Journal of Mathematical Modeling
Vol. , No. , 2026, pp. -. Research Article

JMM

A proficient and optimal local search algorithm for the establishment and operation of single-allocation hub-and-spoke transportation networks

Mona Alizadeh Firozi[†], Vahid Kiani[‡*], Hossein Karimi[†]

[†]*Department of Industrial Engineering, Faculty of Engineering, University of Bojnord, Bojnord, Iran*
[‡]*Department of Computer Engineering, Faculty of Engineering, University of Bojnord, Bojnord, Iran*

Email(s): monaalizadeh29@gmail.com, v.kiani@ub.ac.ir, h.karimi@ub.ac.ir

Abstract. Hub-and-spoke networks are vital in transportation planning for cost-effective transport. The hub-location problem (HLP) optimizes intermediary nodes, their locations, and spoke assignments to minimize network costs. This paper addresses a special variant of HLP called the uncapacitated single allocation hub-location problem (USAHLP). A meta-heuristic algorithm, the multi-start local search (MSLS), is proposed to reliably solve large instances. Given the NP-hardness of USAHLP, MSLS uses a two-level local search, with the second level focusing on the best solutions from the first. Enhancements include three hub location structures, a non-hub reallocation operator, and greedy random restarts. These features ensure MSLS achieves near-optimal solutions and high reliability. The algorithm's performance is compared against CPLEX, GVNS, and eight state-of-the-art algorithms. Computational experiments conducted on USAHLP benchmark instances from the AP dataset demonstrate that the proposed MSLS consistently achieves superior solution quality and high reliability in a tolerable time-frame compared to existing approaches.

Keywords: Transportation networks, hub location problem, optimization, meta-heuristics.
AMS Subject Classification 2010: 90B06, 90B80, 90C59.

1 Introduction

Hub-and-spoke networks are common and cost-effective means of transporting goods, people, and other shipments [23]. In hub networks, direct connections between all nodes of the transportation network are replaced with fewer indirect connections [14]. Hub networks route traffic through special nodes called hubs rather than sending data or commodities directly between the nodes. This type of transportation

*Corresponding author
Received: 07 September 2025/ Revised: 28 April 2026/ Accepted: 02 May 2026
DOI: 10.22124/jmm.2026.31635.2848

network results in lower total transportation and network establishment costs than a fully connected network.

The hub location problem (HLP) involves determining the optimal locations for hub facilities within the transportation network to facilitate the efficient distribution of goods from suppliers to customers [8, 31]. In HLP, hubs act as intermediate points where cargoes can be consolidated, sorted, and redistributed to their final destinations. By strategically locating hubs, companies can minimize transportation costs, reduce delivery lead times, and improve overall supply chain efficiency. To solve HLP, the number of hub nodes must be determined, hub nodes should be selected, and non-hubs should be assigned to the hubs. Thus, the HLP can be divided into two components: location and allocation [14]. The optimization of hub locations in HLPs is primarily aimed at achieving cost savings. However, it is worth noting that additional factors such as the improvement of service levels, the enhancement of network resilience, the address of environmental concerns, and the support of business growth can also be taken into account [4, 24, 26–28, 30].

The uncapacitated single allocation HLP (USAHLP) is an HLP characterized by assigning each non-hub node to a single hub, with hub nodes having no traffic-related restrictions on inbound and outbound flows [8]. The primary objective of this problem is to minimize the overall network cost by strategically selecting optimal hub nodes and allocating non-hub nodes to them. The USAHLP is classified as an NP-hard problem [35]. This difficulty arises because hub selection and single-allocation decisions must be determined simultaneously, leading to a combinatorial growth in the number of feasible hub-allocation configurations with the number of nodes. Therefore, while exact optimization methods can provide benchmark-quality solutions for small and medium-sized instances, their computational burden increases rapidly as the problem size grows. As a result, the development of efficient meta-heuristic algorithms to obtain high-quality solutions within reasonable computational times is essential for large-scale instances [11, 33, 34]. Consequently, the primary objective of this research endeavor is to develop an efficient methodology to obtain optimal or near-optimal solutions for USAHLP within a reasonable timeframe.

Various methodologies have been developed to tackle USAHLP, including heuristic algorithms, meta-heuristics, and exact methods [8]. Previous research efforts demonstrate that a clever combination of meta-heuristics and suitable search operators can yield robust search algorithms that are proficient in efficiently and accurately resolving large-scale instances of USAHLP. The design of such a meta-heuristic method requires careful consideration. This includes selecting appropriate components such as solution representation, search operators, and optimization techniques that align with the problem characteristics. It is important to strike a balance between exploration (diversification) and exploitation (intensification) during the search process. The algorithm should effectively explore the search space to avoid getting trapped in local optima while also intensifying the search around promising solutions.

In the present study, we propose a groundbreaking methodology to effectively tackle the challenges of reliability and speed in resolving USAHLP. We introduce a multi-start local search algorithm (MSLS) that aims to enhance the effectiveness and efficiency of solving USAHLP by employing a multi-start framework coupled with a comprehensive set of local search operators. The framework adopts a twofold approach to the neighborhood search called the two-level local search procedure (TLSP). The TLSP involves a local search procedure in which only promising solutions from the first level undergo further examination at the second level. Consequently, this strategy leads to a good balance between exploration and exploitation, and substantial time savings. Furthermore, the proposed MSLS method demonstrates the ability to achieve optimal solutions consistently across all executions, even when confronted with

large-scale sample problems. This achievement can be attributed to the utilization of diverse neighborhood structures during the local search process and the incorporation of random restarts following local search failures. The primary objective of this article is to detail the design and implementation of our proposed MSLS, highlighting the innovative aspects of the algorithm and its potential advantages over existing methods.

The MSLS framework proposed in our research article makes several key contributions to the field. These contributions can be summarized as follows:

1. Novel approach: Our framework introduces a novel approach for addressing USAHLP. By combining multi-start and local search techniques, we provide a fresh perspective on solving this problem.

2. Greedy random restart strategy: The incorporation of a greedy random restart strategy in our algorithm enhances the exploration of the solution space. This strategy allows for diversification by restarting the search process from different initial solutions, thereby increasing the chances of finding better-quality solutions.

3. Diverse set of local search operators: We employ a diverse set of local search operators in our MSLS framework. By incorporating a range of operator types, such as add-hub, remove-hub, interchange-hub, and reallocation operators, our algorithm can explore and exploit different neighborhoods.

4. Enhanced solution quality: Through extensive experimentation and evaluation, we demonstrate that our MSLS framework yields high-quality solutions for USAHLP instances.

The rest of the paper is organized as follows: Section 2 offers an in-depth review of state-of-the-art techniques and algorithms that have been applied to address USAHLP. Section 3 presents a detailed overview of USAHLP, its mathematical formulation, and the nature of its cost surface. In Section 4, we elaborate on the construction and functioning of our novel MSLS, emphasizing the incorporation of the greedy random restart strategy and various local search operators. Section 5 presents an extensive experimental evaluation of the proposed MSLS, comparing its performance against other established methods on the benchmark instances. Finally, Section 6 concludes the article with a summary of our findings and outlines potential avenues for future research in this domain.

2 Literature review

Over the years, a wide range of algorithms and methodologies have been proposed to tackle USAHLP, to obtain near-optimal solutions capable of efficiently handling large-scale instances within reasonable computational times. The existing literature can be broadly classified into four major categories that have significantly shaped the evolution of this problem: exact methods, heuristic approaches, meta-heuristic techniques, and hybrid frameworks. Each of these methodological directions contributes differently to improving solution quality, computational efficiency, and adaptability to various problem settings.

2.1 Exact methods

Exact approaches for USAHLP aim to determine globally optimal solutions by systematically exploring the solution space through rigorous mathematical programming techniques. Classical strategies—such

as integer programming formulations, branch-and-bound, cutting plane, and Benders decomposition algorithms—provide strong theoretical guarantees of optimality. However, their computational cost increases exponentially with instance size, which restricts their applicability to large-scale problems. One of the earliest contributions in this area was presented by Pirkul et al. [37], who developed a tight linear programming formulation and employed subgradient optimization within a Lagrangian relaxation framework to handle the single allocation hub-and-spoke network efficiently. By augmenting the Lagrangian subproblem with additional cutting constraints, they improved the convergence properties and the quality of lower bounds. Building on decomposition techniques, Ghaffarinasab and Kara [18] introduced an advanced Benders decomposition-based algorithm specifically tailored for large instances. Their approach incorporated several algorithmic refinements, including a novel cut disaggregation scheme, generation of strong optimality cuts, and an efficient dual subproblem solver. Their method successfully solved test instances containing up to 250 nodes. In parallel, Espejo et al. [13] proposed a compact mixed-integer programming formulation requiring fewer variables than previous models, thereby enhancing computational tractability. Their branch-and-cut algorithm operated on a relaxed version of the formulation and dynamically introduced constraints through a cut generation procedure comprising two classes of valid inequalities. This enabled them to solve benchmark instances of up to 200 nodes within reasonable computational times. Overall, exact algorithms serve as essential references for validating heuristic and meta-heuristic frameworks, providing benchmark-quality solutions that define the lower bounds of USAHLP. Nevertheless, their substantial computational requirements and limited scalability have driven research toward meta-heuristic, hybrid, and large-scale optimization techniques, which can deliver high-quality solutions within acceptable computational times.

2.2 Heuristic approaches

In contrast to the exact methods, heuristic algorithms for USAHLP prioritize computational efficiency, aiming to derive high-quality, near-optimal solutions within practical time limits, albeit without a guarantee of global optimality [24, 40, 41]. Research specifically dedicated to purely heuristic methods for USAHLP has been relatively scarce. A foundational contribution in this domain was made by O'Kelly, who pioneered two distinct heuristic methods designed for rapid solution generation [35]. His first heuristic employed a nearest-hub allocation rule, where hub locations are determined by enumerating all potential locational configurations, followed by assigning every non-hub node to its closest designated hub. The second heuristic refined this by considering the allocation to the second-nearest hub as well. These procedures were initially validated using the Civil Aeronautics Board (CAB) dataset, which comprises inter-city traffic data for 25 metropolitan areas. Subsequent research, benefiting from the rapid advancement of computational intelligence, has seen a notable shift, with a significant concentration on employing meta-heuristic algorithms to tackle the complexities of USAHLP more effectively than traditional heuristics.

2.3 Meta-heuristic techniques

Meta-heuristic algorithms represent a vital class of optimization techniques, offering a powerful mechanism to effectively navigate the solution landscape of USAHLP, escape local optima, and achieve near-optimal solutions in computationally feasible timeframes [31]. Unlike problem-specific heuristics, meta-heuristics provide a general framework whose efficacy rests on striking an optimal balance

between solution quality and computational overhead. Early meta-heuristic applications include the work by Topcuoglu et al., who tailored a Genetic algorithm (GA) to address both the hub location and allocation phases of the USAHP [43], employing crossover for hub selection and shifting/exchanging procedures for node assignment. Focusing on local search enhancement, Silva and Cunha introduced multiple variants of the multi-start tabu search (MSTS) heuristic, alongside a straightforward two-stage HubTS approach [42]. Their empirical results indicated that HubTS delivered superior solution costs, while MSTS-3 offered advantages in terms of processing time. More sophisticated exploration capabilities were introduced by Pingle et al., who integrated six local search operators and the novel "Solution Pair" concept into an ant colony optimization (ACO) framework to better align with the problem's structure [36]. Further significant advancements in scalability were achieved through novel tabu search (TS) strategies; Abyazi-Sani and Ghanbari developed specialized rules and a rapid objective function estimation technique, enabling them to solve instances up to 400 nodes from the Australian post (AP) dataset [2]. Similarly, Guan et al. pioneered a probabilistic TS incorporating randomized greedy constructions and robust exploitation procedures, successfully tackling instances as large as 900 nodes [21]. Guan et al. also proposed a Multi-start Iterated Local Search utilizing a perturbation mechanism for diversification, achieving best-known solutions for 53 out of 56 instances tested [22]. In related research, Grine et al. applied a clonal selection algorithm (CSA), an Artificial Immune System variant, to the closely related p-Hub Median Problem, demonstrating superior solution quality across major benchmarks [20].

