Mathematical Heuristics for Single-Source Facility Location and Its Variants

Authors: Kong Yunfeng, Kong Yunfeng

Date: 2021-06-09T00:00:00+00:00

Abstract

Considering spatial continuity of facility service areas and constraints on the number of facilities, the classical Single-Source Capacitated Facility Location Problem (SSCFLP) is extended to form variant problems: SSCFLP with spatial continuity of facility service areas (CFLSAP), SSCFLP with facility quantity constraints (SSCKFLP), and SSCFLP with both spatial continuity of facility service areas and quantity constraints (CKFLSAP). A matheuristic algorithm is proposed for SSCFLP and its variants. The algorithm starts from an initial solution and iteratively improves the current solution through very large-scale neighborhood search until several attempts fail to improve the current solution. The very large-scale neighborhood is defined through the following steps: randomly select a customer, choose Q nearest facilities to that customer and their assigned customers from the current solution, and then select the nearest candidate facilities for these customers. Using the selected subset of candidate facilities and subset of customers, a mathematical model for the subproblem is constructed, the model is solved, and the current solution is updated using the model' s solution. Two sets of case data are constructed to test the proposed algorithm. The results demonstrate that the matheuristic algorithm can effectively solve SSCFLP and its variants, with solution results extremely close to the optimal solution or lower bound of the objective value, showing relative differences of 0.01% (SSCFLP), 0.22% (CFLSAP), 0.00% (SSCKFLP), and 0.08% (CKFLSAP). Additionally, after adding spatial continuity constraints of facility service areas to SSCFLP or SSCKFLP, the increase in objective value is not substantial, but the optimal facility locations may change.

Full Text

A Matheuristic Algorithm for the Single-Source Capacitated Facility Location Problem and Its Variants

Yunfeng Kong[1,2]

1Key Laboratory of Geospatial Technology for the Middle and Lower Yellow River Regions, Ministry of Education, Henan University, Kaifeng, Henan 475000, China
2School of Geography and Environmental Science, Henan University, Kaifeng, Henan 475000, China

Abstract

This paper extends the classical single-source capacitated facility location problem (SSCFLP) by incorporating spatial contiguity constraints for facility service areas and limitations on the number of facilities, yielding three variant problems: SSCFLP with contiguous facility service areas (CFLSAP), SSCFLP with a fixed number of facilities (SSCKFLP), and SSCFLP with both contiguous service areas and a fixed number of facilities (CKFLSAP). To solve SSCFLP and its variants, we propose a matheuristic algorithm that begins with an initial solution and iteratively improves it through very large-scale neighborhood search until no further improvements can be made after several attempts. The very large neighborhood is defined through the following steps: randomly select a customer, identify the Q nearest facilities to this customer and their assigned customers from the current solution, and then select the nearest candidate facilities for these customers. Using the selected subset of candidate facilities and customers, we construct a subproblem mathematical model, solve it, and update the current solution with the model' s solution. We construct two sets of benchmark instances to test the proposed algorithm. The results demonstrate that the matheuristic algorithm effectively solves SSCFLP and its variants, producing solutions extremely close to optimal or lower bounds with relative gaps of 0.01% (SSCFLP), 0.22% (CFLSAP), 0.00% (SSCKFLP), and 0.08% (CKFLSAP). Additionally, adding spatial contiguity constraints to SSCFLP or SSCKFLP results in only modest increases in objective values, though the optimal facility locations may change.

Keywords: single-source capacitated facility location problem; spatial contiguity; matheuristic algorithm; very large-scale neighborhood search

1 Introduction

The facility location problem (FLP) represents a broad class of optimization problems concerned with determining optimal locations for facilities. These problems find widespread applications in planning decisions for public and commercial facilities such as schools, healthcare centers, police stations, elderly care facilities, emergency services, and logistics centers. Depending on the application context, numerous variants of FLP exist: continuous or discrete location, with or without capacity constraints, with splittable or non-splittable demand assignment, with or without facility costs, with specified or unspecified numbers of facilities, and with objectives ranging from efficiency to equity. Based on whether facility service and demand involve uncertainty, FLP can be further classified into deterministic and stochastic variants. Among these, the single-source capacitated facility location problem (SSCFLP)—characterized by non-splittable demand assignment, capacity constraints, and consideration of facility costs in discrete space—is particularly challenging and has attracted significant attention in recent years.

