pos+1] -dm[pos-1][pos+1]
If cost_increase < best_cost_increase:
Update best_cost_increase, best_node, best_position, and best_route.

6. **Insert the node or start a new route**
   - If best_node exists:
     Insert best_node in best_position into best_route.
     Append best_node to visited.
   - Else:
     Start a new route with unvisited node that suitable the capacity:
     [depot, j,depot] for node j
     Append it to routes and mark it as visited
7. **Return the Final Routes:**
   - Return routes

Method: GS

Input: (depot, demands, capacity, dm, distance_weight_options, demand_weight_options)
distance_weight_options: Possible Weights for distance_weight
demand_weight_options: possible weights for demand_weight

Output:
Best_sol: best routes with minimum cost.
Best_weight_combination: Optimal tuple of distance and demand weights

Grid Search Optimization Method

1. **Initialize results variables**
   - best_weight_combination = None; best_cost = Infinity; best_sol = None.
2. **Perform Grid search:**
   - For each pair of (distance_weight, demand_weight) from the Cartesian product of distance_weight_options and demand_weight_options:
     Call WSI using (depot,demands,capacity,dm,distance_weight and demand_weight)
     Store the resulting routes.
3. **Evaluate routes cost:**
   - Calculate total_cost for the generated routes in total_cost
4. **Update the best solution**
   - If total_cost < best_cost:
     Update best_cost = total_cost
     Update best_weight_combination = Tuple (distance_weight, demand_weight)
     Update best_sol= routes
5. **Return the best solution:**
   - Return best_sol

Algorithm2: MI-ITS

Method: Calling Method

Input: (depot, demands, capacity, dm, vehicles_num, max_iteration, k**)**

- vehicles_num: Vehicles Number
- max_iteration: Maximum iterations
- k: tournament sample size

Output:

- best_solution: lowest cost solution from returning solutions from worker method

Algorithm

1. **Variable Initialization:**
   - solutions = empty list to store worker returning solutions.
2. **Call Worker Method by attempting all nodes as first route start node**

- For each node [from 1 to (num_customers_1)]:
  Call worker method (node, depot, demands, capacity, dm, vehicles_num, max_iteration, k).
- Add the returned solution to solutions.

3. **Filter out invalid solutions from solutions**
   - Discard empty and any solution with greater than number of available vehcles
4. **Return best Solution**
   - Select the lowest code solution from solutions
     Best_solution = min(solutions, key=solution cost).

Method: Caller Method

Input: (start_node, depot, demands, capacity, dm, vehicles_num, max_iteration, k)

- start_node: start node index to be the start node for the first route in the solution

Output:

- routes: Solution

1. **Variables Initialization:**
   - num_customers = demands' length; routes = empty list to store routes (solution); append [depot, start_node, depot] to routes; visited = Set {depot,start_node} to track node visits; num_vehicles = ceil(sum(demands) / capacity); chosen_nodes: Use modulo logic to choose consecutive nodes to the start_node according to vehicles number.
   - Initialize additional routes
     for node in chosen_nodes:
     append [depot,node,depot] to routes list.
   - Create an solutions_data as empty dictionary for solutions data.
2. **Extensive investigation of the search space**
   - Choose different candidates nodes for each run
     For each iteration in range(max_iteration):
     Reset routes with solely initial nodes.
     Reset visited nodes to only include depot and starting nodes.
     For each node not in visited:
     Determine all candidate nodes in visited
     Calculate best cost increase for each candidate node
     Use tournament selection to select best_node from a random sample from candidate nodes
     Insert best_node in best position in best route
     best_route=None; best_position=None; best_cost_increase= infinity;
     for r in routes:
     for position in r positions:
     calculate cheapest_insertion_cost cost_increase
     if cost_increase < best_cost_increase:
     best_cost_increase = cost_increase; best_position= position;
     best_route = r
     insert best_node into r in best_position
     store the iteration solution and cost in solutions_data
3. **Return the best_solution from all iterations solution**
   - Best_solution=Min(solutions_data, key=solution cost)

Method: Tournament

Input:

- Candidates: nodes indexes with their best_cost_increase
- k: tournament size: random sample

output:

- best_node: lowest best_cost_increase node index

1. **check the size of the sample**
   - check if the number of candidates is small

if candidates_number < k:
set k= candidates_number

2. **random selection:** sample =choose k candidates
3. **select best candidate**
   - return best_node = min(sample, k=best_cost_increase)