2.4 Hybrid frameworks

Hybrid algorithms represent a synergistic synthesis of diverse optimization paradigms, combining the inherent strengths of various methodologies—such as heuristics, meta-heuristics, or exact methods—to simultaneously boost solution quality and improve computational efficiency. The academic focus in this area is keenly directed toward identifying optimal fusion strategies and rigorously assessing their impact on USAHLP. A seminal example of this integration is the GATS algorithm proposed by Abdinnour-Helm, which sequentially joined a GA for location decisions with a TS for allocation tasks [1]. While this sequential hybridization accelerated the solution process compared to a standalone GA, the authors noted a resultant reduction in the overall exploratory power of the search. More integrated hybridization is demonstrated by the memetic algorithm of Maric et al. [29]. This approach embeds two powerful local search heuristics directly within the framework of an evolutionary algorithm to refine both the location and allocation components concurrently. Applied to extensive datasets, including CAB, AP, and a modified AP set, their method established the first-ever optimal solutions for 30 AP and modified AP instances, significantly outperforming previous heuristic benchmarks, even on problems involving up to 900 nodes. Furthermore, hybrid methods have proven critical in related, complex contexts. More recently, Alizadeh-Firozi et al. refined the GA framework by integrating specialized hub and allocation mutation operators alongside intensive local search exploitation around the best-found solution [3]. This enhanced exploration-exploitation balance resulted in the discovery of optimal solutions for numerous large-scale USAHLP instances.

2.5 Emerging variants and applications

Building upon the established USAHLP, research has significantly evolved to address complex operational requirements and realities, necessitating specialized modeling techniques [4, 19]. Bryan and

O'Kelly highlighted the need for reliable heuristics driven by the introduction of variants like multiple assignment HLPs [7]. Cvokic and Stanimirovic introduced a profit-maximization variant incorporating price-dependent demand and endogenous hub count, reformulating the deterministic case as a mixed-integer linear program (MILP) and the robust counterpart as a mixed-integer conic-quadratic program (MIQCP) to ensure immunity to demand perturbations [10]. In the context of disruptions, Momayezi et al. modeled the capacitated modular single allocation HLP (CMSAHLP) under hub failure scenarios using a two-stage stochastic program [32]. Furthermore, Kemmar et al. proposed the runaway p-HLP (RpHLP), integrating a round-trip structure and introducing 'runaway nodes' for redundancy, modeled via an MIP [25]. Other significant model generalizations include incorporating heterogeneous economies of scale solved via Benders-type decompositions [38], modeling flow delay costs using particle swarm optimization [44], and adapting the problem for Short Sea Shipping with multi-objective optimization [15].

The complexity introduced by these advanced models necessitates sophisticated solution methodologies, often integrating exact methods with meta-heuristics. Addressing sustainability, Gargouri et al. employed four meta-heuristics (GA, SA, PSO, VDO) to solve an NP-hard, three-echelon collaborative network under mixed integer linear problem formulation, balancing economic, environmental (CO2 emissions), and social objectives [17]. For uncertainty, Zhang et al. utilized the NSGA-II algorithm with local reinforcement to solve a multimodal hub network under fuzzy chance constraints regarding capacity and customer satisfaction [46]. In operational planning, Yu et al. designed a two-level approximation algorithm to maximize airline profit utility in a multiple-allocation setting [45], while Cuzzocrea et al. offered heuristic collections to rapidly approximate solutions for hub networks under demand uncertainty [9]. Finally, Sachan proposed a novel unit transportation cost function for the SAHLP, validated using an anti-predatory nature-inspired algorithm, emphasizing the dependency of network configuration on factors beyond simple supply and demand [39].

2.6 Summary and research gap

The cumulative body of literature confirms that strategic hybridization of meta-heuristics with carefully designed search operators yields highly potent algorithms, capable of solving large-scale USAHLP instances with superior speed and accuracy. Nevertheless, achieving an optimal trade-off between solution quality and computational tractability remains a persistent challenge in the field. In response to these identified needs for enhanced exploration, exploitation, and robustness, this research proposes a novel meta-heuristic approach to rigorously address USAHLP. Specifically, we introduce an advanced MSLS algorithm, characterized by a greedy random restart strategy, a diverse collection of local search operators, and the novel TLSP.

3 Problem statement

The USAHLP is a decision problem commonly encountered within the field of operational research. It encompasses two distinct phases: hub selection and node allocation. The former involves identifying suitable hub nodes from all available nodes, while the latter pertains to assigning non-hub nodes to their respective hubs [14]. In the context of USAHLP, each non-hub node must be linked to a single hub node, without any restrictions on traffic inflow or outflow. The primary objective of this problem is to determine the optimal hub nodes and allocate non-hubs to these hubs to minimize the overall cost of the

transportation network. This cost encompasses both the fixed expenses associated with establishing hub nodes and the variable costs associated with commodity transmission.

3.1 Formulation of USAHLP

Consider a given set of nodes, denoted as $N = \{1, 2, \ldots, n\}$, which represents a transportation network. Within this network, the variable w_{ij} is employed to represent the quantity of traffic that needs to be transferred from node i to node j for each pair of nodes (i, j). Furthermore, the distance between any two nodes (i, j) is denoted by d_{ij}. It is possible to select certain nodes, ranging from 1 to n, as hubs. These selected hubs will serve as central locations and incur an establishment cost of F_k at a candidate location k. The decision variable z_{ij} is introduced to indicate the selection of hub nodes and assign non-hub nodes to these hubs. Specifically, if non-hub node i is assigned to hub node j, the value of the decision variable $z_{ij} \in \{0, 1\}$ will be one; otherwise, it will be zero. Moreover, when hub node k is chosen, the decision variable z_{kk} has a value of 1, signifying that node k has been selected as a hub.

The USAHLP aims to minimize the overall transportation cost by reducing both fixed and variable costs. Fixed costs are related to the establishment of hubs, while variable costs consist of three components: collection costs from non-hubs, distribution costs to non-hubs, and transfer costs between hubs. Different per-unit cost factors are considered for each stage of collection, transfer, and distribution, as inter-hub links and non-hub links exhibit varying performance. These cost factors are denoted by α, γ, and δ for collection, transfer, and distribution, respectively. Therefore, USAHLP is modeled as [35]:

$$\textit{Minimize} \quad \sum_{i\in N}\sum_{k\in N}\sum_{l\in N}\sum_{j\in N} w_{ij}(\alpha d_{ik} z_{ik} + \gamma d_{kl} z_{ik} z_{jl} + \delta d_{jl} z_{jl}) + \sum_{k\in N} F_k z_{kk} \tag{1}$$

$$s.t. \quad \sum_{k\in N} z_{ik} = 1, \quad \forall i \in N \tag{2}$$

$$z_{ik} \leqslant z_{kk}, \quad \forall i, k \in N \tag{3}$$

$$z_{ik} \in \{0, 1\}, \quad \forall i, k \in N \tag{4}$$

The objective function (1) minimizes the total cost of the transportation network, where the first term considers the variable costs, and the second term computes the fixed costs. According to constraints (2) and (4), it is not permissible for any non-hub node to be assigned to multiple hub nodes. Additionally, due to constraint (3), all nodes are prohibited from being allocated to non-hubs.

3.2 Complexity of USAHLP

In addition to the NP-hardness statement cited in the introduction, the computational difficulty of USAHLP can also be understood directly from the structure of its decision space. For a network with n nodes, the model must determine both which nodes are opened as hubs and how each remaining node is singly assigned to one selected hub. Even before evaluating routing costs, the number of candidate hub subsets grows exponentially with n. Moreover, for each selected hub set, the non-hub nodes can be assigned in many different ways, which leads to a combinatorial explosion in the number of feasible solutions.

Proposition 1. *The USAHLP is NP-hard.*

Proof. We establish the claim by a polynomial-time reduction from the metric uncapacitated facility location problem (UFL), which is NP-hard [16]. Consider a UFL instance on node set V, with facility-opening costs f_k, customer demands q_i, and assignment costs c_{ik}. Construct a USAHLP instance on $N = V \cup \{t\}$, where t is a dummy node. Set $F_k = f_k$ for all $k \in V$, set $F_t = M$ for a sufficiently large constant M, choose $\alpha = 1$ and $\gamma = \delta = 0$, define $w_{it} = q_i$ for all $i \in V$ and $w_{ij} = 0$ otherwise, and let $d_{ik} = c_{ik}$ for all $i,k \in V$. Under this construction, the inter-hub and distribution terms vanish, and the USAHLP objective reduces exactly to the UFL objective: opening a hub corresponds to opening a facility, and assigning a node to a hub corresponds to assigning a customer to a facility. Because F_t is prohibitively large, the dummy node is never opened in an optimal solution. Hence, any UFL solution of cost at most B yields a USAHLP solution of cost at most B, and conversely, any USAHLP solution of cost at most B induces a UFL solution of the same cost. The transformation is polynomial, so USAHLP is NP-hard. This complexity-driven motivation for meta-heuristic design is also common in related optimization studies [11, 33, 34]. □

3.3 Visualization of cost surface

Solving an HLP can be conceptualized as a search procedure aimed at identifying the most favorable solution within the solution space. The attainment of an optimal solution entails determining the ideal quantity of hub nodes, selecting an optimal set of hub nodes, and subsequently optimally allocating non-hub nodes to the chosen hub nodes. To facilitate this process, researchers can devise suitable search strategies during the location and allocation phases through the application of visualizations and analyses of the geometric attributes of the cost surface.

According to the USAHLP framework, an accurate determination of the optimal allocation cannot be achieved unless the optimal set of hub nodes is selected. To gain a deeper comprehension of the challenges associated with the location component of USAHLP, an assessment of cost values was conducted for various candidate hub sets in a specific problem scenario involving 40 nodes. It was assumed that the ideal number of hubs for this particular problem, referred to as 40L in the AP problem set, is already known to be two nodes [12]. Consequently, the cost value was calculated for all potential pairs of hubs. A non-smooth surface with numerous local maxima and minima is obtained for this instance, as shown in Figure 1. According to this result, the search space encompasses a distribution of local minima that are separated by saddle points and local maxima. The presence of local minima may impede neighborhood search algorithms from attaining optimal solutions. To mitigate this issue, we aim to employ a greedy random restart strategy, alongside a comprehensive range of hub modification operators, to decrease the likelihood of the search algorithm becoming trapped in local minima.

4 Proposed method

In the following section, a thorough discussion of the constituent elements of the MSLS approach and their interrelationships will be presented. First, an outline of the general framework of MSLS is provided. This is followed by an explanation of the greedy randomized solution construction procedure (GRSCP), which is employed to generate the initial solution and execute random restarts. Subsequently, a detailed description of TLSP is presented to investigate the neighborhood.