Let I denote the set of potential facility locations, J the set of customer locations. Each facility i (i I) has a maximum service capacity s_i and fixed construction cost f_i, while each customer j (j J) has service demand d_j and incurs cost c_{ij} when served by facility i. Define binary decision variable y_i indicating whether a facility is established at location i, and binary decision variable x_{ij} indicating whether customer j is assigned to facility i. The SSCFLP mathematical model is formulated as:

Minimize

$$\sum_{i\in I} f_i y_i + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}$$

Subject to:

$$\sum_{i\in I} x_{ij} = 1, \quad \forall j \in J$$

$$\sum_{j\in J} d_j x_{ij} \le s_i y_i, \quad \forall i \in I$$

$$y_i \in \{0,1\}, \quad \forall i \in I$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in I, j \in J$$

In this model, the objective function minimizes total facility fixed costs and service costs. Constraint (2) ensures each customer is served by exactly one facility, while constraint (3) requires that the total demand assigned to each facility does not exceed its maximum capacity. Constraints (4) and (5) define the two types of binary decision variables.

Solution methods for SSCFLP can be divided into two categories: exact algorithms and heuristic algorithms [?, ?]. Exact algorithms include branch-and-bound [?, ?], column generation [?], and Cut-and-Solve [?, ?]. Heuristic algorithms encompass Lagrangian relaxation heuristics (LH) [?, ?, ?, ?, ?, ?, ?], tabu search [?, ?, ?], very large-scale neighborhood search (VLNS) [?, ?], scatter search [?], kernel search [?], corridor method [?], and relaxation adaptive memory programming (RAMP) [?]. Due to the extremely high computational complexity of SSCFLP, exact algorithms struggle to efficiently solve large-scale instances. While LH algorithms are fast, they cannot guarantee solution quality.

Metaheuristic algorithms such as VLNS, kernel search, and corridor method are competitive in terms of solution quality, but their computational times remain relatively long.

Practical applications of SSCFLP often involve specific requirements, such as constraints on the number of facilities to be established [?, ?] or requirements for spatial contiguity of each facility' s service area. For example, in planning the layout of compulsory education schools in a region, decision-makers must consider school construction costs and requirements on the number of schools while simultaneously determining school locations and delineating their catchment areas. Similarly, in planning community health service centers in a city, national industry standards require each center to cover a specific spatial area to meet 15-minute accessibility requirements while considering service capacity constraints. To address these problems, it is necessary to extend the SSCFLP model by adding these two types of constraints. In districting problem research, three modeling approaches exist for spatial contiguity constraints: spanning tree models, order models, and flow models [?, ?]. Tree models involve a large number of constraints and are only suitable for small-scale cases (fewer than 50 nodes). Order and flow models have relatively fewer constraints. For p-regions problems, these three models can solve instances with up to 49 spatial units within three hours. The exact solution capability for SSCFLP extended with spatial contiguity constraints remains unknown.

Based on the above discussion, this paper extends the SSCFLP model by considering spatial contiguity of facility service areas and requirements on the number of facilities. We design a matheuristic algorithm to solve SSCFLP and its extended variants, and test the algorithm' s performance.

2 Problem Definition

Consider a specific geographic region containing n spatial units. Each spatial unit j has attribute d_j representing its service demand. Among the n units, m ($m \leq n$) units are suitable for facility construction and serve as candidate facility locations. Facility i has service capacity s_i and fixed construction cost f_i. The cost for unit i to be served by facility j is variable c_{ij} (the product of unit transportation cost between facilities i and j and the demand d_j). Let set I represent the m candidate facility units and set J represent the n demand units (I J). The SSCFLP model (1)-(5) applies to facility location decisions within this region.

Requiring spatial contiguity of facility service areas necessitates adding spatial contiguity constraints to the SSCFLP model. The geographic space is represented as a network graph where spatial units serve as nodes and adjacent units are connected by edges, forming an undirected network graph. Based on this network graph, the concept of "network flow" can be used to express spatial contiguity in districting [?]. Within a facility' s service area, each unit generates one unit of flow that travels through network edges and ultimately converges at the facility unit within the area. The facility unit does not generate flow but can receive up to n-K units of flow (where K is the number of facilities). The region formed by flow-generating and flow-converging units constitutes a spatially contiguous area.

Let N_j denote the set of units adjacent to unit j, and integer variable f_{ijk} represent the flow from unit j to unit k within facility i' s service area. The spatial contiguity constraints can be formulated as:

$$f_{ijk} \leq n * x_{ij}, \quad \forall i \in I, j \in J, k \in N_j$$

$$f_{ijk} \leq n * x_{ik}, \quad \forall i \in I, j \in J, k \in N_j$$

$$\sum_{k \in N_j} f_{ijk} - \sum_{k \in N_j} f_{ikj} \geq x_{ij}, \quad \forall i \in I, j \in J \setminus i$$

$$f_{ijk} \geq 0, \quad \forall i \in I, j \in J, k \in N_j$$

Constraints (6) and (7) ensure that flow can only occur between two adjacent units belonging to the same service area. For facility i not serving unit j (x_{ij} = 0), unit j generates no flow f_{ijk} = 0 and receives no flow f_{ikj} = 0 within facility i' s service area. When facility i serves unit j (x_{ij} = 1), flow may enter or leave unit j. If unit j is not a sink node (facility unit), constraint (8) guarantees that this unit generates at least one unit of flow. Constraints (6)-(8) cause all flow generated within each facility' s service area to converge at the facility unit. Within a facility service area, each spatial unit must generate flow that travels to neighboring units and ultimately converges at the facility unit, ensuring connectivity between any two units in the service area and thereby guaranteeing spatial contiguity.