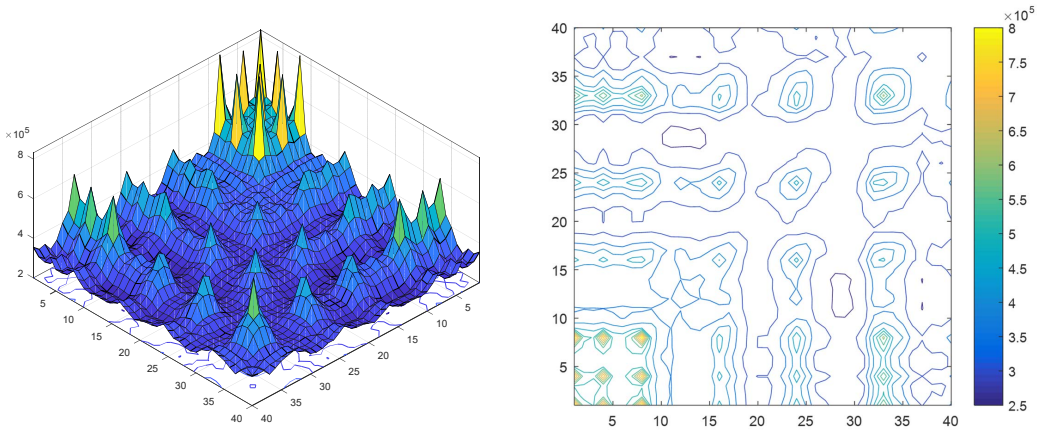


Figure 1: Visualization of cost surface for the location part of USAHLP

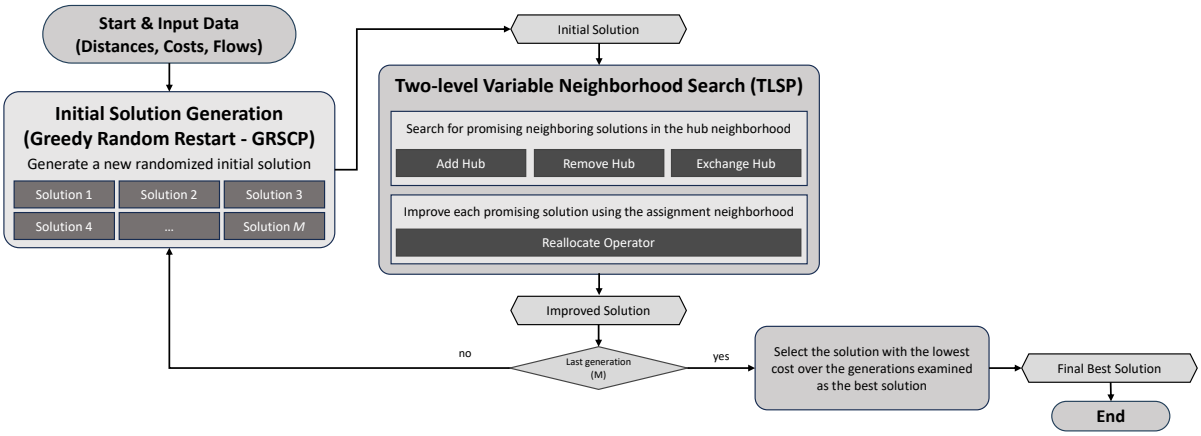


Figure 2: General structure of the proposed MSLS

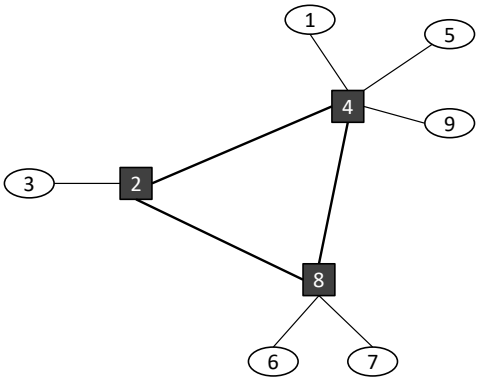
4.1 General structure of MSLS

The proposed MSLS is a modified local search algorithm that exploits an enhanced neighborhood search approach in its local search step. The MSLS, as shown in Figure 2, is a general search procedure consisting of multiple iterations. In each iteration, the GRSCP is employed to generate an initial feasible solution and restart the search process. Subsequently, a two-level local search is performed, iteratively exploring the neighborhood of the current solution to find an improved solution. This greedy walking process terminates when no further enhancements can be achieved. Upon termination of the local search, the next iteration begins by restarting the search procedure from a newly generated random solution. In this procedure, a predefined number of iterations is performed, and multiple search generations are investigated. Ultimately, the most optimal solution discovered during the entire search process is returned as the outcome. The overall procedure of the proposed MSLS is presented in Algorithm 1.

In the proposed MSLS, we exclusively generate solutions that are considered feasible. Moreover, we ensure that the feasibility of each solution remains intact even when manipulated. Through the

Algorithm 1: The MSLS

```
iteration ← 1
f* ← ∞
while iteration < M do
    construct random feasible solution: S ← ConstructRandomSolution()
    perform local search around current solution: S ← LocalSearch(S)
    if f(S) < f* then
        f* ← f(S)
        S* ← S
    end if
end while
return S
```



Solution Representation	1	2	3	4	5	6	7	8	9
Hub Vector	0	1	0	1	0	0	0	1	0
Assignment Vector	4	2	2	4	4	8	8	8	4

Figure 3: A sample transportation network and its corresponding solution representation

implementation of the GRSCP, an initial solution is generated in a randomized manner by selecting the first hub node at random, followed by greedy selections of the second and third hub nodes, and so forth. The proposed MSLS entails a two-level neighborhood search process, characterized by expedited exploration for location at the first level and allocation exploration at the second level. Further elaboration on these constituent elements is provided in the subsequent subsections.

4.2 Initial solution generation

Considering the set of nodes $N = \{1,2,\ldots,n\}$, a solution to USAHLP problem can be represented as $X = (H,A)$, with $H = \{h_1,h_2,\ldots,h_m\}$ a set containing indices of m nodes selected as hubs, and $A = (a_1,a_2,\ldots,a_i,\ldots,a_n)$ the assignment vector by which node i is assigned to the hub $a_i \in H$. In addition, each hub node is assigned to itself, i.e., $a_i = i$. In our proposed methodology, the cardinality of hub nodes is not predetermined and exhibits variability across different solutions. Figure 3 presents a depiction of the transportation network employed in our methodology, showcasing both its graphical and numerical representations.

The initial solution generation method in our proposed MSLS algorithm is the GRSCP, which gen-

erates starting solutions at random [22]. A pseudocode for this procedure is shown in Algorithm 2. To begin with, the first hub-node i is randomly selected from the set of all nodes and is considered to be the hub, $H = \{i\}$. All nodes are then assigned to the hub node i, i.e. $A = (i,i,i,\ldots,i)$, and a solution $X = (H,A)$ is obtained. The solution is then accepted as the initial solution. To select the second hub node, all non-hub nodes $(N-H)$ are checked, and each is independently added to the current initial solution as a new hub node to determine which causes the lowest cost between them. Each non-hub node is assigned to the nearest selected hub after a new hub is temporarily added to the solution. The candidate non-hub node with the lowest cost, provided that it also reduces the cost compared to the current initial solution, is selected as the new hub. If adding any of the non-hub nodes does not reduce the cost of the solution, the GRSCP is terminated, and the current initial solution is returned as the final solution.

Algorithm 2: The GRSCP

ConstructRandomSolution() :
Consider all nodes as candidates: $N \leftarrow \{1,2,3,\ldots,n\}$
Randomly select a node i from N as the first hub: $i \leftarrow rand(n)$
Consider node i as a set of hub nodes: $H \leftarrow \{i\}$
Assign all nodes to hub node i: $A \leftarrow (i,i,i,\ldots,i)$
Construct initial solution: $S \leftarrow (H,A)$
repeat
  Consider the last solution as the best: $S^* \leftarrow S$
  Initialize cost of new best solution: $f_{best} \leftarrow \infty$
  for all candidate node $j \in (N-H)$ **do**
    Add candidate node to the set of hub nodes: $H' \leftarrow H \cup \{j\}$
    Assign each non-hub node to the nearest hub: $A' \leftarrow AssignNearest(H')$
    Construct new candidate solution: $S' \leftarrow (H',A')$
    if $f(S') < f_{best}$ **then**
      Save new best solution cost: $f_{best} \leftarrow f(S')$
      Save new best candidate hub: $hub_{best} \leftarrow j$
    end if
  end for
  Accept best candidate hub: $H \leftarrow H \cup \{hub_{best}\}$
  Assign each non-hub node to the nearest hub: $A \leftarrow AssignNearest(H)$
  Construct new initial solution: $S \leftarrow (H,A)$
until $f(S) > f(S^*)$
return S^*

4.3 Enhanced local search procedure

Local search algorithms initiate the optimization process by commencing with an initial candidate solution. Subsequently, these algorithms iteratively advance towards neighboring solutions that exhibit minimal differences from the current solution. A neighborhood, in this context, signifies a collection of potential solutions that are derived from the current solution using a consistent mechanism. In our proposed methodology, we incorporate two distinct types of neighborhood structures to facilitate the

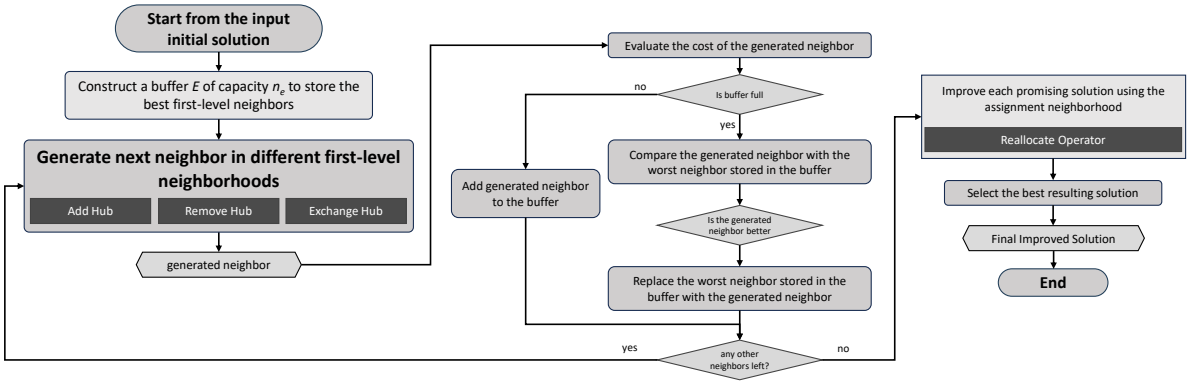


Figure 4: Flowchart of the proposed TLSP

local search procedure: hub neighborhoods and assignment neighborhoods. Moreover, to ensure fast execution of the local search process, a two-level approach is employed. At the initial level, emphasis is placed on carefully checking the hub neighborhoods, wherein a collection of potential neighbor solutions displaying promise is accumulated. Subsequently, at the second level, each of these promising neighbor solutions undergoes further investigation by modifying the assignment component. The main steps of our proposed TLSP are shown in Figure 4.

Details of TLSP are described in Algorithm 3. In the first level, the best neighbors of the add-hub, remove-hub, and interchange-hub neighborhoods of the current solution X are stored in a memory of capacity n_e referred to as the promising solutions set (PSS) and denoted by E. If the number of promising solutions in E is less than n_e, the generated neighbor is directly added to the PSS. Otherwise, if the new neighboring solution is better than the worst solution in E, the new neighbor will replace the worst solution in E. Each promising solution in E is subjected to a local search procedure at the second level to improve it. At the second level of the enhanced local search, a reallocation local search procedure is applied to each promising solution to obtain an improved solution. To minimize the overall execution time, the size of PSS is considered to be small. This strategic implementation allows for the exclusion of suboptimal solutions from the first search level, thereby effectively reducing the magnitude of the search space.

4.3.1 Hub neighborhoods

The first level of our enhanced local search procedure is a hub neighborhood search, which uses three neighborhood structures, including add-hub, remove-hub, and interchange-hub. To restrict the further investigation to only promising regions of the search space, the best neighboring solutions found in these three neighborhoods are stored in the promising solutions set (PSS). As illustrated in Figure 5, the three hub neighborhood structures are as follows:

Add-hub structure: The add-hub structure applies the $AddHub(S,i)$ procedure on the current solution S for each non-hub node i to obtain $|N|-|H|$ neighboring solutions. $AddHub(S,i)$ generates a neighboring solution S' from solution $S=(H,A)$ by adding node i as a new hub to the solution S through the following steps: 1) Add non-hub node i to the hub set H, and get the new hub set H' for S', i.e. $H'=H\in\{i\}$, 2) For each non-hub node, if its distance to the new hub i is less than its distance to the

Algorithm 3: The TLSP

LocalSearch(S, n_e) :
$E \leftarrow \emptyset$
for all $S' \in (n_{add}(S) \cup n_{remove}(S) \cup n_{interchange}(S))$ **do**
  if $|E| < n_e$ **then**
    Append the neighboring solution to the promising solution set (PSS): $E \leftarrow E \cup \{S'\}$
  else
    Find worst solution in PSS: $W \leftarrow$ worst solution in E
    if $f(S') < f(W)$ **then**
      Remove W from PSS: $E \leftarrow E - W$
      Add S' to PSS: $E \leftarrow E \cup \{S'\}$
    end if
  end if
end for
$F \leftarrow \emptyset$
for all $S' \in E$ **do**
  Unpack solution S': $(H', A') \leftarrow S'$
  Construct a list of all nodes: $N \leftarrow \{1, 2, 3, \ldots, n\}$
  Randomly permute the list of all nodes: $N \leftarrow randperm(N)$
  Compute the list of non-hub nodes: $T' \leftarrow N - H'$
  for all non-hub node $i \in T'$ **do**
    for all hub node $j \in H'$ **do**
      Create a new solution by assigning non-hub node i to hub node j in the current promising solution: S" $\leftarrow Reallocate(S', i, j)$
      if $f(S'') < f(S')$ **then**
        Update current promising solution to modified solution: $S' \leftarrow S''$
      end if
    end for
  end for
  Add current promising solution to F: $F \leftarrow F \cup \{S'\}$
end for
Find best promising solution: $S_{best} \leftarrow \arg\min_{S' \in F} f(S')$
return S_{best}

previous hub, assign it to the new hub i, $A' = AssignWhenCloser(H', A, i)$, 3) $S' = (H', A')$. In this way, there are $|N| - |H|$ solutions constructed. An example of a neighboring solution in an add-hub neighborhood is shown at the top of Figure 5, where new hub 6 is added to the solution. In the assignment vector, hub node 6 is assigned to itself, and non-hub node 1 is assigned to 6 since the new hub 6 is closer than the old hub 4.

Remove-hub structure: The remove-hub structure applies the $RemoveHub(S, i)$ procedure on the current solution S for each hub node i to obtain $|H|$ neighboring solutions. $RemoveHub(S, i)$ creates a neighboring solution S' from solution S by removing hub node $i \in H$ through the following steps: 1) Remove the hub node i from the hub set H in solution S to obtain the hub set H' for neighboring solution

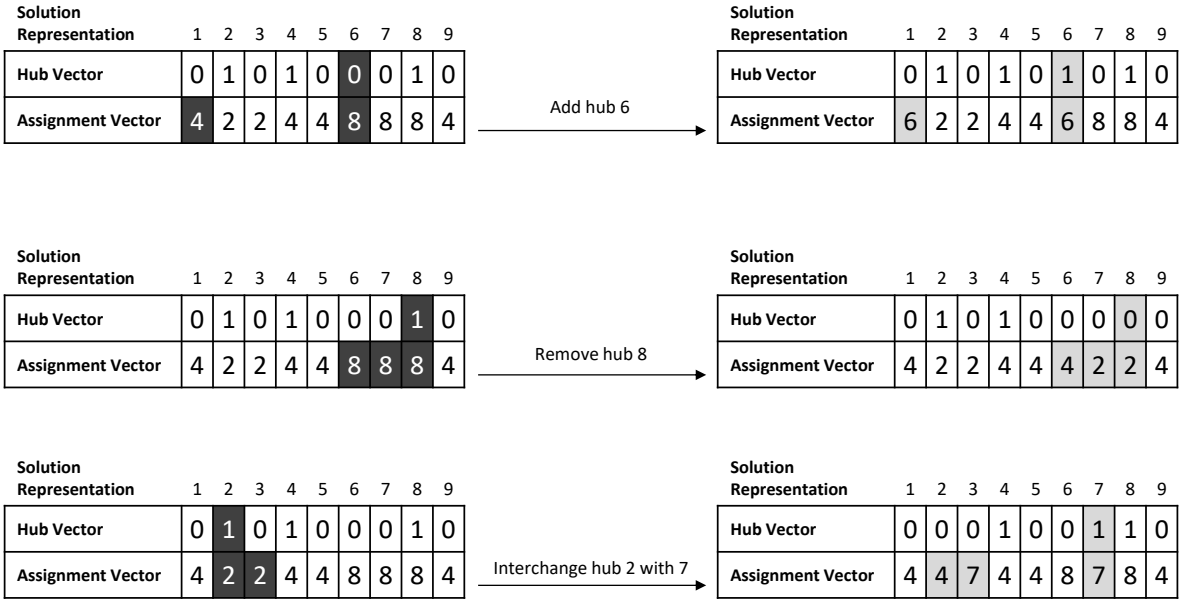


Figure 5: Hub-level neighborhood structures

S', i.e. $H' = H - \{i\}$, 2) For each node that was assigned to the removed hub i, assign that node to the nearest hub in H' to obtain A', $A' = ResetAssignment(H', A, i)$, 3) $S' = (H', A')$. Here, there are $|H|$ solutions constructed. For example, in the middle of Figure 5, hub 8 is removed, and its allocated nodes are assigned to their nearest hub in the new solution.

Interchange-hub structure: In the interchange-hub structure, the add-hub operator is applied after the remove-hub operator on the solution S to obtain $|H| \times (N - |H|)$ neighboring solutions. A new hub is added after an existing hub is removed. This procedure is executed for all existing hub nodes in set H and all current non-hub nodes in set $(N - H)$ to generate a comprehensive collection of neighboring solutions within this specific neighborhood. The allocation configurations are adjusted through the utilization of the *AddHub* and *RemoveHub* operators. At the bottom of Figure 5, hub node 2 is removed, and non-hub node 7 is added. There are $|H| \times (N - |H|)$ solutions constructed.

4.3.2 Assignment neighborhoods

In the second level of our enhanced local search, a neighborhood structure referred to as the reallocation structure is employed to alter the assignment component of each promising solution. To enhance the allocation quality, a greedy local search approach employing the reallocation neighborhood structure is implemented for each promising solution. Within this method, the non-hub nodes are sequentially evaluated in a random order within a given promising solution, assigning them to hub nodes one by one. Whenever a new assignment results in a decreased total cost for the solution, it is accepted, and the local search process proceeds to explore the next potential assignment. Upon exhaustively enumerating all non-hub nodes and their corresponding potential assignments, the greedy local search procedure concludes.

The reallocation neighborhood structure applies the procedure $Reallocate(S',i,j)$ for each non-hub node i and each hub node j in the current promising solution S' to obtain neighboring solutions. $Reallocate(S',i,j)$ generates a neighboring solution S" from solution S' through the following steps: 1) Consider the hub set H" of the new solution the same as hub set H' of the current solution S', i.e. $H" = H'$, 2) For non-hub node i, assign it to the hub $j \in H$, and obtain A". 3) $S" = (H",A")$. In this way, $|H| \times (N - |H|)$ neighboring solutions are constructed, and the best neighboring solution is accepted if it improves upon the current solution.

4.4 Characteristics and advantages

As part of our proposed MSLS algorithm, we employed the TLSP, a rich set of neighborhood structures, and random restarts to efficiently and effectively resolve USAHLP. Due to the slow and time-consuming nature of the nested neighborhood search, we selectively focused on a limited subset of promising neighboring solutions at the first level, which was subsequently enhanced through the utilization of the second-level reallocation neighborhood. This enhanced local search procedure is faster than a complete nested local search, but it may reduce the likelihood of obtaining optimal solutions. To overcome this weakness, we employed a diverse set of hub neighborhood structures as well as random restarts, which increases the chance of achieving an optimal solution. Consequently, our algorithm demonstrates a higher likelihood of exploring distinct regions within the search space in contrast to a general variable neighborhood search (GVNS) algorithm. In summary, the proposed MSLS procedure is both reliable and efficient.

5 Computational results and comparisons

As a comparison, we compare the results of our proposed MSLS algorithm with the CPLEX solver in GAMS, GVNS, and eight state-of-the-art algorithms. In this regard, we implemented the mathematical programming model of [42] in GAMS. Both the GVNS and our proposed MSLS algorithms were implemented in MATLAB R2015b and executed on a computer equipped with an Intel Core 2 Duo P8700 2.53 GHz processor and 4 GB of RAM. To facilitate a comprehensive comparison of the algorithms, a selection of sample problems sourced from the AP dataset was employed. AP dataset instances were solved 20 times with MSLS method, and the best cost obtained $(Best.Sol.)$, average cost obtained $(Average.Sol.)$, the average distance between the cost of obtained solutions and the cost of the optimal solution $(Average.Gap.)$, the average CPU processing time in seconds (CPUt), and the reliability of obtaining optimal solutions $(Rel.)$ are reported. The distance to the optimal solution was calculated as $Gap = (Solution.Cost. - Optimal.Cost.)/(Optimal.Cost.)$. In the case of 200-node sample problems, where an optimal solution is not available, the cost of the best solution reported in previous studies is used instead.