When specifying the number of facilities is required, constraint (10) is added to fix the number of facilities at K:

$$\sum_{i \in I} y_i = K$$

Based on these constraints, three SSCFLP extensions exist: (1) SSCFLP with spatial contiguity constraints for facility service areas, denoted as CFLSAP, comprising equations (1)-(9); (2) SSCFLP with facility quantity constraints, denoted as SSCKFLP, comprising equations (1)-(5) and (10); and (3) SSCFLP with both spatial contiguity constraints and facility quantity constraints, denoted as CKFLSAP, comprising equations (1)-(10).

3 Matheuristic Algorithm

Local search algorithms employ neighborhood search to progressively improve current solutions and are widely applied in discrete optimization. In facility location problems, changes to facilities inevitably alter demand assignments, making neighborhood search relatively complex. Therefore, a core challenge in designing location problem algorithms is how to effectively relocate facilities and update assignments to find feasible, cost-saving neighborhood solutions. Due to facility capacity constraints, very large-scale neighborhood search involving multiple facilities is necessary to achieve a high probability of improving the current solution. However, defining the neighborhood structure for SSCFLP is complex, and neighborhood search has high computational complexity, with an exponential number of possible customer reassignments and facility moves [**?**, **?**].

This paper proposes using mathematical models to solve very large neighborhoods. The algorithm principle is as follows: (1) obtain an initial solution using LH methods [**?**], linear relaxation, or heuristic algorithms; (2) randomly select a very large neighborhood from the current solution to obtain subsets of demand points, current facilities, and candidate facilities; (3) construct and solve a subproblem model for the neighborhood and update the current solution; and (4) repeat steps (2) and (3) until no updates occur after several attempts. Step (2) constructs a large neighborhood, while step (3) discovers the optimal solution within that neighborhood. Based on this concept, the matheuristic algorithm is outlined as follows:

Parameters: Maximum consecutive non-improving loops (mloops)
1. s = GenerateInitialSolution();
2. notImpr = 0;
3. While notImpr < mloops do
4. Select a large neighborhood I* and J* from solution s;
5. s* = SolveSubProblem(I*, J*);
6. s' = UpdateSolution(s, s*);
7. s'' = VNDSearch(s');
8. If s'' is better than s then
9. s = s''; notImpr = 0;
10. Else notImpr = notImpr + 1;
11. Output(s).

In this algorithm, step (1) obtains an initial solution. Since many instances are difficult to solve to feasibility, we consider feasibility in the following order: assignment constraints, facility quantity constraints, spatial contiguity constraints, and facility capacity constraints. Given that capacity constraints are particularly difficult to satisfy, our algorithm allows capacity violations but penalizes them in the objective function. We add integer decision variable H_i = {0,1,2, ⋯}, i I to represent the amount by which facility i' s capacity is exceeded, modifying the capacity constraint and objective function to:

$$\sum_{j\in J} d_j x_{ij} \leq s_i y_i + H_i, \quad \forall i \in I$$

$$\text{Minimize: } \sum_{i\in I} f_i y_i + \sum_{i\in I}\sum_{j\in J} c_{ij} x_{ij} + \alpha \sum_{i\in I} H_i$$

This modification allows capacity violations, making it easier to generate initial solutions. By setting the penalty coefficient α sufficiently large, the algorithm gradually reduces H_i values until they reach zero, thereby satisfying capacity constraint (3). This soft capacity constraint also enables MIP solvers to quickly obtain feasible solutions, facilitating exact solution of the model.

Initial solution generation methods may include: Lagrangian relaxation heuristic, linear relaxation heuristic, or ADD/DROP heuristics. The Lagrangian relaxation heuristic algorithm [**?**] can directly solve SSCFLP instances and, by adjusting the number of facilities, can also solve SSCKFLP. The linear relaxation heuristic modifies binary variables x_{ij} to continuous variables in [0,1], solves the model with an optimizer, and repairs the solution. ADD or DROP heuristics are classical constructive algorithms that obtain feasible solutions in a greedy manner. When considering service area contiguity, the results from these algorithms require contiguity repair.