5.1 Australian post dataset

The Australian Post (AP) dataset is a famous benchmark within the field of hub location. It refers to a collection of real-world hub location sample problems that are exclusively owned by an Australian postal service. The dataset was first described in [12], and is available from the OR-Library [5, 6]. The dataset encompasses a diverse range of sample problems, including instances with node quantities of 10, 20, 25, 40, 50, 100, and 200. AP data includes postcode areas and their coordinates, flows between nodes, and

Table 1: Optimal results for AP dataset using CPLEX solver of GAMS

	Loose			Tight		
Size	Optimal. Hubs.	Optimal. Sol.	CPUt	Optimal. Hubs.	Optimal. Sol.	CPUt
10	3,4,7	224250.05	0.44	4,5,10	263399.92	0.14
20	7,14	234690.94	0.22	7,19	271128.14	0.27
25	8,18	236650.6	0.5	13	295667.83	0.28
40	14,28	240986.23	10.7	19	293164.83	1.16
50	15,36	237421.98	12.8	24	300420.98	9.2
100	29,73	238016.28	279.34	52	305097.95	99.88
200	N/A	N/A	N/A	N/A	N/A	N/A

cost factors α, δ, and γ. The AP problem instances are characterized by non-uniform flows and high input and output traffic for postal centers. Additionally, the AP dataset incorporates fixed costs associated with establishing hubs on each node. To provide diverse scenarios, we explored two variations of each sample problem: the loose (L) version and the tight (T) version [12, 21, 42, 43]. In the tight sample problems, the fixed costs for nodes exhibiting high input and output traffic are set to a significantly higher value. This deliberate increase in fixed costs aims to prevent these high-volume nodes from being considered potential hubs during the solution algorithms' optimization process. Consequently, this augmentation renders the problem-solving process more challenging in the case of tight sample problems. Conversely, the loose sample problems lack this specific feature, allowing for a less constrained optimization setting.

5.2 Optimal results

The optimal results for the AP dataset are summarized in Table 1. These results were obtained for sample problems containing up to 100 nodes using the CPLEX solver in GAMS. The CPLEX solver is capable of handling models with quadratic constraints and generating provably optimal solutions. It formulates the problem as a quadratically constrained program and solves it by the Barrier optimization method, in which the constraint coefficient matrix is constructed, and its Cholesky factorization is computed. During the experiments, the CPLEX solver exhibited high memory and computational demands for large instances. Due to the limited memory capacity of the computational system, CPLEX was able to solve only sample problems of up to 100 nodes from the AP dataset. The runtime growth shown in Table 1 also provides experimental evidence of the computational complexity of the formulation. For example, in loose instances, the CPLEX runtime increased from 12.8 seconds for the 50-node case to 279.34 seconds for the 100-node case, while in tight instances it increased from 9.2 seconds to 99.88 seconds. Moreover, the 200-node instances could not be solved on the available hardware. Although these observations do not constitute a formal proof of complexity, they clearly demonstrate that the computational burden increases rapidly with problem size. These findings further justify the development of faster and more scalable meta-heuristic algorithms for large-scale USAHLP instances.

5.3 Parameters tuning

The proposed MSLS algorithm involves two key parameters: the number of restarts (M) and the capacity of the PSS, denoted by (n_e). The parameter M specifies the number of iterations executed within the

Table 2: Results of parameters tuning for MSLS algorithm considering 20 runs

	Rel.					Average. Sol.					CPUt			
	M					*M*					*M*			
n_e	5	10	15	20	n_e	5	10	15	20	n_e	5	10	15	20
5	0.60	0.70	0.75	0.80	5	300647.12	300582.51	300582.51	300550.21	5	0.80	1.58	2.33	3.05
10	0.75	0.85	0.90	0.90	10	300582.51	300517.90	300485.60	300485.60	10	1.01	2.06	3.14	4.17
15	0.75	0.85	0.90	0.95	15	300582.51	300517.90	300485.60	300453.29	15	1.29	2.64	3.98	5.36
20	0.75	0.85	0.95	**1.00**	20	300582.51	300517.90	300453.29	**300420.99**	20	1.60	3.38	4.86	**6.74**

main loop of the algorithm. A higher value of M results in more restarts, thereby allowing the algorithm to explore a wider search space and increasing the likelihood of reaching the optimal solution, albeit at the cost of higher computational time. In the enhanced local search phase, the parameter n_e defines the size of the PSS, representing the subset of high-quality solutions retained for further exploration in the secondary search stage. Similar to M, increasing n_e enhances the probability of identifying the global optimum but also significantly increases computational effort.

To identify the appropriate values for the two fundamental parameters, a grid search technique was utilized. In this procedure, candidate values for both parameters were jointly evaluated to determine the optimal combination. The tight instance with 50 nodes from the Australian Post dataset was selected as a representative case for tuning, given its typical structure among other instances in the dataset. During the grid search, the number of restarts (M) was tested at 5, 10, 15, and 20, while the capacity of the promising solution set (n_e) was examined for the same set of values. For each candidate pair (M, n_e), the 50-node problem was solved twenty times, and the average solution cost (Average.Sol.), average CPU time, and reliability (Rel.) were calculated. The results are summarized in Table 2. Based on the reliability criterion, the highest reliability was achieved at $M = 20$ and $n_e = 20$. The corresponding average transportation cost and CPU time for each parameter combination are also reported in Table 2. Consequently, to ensure the greatest robustness in achieving optimal solutions, these values—$M = 20$ and $n_e = 20$—were adopted for all subsequent experiments in the proposed method.

5.4 Comparison with CPLEX solver

As previously noted, the CPLEX solver within GAMS is resource-intensive, particularly concerning memory and time, when dealing with larger problem instances. CPLEX was effectively limited to solving only the sample problems up to 100 nodes from the AP dataset. In sharp contrast, the proposed MSLS method demonstrated the capability to successfully solve significantly larger instances, specifically up to 200 nodes, achieving this in substantially less computational time compared to CPLEX. A detailed performance comparison between the proposed MSLS solver and the GAMS CPLEX solver is presented in Table 3. As the problem scale increases, the MSLS approach consistently requires significantly less time than CPLEX. For the 100L sample problem, our method matched the optimal solution obtained by CPLEX in only 66.54 seconds, whereas CPLEX required 279.34 seconds. Furthermore, exact computation using CPLEX mandates substantial RAM, a limitation that prevented it from solving the 200-node problems entirely. Conversely, MSLS successfully solved these larger instances and reproduced the best-known solutions reported for them. For problem instances up to 100 nodes, all 20 runs of the proposed MSLS matched the optimal solutions obtained by CPLEX. These results reveal that the

Table 3: Results of the proposed MSLS algorithm for sample problems of the AP dataset compared with the CPLEX solver of GAMS

Problem		GAMS + CPLEX			Proposed MSLS					
Size	Type	Optimal. Hubs.	Optimal. Sol.	CPUt	Best. Hubs.	Best. Sol.	Average. Sol.	Average. Gap.	CPUt	Rel.
10	Loose	3,4,7	224250.05	0.44	3,4,7	224250.06	224250.06	0	0.2	1
20	Loose	7,14	234690.94	0.22	7,14	234690.96	234690.96	0	0.8	1
25	Loose	8,18	236650.6	0.5	8,18	236650.63	236650.63	0	1.27	1
40	Loose	14,28	240986.23	10.7	14,28	240986.23	240986.23	0	5	1
50	Loose	15,36	237421.98	12.8	15,36	237421.99	237421.99	0	12.02	1
100	Loose	29,73	238016.28	279.34	29,73	238016.28	238016.28	0	66.54	1
200	Loose	N/A	N/A	N/A	43,184	233802.98	233802.98	0	534.25	1
10	Tight	4,5,10	263399.92	0.14	4,5,10	263399.95	263399.95	0	0.23	1
20	Tight	7,19	271128.14	0.27	7,19	271128.18	271128.18	0	0.83	1
25	Tight	13	295667.83	0.28	13	295667.84	295667.84	0	1.1	1
40	Tight	19	293164.83	1.16	19	293164.84	293164.84	0	4.16	1
50	Tight	24	300420.98	9.2	24	300420.99	300420.99	0	6.74	1
100	Tight	52	305097.95	99.88	52	305097.95	305097.95	0	45.87	1
200	Tight	N/A	N/A	N/A	54,122	272188.11	272188.11	0	668.9	1

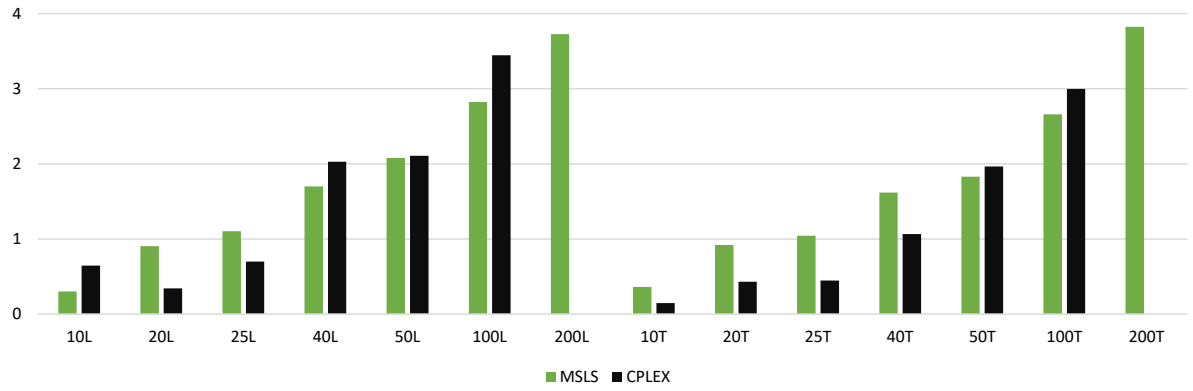


Figure 6: Log-Scale comparison of computation time for the proposed MSLS method and CPLEX solver

computational burden of the exact formulation increases sharply with instance size, whereas the proposed MSLS algorithm provides a more scalable alternative for medium and large-sized sample problems.

The bar chart in Figure 6 illustrates a direct comparison of the execution time required by the proposed MSLS method versus the industry-standard benchmark, the GAMS CPLEX Solver. This comparison is critical for thoroughly evaluating the computational efficiency of the MSLS algorithm against established high-performance solvers. Given that both methods may have achieved comparable solution quality (as indicated by average gap analyses), the time complexity becomes a decisive factor for practical deployment. The Y-axis is appropriately scaled logarithmically (by applying $log_{10}(10 * Y)$) to effectively visualize execution times spanning several orders of magnitude, ensuring that variations in fast-running instances are clearly discernible while managing the display of potentially very long computation times. Figure 6 demonstrates a significant efficiency advantage of MSLS over the established CPLEX benchmark for sample problems larger than 40 nodes, validating the algorithm's performance not only in solution quality but also in runtime efficiency.

5.5 Comparison with baseline

To quantitatively assess the combined impact of the random restart strategy and the PSS mechanism on the reliability and computational speed of MSLS, we conduct a comparative analysis against the GVNS algorithm, which serves as the baseline. The GVNS algorithm shares a fundamental structural similarity with MSLS, differing primarily in two core components. First, GVNS employs a complete nested local search utilizing the operators detailed in Section 4.3, rather than the specialized TLSP implemented in MSLS. Second, instead of a random restart, each iteration of GVNS incorporates a shaking mechanism to perturb the current solution slightly, thereby facilitating a transition to a new region of the solution space. If the local search fails to yield an improvement, the shaking intensity is systematically increased by one, and the process is repeated. This perturbation is executed for a maximum of K times, though the shaking size is reset to one immediately upon any improvement. Should the local search remain unsuccessful after K consecutive shakings, the current iteration is terminated, and the subsequent iteration commences with the shaking size reset to 1. Specifically, at a shaking size $k \in 1,2,..,K$, the perturbation applied to the GVNS solution involves randomly replacing k hub nodes with non-hub nodes. For our experimental validation, the GVNS configuration utilized a maximum shaking size of $K = 3$ and a maximum iteration limit of 5.

Table 4 summarizes the comparative performance results between the baseline GVNS algorithm and the proposed MSLS across the AP dataset sample problems. Each instance was subjected to 20 independent executions using GVNS, with the averaged metrics reported. For the medium and large-scale instances detailed in Table 4, the GVNS demonstrated excessive computational burden, rendering its performance generally unacceptable for practical application. In contrast, the integration of TLSP within our MSLS framework achieved a substantial reduction in required computation time. Specifically, the proposed MSLS is at least tenfold faster than GVNS across these challenging instances. As a case in point, GVNS required an average of 1569.93 seconds to solve the 100L problem, whereas MSLS achieved the solution in just 66.54 seconds. Furthermore, for sample problems involving 100 nodes or more, the execution time for the GVNS algorithm became computationally prohibitive; consequently, GVNS was incapable of solving the 200-node problems within a one-hour limit. Beyond speed, the inherent randomization in the GVNS method compromises solution certainty; for the 20T sample problem from the AP dataset, GVNS failed to converge to the optimal solution in one run out of the 20 trials. These results underscore that while TLSP in MSLS restricts the search primarily to promising solutions, the accompanying random restart mechanism effectively counterbalanced this constraint, boosting the reliability of MSLS to 100% across all tested sample problems. In conclusion, the GVNS algorithm suffers from intolerably high computation times in most application scenarios and exhibits lower solution reliability when compared to the proposed MSLS algorithm.

Figure 7 presents a comparative evaluation of the execution time required by the novel MSLS method against the baseline GVNS algorithm across the entire set of AP dataset instances. This analysis critically highlights the computational superiority achieved by the MSLS approach. As clearly depicted on the logarithmically scaled Y-axis, the MSLS method consistently demonstrates significantly shorter computation times compared to the baseline GVNS algorithm across all tested problem sizes. The performance improvement is particularly pronounced when considering scalability. Specifically, the GVNS algorithm failed to converge on the largest instance (200 nodes) within a reasonable operational threshold (one hour). In contrast, the proposed MSLS method successfully solved this same challenging 200-node problem in approximately 11 minutes. This rapid convergence confirms that the modifications integrated

Table 4: Results of our proposed MSLS algorithm for sample problems of the AP dataset compared with GVNS

Problem		GVNS						Proposed MSLS					
Size	Type	Best. Hubs.	Best. Sol.	Average. Sol.	Average. Gap.	CPUt	Rel.	Best. Hubs.	Best. Sol.	Average. Sol.	Average. Gap.	CPUt	Rel.
10	Loose	3,4,7	224250.06	224250.06	0	0.53	1	3,4,7	224250.06	224250.06	0	0.2	1
20	Loose	7,14	234690.96	234690.96	0	3.31	1	7,14	234690.96	234690.96	0	0.8	1
25	Loose	8,18	236650.63	236650.63	0	7.18	1	8,18	236650.63	236650.63	0	1.27	1
40	Loose	14,28	240986.23	240986.23	0	43.38	1	14,28	240986.23	240986.23	0	5	1
50	Loose	15,36	237421.99	237421.99	0	94.03	1	15,36	237421.99	237421.99	0	12.02	1
100	Loose	29,73	238016.28	238016.28	0	1569.93	1	29,73	238016.28	238016.28	0	66.54	1
200	Loose	N/A	N/A	N/A	N/A	N/A	N/A	43,184	233802.98	233802.98	0	534.25	1
10	Tight	4,5,10	263399.95	263399.95	0	0.51	1	4,5,10	263399.95	263399.95	0	0.23	1
20	Tight	7,19	271128.18	271480.16	0.0013	2.34	0.95	7,19	271128.18	271128.18	0	0.83	1
25	Tight	13	295667.84	295667.84	0	3.56	1	13	295667.84	295667.84	0	1.1	1
40	Tight	19	293164.84	293164.84	0	20.46	1	19	293164.84	293164.84	0	4.16	1
50	Tight	24	300420.99	300420.99	0	51.61	1	24	300420.99	300420.99	0	6.74	1
100	Tight	52	305097.95	305097.95	0	723.12	1	52	305097.95	305097.95	0	45.87	1
200	Tight	N/A	N/A	N/A	N/A	N/A	N/A	54,122	272188.11	272188.11	0	668.9	1

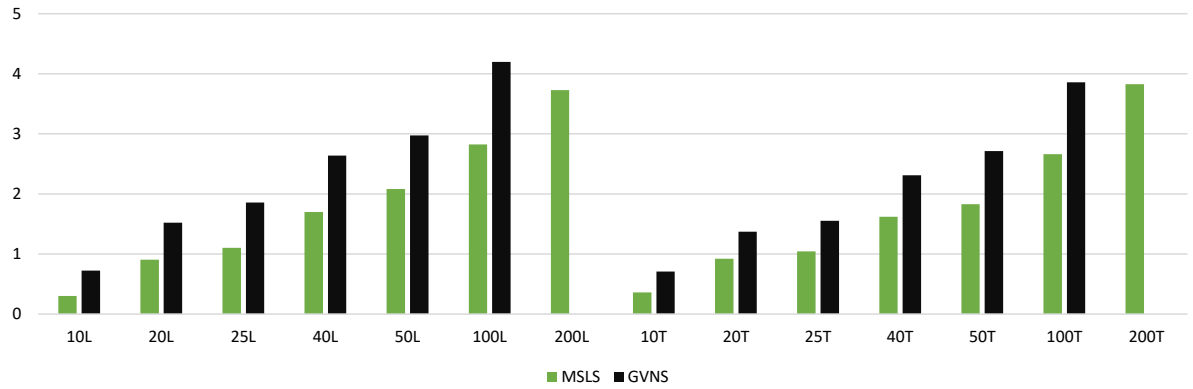


Figure 7: Log-scale comparison of computation time for the proposed MSLS method and GVNS baseline

into MSLS effectively mitigate the performance bottlenecks observed in the original GVNS structure, leading to superior efficiency and scalability for larger-scale optimization tasks.

5.6 Comparison with other algorithms

This section is dedicated to benchmarking the performance of the proposed MSLS algorithm against several prominent algorithms previously proposed in the literature for solving USAHLP. The current state-of-the-art methods considered for this comparison include the learning-based probabilistic TS (LPTS) developed by Guan et al. [21], the improved GA (IGA) by Alizadeh et al. [3], and the memetic algorithm (MA) introduced by Maric et al. [29]. Furthermore, this comparative analysis incorporates results from other established techniques, such as the GA by Topcuoglu et al. [43], the GA and TS (GATS) proposed by Abdinnour-Helm [1, 43], and the various multi-start TS (MSTS-1, MSTS-2, and MSTS-3) variants developed by Silva and Cunha [42]. Consistent with the evaluation methodology of our own approach, all competing algorithms were assessed by their respective authors using the instances derived from the established AP dataset; the pertinent performance metrics for these algorithms were carefully extracted from their corresponding publications for this study. For any given criterion, an algorithm was excluded

Table 5: Performance comparison of MSLS algorithm with six other algorithms in terms of Rel. criterion on the AP dataset

Size	Type	MSLS	IGA	MSTS-1	MSTS-2	MSTS-3	GA	GATS
10	Loose	1.00 (1)	1.00 (1)	0.02 (6)	0.02 (6)	0.04 (5)	0.83 (3)	0.71 (4)
20	Loose	1.00 (1)	1.00 (1)	0.02 (7)	0.10 (4)	0.06 (5)	0.96 (3)	0.06 (5)
25	Loose	1.00 (1)	1.00 (1)	0.02 (6)	0.04 (5)	0.08 (4)	0.90 (3)	0.02 (6)
40	Loose	1.00 (1)	1.00 (1)	0.02 (6)	0.06 (4)	0.04 (5)	0.55 (3)	0.01 (7)
50	Loose	1.00 (1)	1.00 (1)	0.02 (4)	0.02 (4)	0.02 (4)	0.59 (3)	0.01 (7)
100	Loose	1.00 (1)	0.95 (2)	0.02 (3)	0.02 (3)	0.02 (3)	0.02 (3)	0.01 (7)
200	Loose	1.00 (1)	0.90 (2)	0.00 (6)	0.00 (6)	0.02 (3)	0.01 (4)	0.01 (4)
10	Tight	1.00 (1)	1.00 (1)	0.04 (5)	0.02 (6)	0.02 (6)	0.66 (3)	0.62 (4)
20	Tight	1.00 (1)	1.00 (1)	0.06 (5)	0.02 (6)	0.02 (6)	0.28 (3)	0.09 (4)
25	Tight	1.00 (1)	1.00 (1)	0.40 (6)	0.60 (4)	0.58 (5)	1.00 (1)	0.06 (7)
40	Tight	1.00 (1)	1.00 (1)	0.54 (3)	0.44 (4)	0.44 (4)	0.10 (6)	0.01 (7)
50	Tight	1.00 (1)	1.00 (1)	0.44 (3)	0.36 (5)	0.40 (4)	0.04 (6)	0.01 (7)
100	Tight	1.00 (1)	1.00 (1)	0.20 (3)	0.14 (4)	0.12 (5)	0.01 (6)	0.01 (6)
200	Tight	1.00 (1)	0.45 (2)	0.00 (7)	0.02 (3)	0.02 (3)	0.02 (3)	0.01 (6)
Avg. Rel.		1.00	0.95	0.13	0.13	0.13	0.43	0.12
Avg. Rank		1.00	1.21	5.00	4.57	4.43	3.57	5.79

from the comparative analysis if its corresponding results were not explicitly evaluated in the source literature or if the metric could not be calculated directly from the reported data within the reference article.

Table 5 presents the comparative performance of algorithms based on the Rel. criterion, which measures the success rate in achieving the provably optimal solution over multiple runs. Reliability values are taken from the literature. For clarity, each algorithm's relative rank within each instance is shown in parentheses. The bottom rows of Table 5 summarize the average reliability and rank across all AP dataset samples for the seven algorithms. The proposed MSLS clearly leads, achieving a perfect average reliability of 100% on the AP dataset. The second-best was the IGA of Alizadeh et al. [3], with an average reliability of 95%. This results in MSLS holding an average rank of 1.00, better than the IGA's 1.21 and Topcuoglu's GA at 3.57. Notably, the GATS and the three MSTS variants achieved average reliabilities below 15%, a level considered unacceptable in most practical applications. These results confirm the MSLS algorithm as exceptionally reliable in consistently reaching the optimal solution for USAHLP.

The bar chart in Figure 8 illustrates the average reliability achieved by the proposed MSLS method in comparison with several established algorithms (IGA, MSTS variants, GA, GATS) tested on the comprehensive AP dataset. Reliability is quantified on a scale where 1.00 represents perfect success or the highest stability in solution quality across multiple runs. The results unequivocally establish the superior reliability of the proposed MSLS method, which attained an average reliability score of 1.00. This perfect score indicates that the MSLS algorithm successfully found the desired solution quality consistently across all trials within the AP benchmark suite. In contrast, while IGA shows a respectable 0.95 average reliability, the remaining algorithms exhibit significantly lower and, in some cases, critically low reliability scores (e.g., MSTS variants at 0.13). This disparity underscores that the enhancements implemented within the MSLS framework have successfully overcome the inherent instability issues present in the

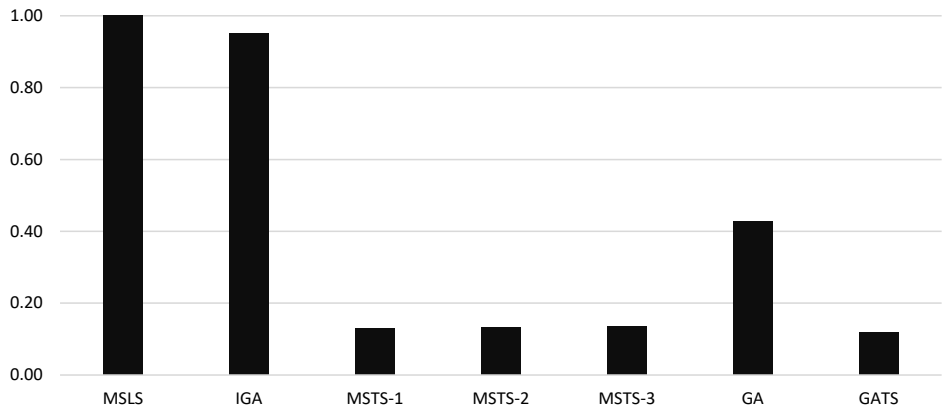


Figure 8: Comparison of average reliability for the proposed MSLS method and six other algorithms

compared methods, leading to a robust and dependable optimization performance.

The comparative results concerning the average gap (Avg. Gap.) criterion are summarized in Table 6. The rank achieved by each algorithm for every tested instance is reported parenthetically within the table. The bottom rows of Table 6 display the mean Avg. Gap. and the average rank for the seven considered algorithms across all AP dataset sample problems. Critically, the proposed MSLS algorithm, along with the LPTS algorithm proposed by Guan et al. [21], achieved a perfect Avg. Gap. of zero for all sample problems. Consequently, these two algorithms dominate the comparison. This superior performance is reflected in their average ranks: both MSLS and LPTS secured an average rank of 1.00. In contrast, the IGA of Alizadeh et al. [3] recorded an average rank of 1.14, and the Memetic Algorithm (MA) by Maric et al. [29] achieved an average rank of 2.14. The three MSTS variants yielded results significantly weaker than acceptable thresholds for most practical scenarios. These findings conclusively demonstrate that the proposed MSLS algorithm consistently delivers solutions that are either precisely optimal or extremely close to the optimum.

To provide a fair and meaningful comparison regarding computational efficiency, we conducted an evaluation of the computation time against state-of-the-art methods. Acknowledging that the hardware configurations and programming languages used in the original literature vary significantly, a direct comparison of raw CPU times would be inherently biased. The proposed MSLS algorithm was implemented in MATLAB and executed on an Intel Core 2 Duo P8700 at 2.53 GHz with 4 GB RAM. IGA [3] also used similar hardware and programming language. In contrast, all seven other algorithms were coded in the C programming language. The three MSTS variants [42] were executed on significantly older hardware (Intel Pentium IV 3.0 GHz with 1 GB RAM). Similarly, the GA algorithm of Topcuoglu and GATS was executed on an Intel Pentium IV 1.6 GHz processor [43]. On the other hand, MA [29] used a modern Intel Core i7-860 at 2.8 GHz with 8 GB RAM. Similarly, the LPTS algorithm was executed on an Intel Core i3-2350M 2.30 GHz processor with 4 GB RAM [21]. To establish a level playing field for this performance metric, all reported CPU times from the literature were normalized based on CPU speed benchmarks sourced from https://www.cpubenchmark.net/. For instance, to account for the superior processing power of the hardware used for the MA results (which was approximately 4.29 times faster than our execution platform), the running times reported in [29] were multiplied by a normalization factor of 4.29 before comparison with the MSLS execution time. This rigorous normalization process allows for a

Table 6: Performance comparison of MSLS algorithm with six other algorithms in terms of Avg. Gap. criterion on the AP dataset

Size	Type	MSLS	IGA	MSTS-1	MSTS-2	MSTS-3	MA	LPTS
10	Loose	0.0000 (1)	0.0000 (1)	0.0809 (7)	0.0512 (5)	0.0788 (6)	0.0000 (1)	0.0000 (1)
20	Loose	0.0000 (1)	0.0000 (1)	0.0714 (6)	0.0552 (5)	0.0987 (7)	0.0000 (1)	0.0000 (1)
25	Loose	0.0000 (1)	0.0000 (1)	0.0752 (6)	0.0665 (5)	0.0931 (7)	0.0000 (1)	0.0000 (1)
40	Loose	0.0000 (1)	0.0000 (1)	0.1221 (7)	0.0571 (5)	0.0875 (6)	0.0110 (4)	0.0000 (1)
50	Loose	0.0000 (1)	0.0000 (1)	0.1345 (7)	0.0873 (5)	0.1116 (6)	0.0020 (4)	0.0000 (1)
100	Loose	0.0000 (1)	0.0000 (1)	0.1091 (7)	0.0906 (6)	0.0795 (5)	0.0000 (1)	0.0000 (1)
200	Loose	0.0000 (1)	0.0000 (1)	0.1392 (7)	0.1178 (6)	0.1027 (5)	0.0150 (4)	0.0000 (1)
10	Tight	0.0000 (1)	0.0000 (1)	0.0302 (5)	0.0442 (7)	0.0431 (6)	0.0000 (1)	0.0000 (1)
20	Tight	0.0000 (1)	0.0000 (1)	0.0357 (5)	0.0421 (6)	0.0485 (7)	0.0000 (1)	0.0000 (1)
25	Tight	0.0000 (1)	0.0000 (1)	0.0399 (7)	0.0297 (5)	0.0332 (6)	0.0000 (1)	0.0000 (1)
40	Tight	0.0000 (1)	0.0000 (1)	0.0359 (5)	0.0478 (6)	0.0495 (7)	0.0000 (1)	0.0000 (1)
50	Tight	0.0000 (1)	0.0000 (1)	0.0363 (5)	0.0414 (6)	0.0484 (7)	0.0220 (4)	0.0000 (1)
100	Tight	0.0000 (1)	0.0000 (1)	0.2099 (5)	0.2371 (6)	0.2423 (7)	0.0000 (1)	0.0000 (1)
200	Tight	0.0000 (1)	0.0013 (3)	0.3167 (7)	0.2641 (6)	0.2013 (4)	0.2080 (5)	0.0000 (1)
Mean Avg. Gap.		0.0000	0.0001	0.1026	0.0880	0.0942	0.0184	0.0000
Avg. Rank		1.00	1.14	6.14	5.64	6.14	2.14	1.00

better comparison of the actual computational efficiency of the MSLS against the other eight algorithms for solving USAHLP. However, the gap caused by the programming language used to implement each algorithm still exists and will certainly affect the results.

Comparison results in terms of normalized computation time are presented in Table 7. The rank of each algorithm for every row is written inside parentheses. The average computation time and average rank of these nine algorithms for sample problems of the AP dataset are presented in the bottom rows of the table. The resulting execution times reveal a distinct pattern: algorithms implemented in C, except GATS, consistently yielded significantly lower run times than those coded in MATLAB. This discrepancy is attributed to the inherent characteristics of the MATLAB environment, including its slower execution of iterative loops (unless heavily vectorized), dynamic memory allocation/deallocation overhead, computational costs associated with external library calls, and generally lower programmer control over fine-grained execution optimization. Specifically, six out of the seven algorithms implemented in C achieved an average solution time of less than 5 seconds for the AP dataset problems. Conversely, the proposed MSLS method and the IGA method [3], both implemented in MATLAB, required substantially longer average times of 96 seconds and 30 seconds, respectively, for the same set of AP dataset problems. Further analysis of the MATLAB implementations showed that the MSLS method exhibited faster execution than IGA for problems with fewer than 40 nodes. However, as the problem size increased, the IGA algorithm demonstrated superior execution speed. This crossover suggests that the complexity of TLSP search inherent in the proposed MSLS method, while providing a more thorough spatial exploration, introduces a significant time cost, particularly when scaling to larger problem instances. Finally, the proposed method was faster than GATS in terms of average computation time.

In summation, the comparative analysis across multiple performance criteria demonstrates that the proposed MSLS algorithm offers a superior balance of solution quality and reliability for USAHLP. In

Table 7: Performance comparison of MSLS algorithm with eight other algorithms in terms of normalized computation time on the AP dataset

Size	Type	MSLS	IGA	MSTS-1	MSTS-2	MSTS-3	GA	GATS	MA	LPTS
10	Loose	0.2000 (8)	1.2200 (9)	0.0001 (2)	0.0001 (1)	0.0001 (2)	0.0228 (6)	0.1162 (7)	0.0225 (5)	0.0151 (4)
20	Loose	0.8000 (8)	2.0100 (9)	0.0015 (3)	0.0004 (2)	0.0002 (1)	0.0955 (6)	0.6860 (7)	0.0613 (4)	0.0633 (5)
25	Loose	1.2700 (7)	2.5900 (9)	0.0044 (3)	0.0007 (2)	0.0004 (1)	0.1526 (6)	1.4696 (8)	0.0818 (4)	0.0905 (5)
40	Loose	5.0000 (7)	5.0100 (8)	0.0004 (1)	0.0029 (3)	0.0013 (2)	0.4971 (6)	6.7147 (9)	0.1676 (4)	0.2835 (5)
50	Loose	12.020 (8)	8.6200 (7)	0.0015 (1)	0.0060 (3)	0.0025 (2)	0.8635 (6)	16.236 (9)	0.2494 (4)	0.5368 (5)
100	Loose	66.540 (8)	33.220 (7)	0.0472 (3)	0.0425 (2)	0.0160 (1)	5.6281 (6)	133.96 (9)	0.8053 (4)	3.5255 (5)
200	Loose	534.25 (8)	183.28 (7)	4.0975 (3)	2.6543 (2)	2.4227 (1)	45.568 (6)	17933 (9)	5.9050 (4)	27.851 (5)
10	Tight	0.2300 (8)	1.3700 (9)	0.0001 (3)	0.0001 (1)	0.0001 (1)	0.0187 (5)	0.1162 (7)	0.0225 (6)	0.0151 (4)
20	Tight	0.8300 (8)	2.5000 (9)	0.0015 (3)	0.0004 (2)	0.0002 (1)	0.0737 (6)	0.6829 (7)	0.0613 (5)	0.0513 (4)
25	Tight	1.1000 (7)	2.4100 (9)	0.0041 (3)	0.0008 (2)	0.0003 (1)	0.1183 (6)	1.7249 (8)	0.0634 (4)	0.0965 (5)
40	Tight	4.1600 (8)	3.9800 (7)	0.0005 (1)	0.0029 (3)	0.0012 (2)	0.3622 (6)	8.5859 (9)	0.1226 (4)	0.2744 (5)
50	Tight	6.7400 (8)	5.5700 (7)	0.0016 (1)	0.0056 (3)	0.0026 (2)	0.6040 (6)	15.651 (9)	0.2167 (4)	0.5368 (5)
100	Tight	45.870 (8)	14.420 (7)	0.0439 (3)	0.0421 (2)	0.0185 (1)	4.0797 (5)	139.56 (9)	0.9770 (4)	4.1106 (6)
200	Tight	668.90 (8)	162.10 (7)	3.9720 (3)	2.6503 (2)	2.4779 (1)	43.154 (6)	17933 (9)	6.6061 (4)	29.347 (5)
Average Time		96.28	30.59	0.58	0.39	0.35	7.23	2585.18	1.10	4.77
Avg. Rank		7.79	7.93	2.36	2.14	1.36	5.86	8.29	4.29	4.86

terms of Reliability, the MSLS algorithm secured the highest average rank among all evaluated methods. Similarly, concerning solution proximity to optimality, MSLS shared the top rank with the LPTS algorithm [21] under the Avg. Gap. criterion, confirming that both yield solutions closest to the true optimal values. While the computation time analysis indicated that MSLS (with an average rank of 7.79 out of nine algorithms) was not the fastest—largely due to its implementation in MATLAB relative to C-based counterparts—its execution time was deemed tolerable, particularly for some application scenarios where solution quality supersedes instantaneous performance. Collectively, these findings establish that the MSLS algorithm is highly reliable and consistently delivers high-quality solutions, justifiable trade-off for the modest increase in computation time observed.

6 Conclusion

To solve the USAHLP problem, a robust Multi-Start Local Search algorithm (MSLS) is proposed in this paper. This approach effectively searches the solution space by leveraging a strategic random restart mechanism alongside a rich set of hub neighborhood structures to perform a fast, two-level exploration and exploitation of the search space. Our comparative analysis confirms the efficacy of MSLS. When benchmarked against the CLPLEX solver in GAMS and the GVNS algorithm, the proposed MSLS method was able to solve large-scale USAHLP sample problems much faster. Furthermore, in a rigorous comparison against eight other state-of-the-art algorithms from the literature, MSLS demonstrated superior reliability and consistently provided solutions that were equal to or very close to the best-known solutions. This study confirms that meta-heuristic algorithms and their improved variants, like MSLS, are highly suitable methods for solving large-scale hub location problems while maintaining high solution quality. While MSLS achieves excellent solution quality, a primary limitation observed is the increased computational time required for convergence compared to some faster, simpler heuristics. As a direction for future research, it may be beneficial to explore hybridization strategies, combining population-based

search algorithms with single-solution search algorithms like MSLS to potentially improve computational efficiency without sacrificing solution quality. Furthermore, the MSLS framework can be extended to other variants of hub location problems, such as capacitated, r-allocation, and multi-allocation modes, for further investigation.

Conflict of interest

The authors declare that they have no conflict of interest.

References

[1] S. Abdinnour-Helm, *A hybrid heuristic for the uncapacitated hub location problem*, European J. Oper. Res. **106(2-3)** (1998) 489–499.

[2] R. Abyazi-Sani, R. Ghanbari, *An efficient tabu search for solving the uncapacitated single allocation hub location problem*, Comput. Ind. Eng. **93** (2016) 99–109.

[3] M. Alizadeh Firozi, V. Kiani, H. Karimi, *Improved genetic algorithm with diversity and local search for uncapacitated single allocation hub location problem*, J. Decis. Oper. Res. **6(4)** (2022) 536–552.

[4] A. Azodi, J. Fathali, M. Ghiyasi, T. Sayyar, *Optimal location of healthcare and treatment centers with complex structures based on performance*, J. Math. Model. (2026) [In Press].

[5] J. E. Beasley, *OR-Library*, http://people.brunel.ac.uk/ mastjjb/jeb/orlib/phubinfo.html, Accessed 06 Sep 2025.

[6] J. E. Beasley, *OR-Library: distributing test problems by electronic mail*, J. Oper. Res. Soc. **41(11)** (1990) 1069–1072.

[7] D.L. Bryan, M. E. O'kelly, *Hub-and-spoke networks in air transportation: an analytical review*, J. Reg. Sci. **39** (1999) 275–295.

[8] I. Contreras, M. O'Kelly, *Hub Location Problems*, In: Laporte, G., Nickel, S., Saldanha da Gama, F. (Eds.), Location Science, Springer International Publishing, Cham (2019) 327–363.

[9] A. Cuzzocrea, L. Canadè, G. Fornari, V. Gatto, A. Hafsaoui, *Effective and efficient heuristic algorithms for supporting optimal location of hubs over networks with demand uncertainty*, International Conference on Database and Expert Systems Applications, (2023) 84–98.

[10] D.D. Čvokić, Z. Stanimirović, *A single allocation hub location and pricing problem*, Comput. Appl. Math. **39** (2020) 40.

[11] F. Daneshdoost, M. Hajiaghaei-Keshteli, R. Sahin, S. Niroomand, *Tabu search based hybrid metaheuristic approaches for schedule-based production cost minimization problem for the case of cable manufacturing systems*, Informatica **33(3)** (2022) 499–522.

[12] A.T. Ernst, M. Krishnamoorthy, *Efficient algorithms for the uncapacitated single allocation p-hub median problem*, Locat. Sci. **4(3)** (1996) 139–154.

[13] I. Espejo, A. Marín, J.M. Muñoz-Ocaña, A.M. Rodríguez-Chía, *A new formulation and branch-and-cut method for single-allocation hub location problems*, Comput. Oper. Res. **155** (2023) 106241.

[14] R.Z. Farahani, M. Hekmatfar, A.B. Arabani, E. Nikbakhsh, *Hub location problems: A review of models, classification, solution techniques, and applications*, Comput. Ind. Eng. **64(4)** (2013) 1096–1109.

[15] P. Farmakis, A. Chassiakos, S. Karatzas, *A multi-objective tri-level algorithm for hub-and-spoke network in short sea shipping transportation*, Algorithms **16** (2023) 379.

[16] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco (1979).

[17] M.A. Gargouri, N. Hamani, N. Mrabti, L. Kermad, *Optimization of the collaborative hub location problem with metaheuristics*, Mathematics **9** (2021) 2759.

[18] N. Ghaffarinasab, B.Y. Kara, *Benders decomposition algorithms for two variants of the single allocation hub location problem*, Netw. Spat. Econ. **19(1)** (2019) 83–108.

[19] M. Gholami, J. Fathali, *Mathematical models for the variable weights version of the inverse minimax circle location problem*, J. Math. Model. **9(1)** (2021) 137–144.

[20] F.Z. Grine, O. Kamach, A. Khatab, N. Sefiani, *An artificial immune system algorithm for solving the uncapacitated single allocation p-Hub median problem*, Int. J. Electr. Comput. Eng. **11(3)** (2021) 2293–2306.

[21] J. Guan, G. Lin, H.B. Feng, *A learning-based probabilistic tabu search for the uncapacitated single allocation hub location problem*, Comput. Oper. Res. **98** (2018) 1–12.

[22] J. Guan, G. Lin, H B. Feng, *A multi-start iterated local search algorithm for the uncapacitated single allocation hub location problem*, Appl. Soft Comput. **73** (2018) 230–241.

[23] S.Y. Hsieh, S.S. Kao, *A survey of hub location problems*, J. Interconnect. Netw. **19(1)** (2019) 1940005.

[24] M. Imanparast, V. Kiani, *A practical heuristic for maximum coverage in large-scale continuous location problem*, J. Math. Model. **9(4)** (2021) 555–572.

[25] O. Kemmar, K. Bouamrane, S. Gelareh, *Hub location problem in round-trip service applications*, RAIRO Oper. Res. **55** (2021) 2831–2858.

[26] O. Kulkarni, M. Dahan, B. Montreuil, *Resilient hyperconnected parcel delivery network design under disruption risks*, Int. J. Prod. Econ. **251** (2022) 108499.

[27] M. Leyerer, M.O. Sonneberg, M. Heumann, M. Breitner, *Decision support for sustainable and resilience-oriented urban parcel delivery*, EURO J. Decis. Process. **7(3-4)** (2019) 267–300.

[28] R. Maharjan, S. Hanaoka, *A credibility-based multi-objective temporary logistics hub location-allocation model for relief supply and distribution under uncertainty*, Socioecon. Plann. Sci. **70** (2020) 100727.

[29] M. Marić, Z. Stanimirović, P. Stanojević, *An efficient memetic algorithm for the uncapacitated single allocation hub location problem*, Soft Comput. **17** (3) (2013) 445–466.

[30] M. Mohammadi, P. Jula, R. Tavakkoli-Moghaddam, *Reliable single-allocation hub location problem with disruptions*, Transp. Res. Part E Logist. Transp. Rev. **123** (2019) 90–120.

[31] M. Mohebbi, H.R. Maleki, S. Niroomand, *Mathematical model and hybrid meta-heuristic solution approaches for hub location problem with hybrid drone-airplane delivery mode*, J. Math. Model. **14(2)** (2026) 609–646.

[32] F. Momayezi, S.K. Chaharsooghi, M.M. Sepehri, A.H. Kashan, *The capacitated modular single-allocation hub location problem with possibilities of hubs disruptions: modeling and a solution algorithm*, Oper. Res. **21** (2021) 139–166.

[33] S. Niroomand, *Hybrid artificial electric field algorithm for assembly line balancing problem with equipment model selection possibility*, Knowl.-Based Syst. **219** (2021) 106905.

[34] S. Niroomand, B. Vizvari, *Exact mathematical formulations and metaheuristic algorithms for production cost minimization: a case study of the cable industry*, Int. Trans. Oper. Res. **22(3)** (2015) 519–544.

[35] M E. O'kelly, *A quadratic integer program for the location of interacting hub facilities*, European J. Oper. Res. **32(3)** (1987) 393–404.

[36] Y. Pingle, Z. Qinge, *Single allocation hub-and-spoke networks design based on ant colony optimization algorithm*, Sensors & Transducers **180** (2014) 131.

[37] H. Pirkul, D. A. Schilling, *An efficient procedure for designing single allocation hub and spoke systems*, J. Manag. Sci. **44** (1998) 235–242.

[38] B. Rostami, M. Chitsaz, O. Arslan, G. Laporte, A. Lodi, *Single allocation hub location with heterogeneous economies of scale*, Oper. Res. **70** (2022) 766–785.

[39] R.K. Sachan, *A realistic and sustainable logistics transportation planning: a new cost model, meta-heuristic solving approach, and results*, Evol. Intell. **17** (2024) 3687–3705.

[40] E. Shadkam, *Multi-objective supplier selection with the new hybrid COAW method*, International Journal of Supply Chain and Operations Resilience **1** (2021) 60–78.

[41] E. Shadkam, M. Parvizi, R. Rajabi, *he study of multi-objective supplier selection problem by a novel hybrid method: COA/ε-constraint*, Int. J. Res. Ind. Eng. **10** (2021) 223–237.

[42] M.R. Silva, C.B. Cunha, *New simple and efficient heuristics for the uncapacitated single allocation hub location problem*, Comput. Oper. Res. **36(12)** (2009) 3152–3165.

[43] H. Topcuoglu, F. Corut, M. Ermis, G. Yilmaz, *Solving the uncapacitated hub location problem using genetic algorithms*, Comput. Oper. Res. **32** (2005) 967–984.

[44] B. Wang, G. Shen, X. Wang, Y. Dong, Z. Li, *Hub-and-spoke network optimization with flow delay cost: the case of goods delivery on urban logistics networks in eastern China*, Mathematics **12** (2024) 1496.

[45] N. Yu, B. Dong, Y. Qu, M. Zhang, G. Chen, Q. Tan, Y. Wang, H. Dai, *Multiple-allocation hub-and-spoke network design with maximizing airline profit utility in air transportation network*, IEEE Trans. Intell. Transp. Syst. **25** (2024) 7294–7310.

[46] J. Zhang, H. Li, W. Han, Y. Li, *Research on optimization of multimodal hub-and-spoke transport network under uncertain demand*, Arch. Transp. **70** (2024) 137–157.