Algorithm step (4) randomly selects a very large neighborhood from the current solution, including a subset of facilities I* and a subset of customers J*. Neighborhood selection is a critical design point, implemented as follows: (1) randomly select a demand point; (2) from the current solution' s facility set, select the Q nearest facilities to this point and their assigned customers; (3) the customers assigned to these Q facilities constitute subset J*; and (4) the nearest facilities to each customer in J* plus the previously selected Q facilities constitute subset I*. The neighborhood size is influenced by parameter Q, which ranges between [Q_{min}, Q_{max}], where Q_{min} = min(L/2, 7) and Q_{max} = min(L, 10), with L being the number of facilities in the current solution. When L $\leq$ 10, Q falls in [L/2, L]; when 11 $\leq$ L $\leq$ 13, Q falls in [L/2, 10]; and when L $\geq$ 14, Q falls in [7, 10].

Neighborhood scale affects algorithm performance: if the neighborhood is too small, facility locations in the current solution are often difficult to improve; if the neighborhood is too large, model computational efficiency suffers. When the size of set I* is excessively large, model solving may become inefficient. Therefore, it is necessary to prune some candidate facilities before subproblem modeling and solving. The pruning method selects Q facilities currently used in the solution from I*, then randomly selects another Q facilities from the remaining facilities, updating set I* to contain 2Q facilities.

Step (5) constructs a subproblem model using facility subset I* and customer subset J*, solved with a MIP optimizer. For the four problems SSCFLP, CFLSAP, SSCKFLP, and CKFLSAP, the first two construct the SSCFLP model while the latter two construct the SSCKFLP model to ensure constraint (10) is satisfied.

Step (6) updates the current solution with the local solution s* to obtain new solution s' . For CFLSAP and CKFLSAP problems, new solution s' may violate spatial contiguity requirements, necessitating repair.

The repair process in step (6) often degrades solution quality, so step (7) employs a Variable Neighborhood Descent (VND) algorithm to improve the current solution. The VND algorithm uses 1-move and 2-move operators [**?**] to obtain local optima.

Step (8) attempts to update the current solution. If new solution s' ' is better than current solution s, update s; otherwise, maintain s. Steps (4)-(9) iterate until the algorithm terminates after several consecutive loops fail to update the current solution.

3.1 Instance Design

This study constructs test instances based on two typical regions. Region ZY is an urban area of approximately 13.4 km^2 containing 326 spatial units, 15 primary schools, and 3,783 students. Region GY is a county-level area of approximately 1,000 km^2 containing 1,276 spatial units, 20 township offices, and a population of 819,812. For region ZY, student enrollment serves as demand, with additional potential facility locations manually added based on existing schools and facility capacities referencing the average of existing schools. For region GY, population serves as demand, with additional locations manually added based on existing township offices and facility capacities referencing the average population of existing offices. Figure 1 [Figure 1: see original paper] illustrates the spatial distribution of service demand and candidate facilities in the case study areas, where light gray circles represent demand and star symbols represent potential facilities.

Figure 1 [Figure 1: see original paper] The study areas ZY (left) and GY (right)

First, we construct two test instances based on data from these geographic regions. In each instance, the centroid of each spatial unit serves as both demand point location and candidate facility location. The cost for facility i to serve customer j is assumed to be c_{ij} = 1.0 * d_{ij} * d_j, where d_{ij} is the Euclidean distance between facility i and customer j in kilometers, and d_j is the demand of customer j. Facility construction cost is f_i = (+ _i) * s_i, where s_i is facility service capacity and _i is a random number within a certain range. For region ZY, = 0.8 and _i is a random number in [-0.1, 0.1]; for region GY, = 1.8 and _i is a random number in [-0.2, 0.2]. Parameter was tuned through multiple trials to ensure facility costs account for 30%-70% of total SSCFLP costs.

Second, we generate test instance sets based on the ZY and GY data. For each instance, facility capacity is scaled by factors of 1.2 and 1.4, and facility costs are increased by factors of 1.1, 1.2, 1.3, and 1.4. Combining capacity and cost scaling yields 15 instances per region. Table 1 presents basic instance information, where the supply-demand ratio equals total capacity of all candidate facilities divided by total demand, and the cost-capacity ratio equals total cost divided by total capacity of candidate facilities. The instance data can be downloaded from https://github.com/yfkong/unified.

Table 1 The benchmark instances

3.2 Instance Testing

The proposed matheuristic algorithm is implemented in Python. The algorithm proceeds as follows: read data, obtain an initial solution using the LH method, and iteratively execute neighborhood selection, subproblem solving, and current solution updating until termination conditions are met. For subproblem solving, the PuLP linear programming modeling tool (https://github.com/coin-or/pulp) constructs SSCFLP or SSCKFLP subproblem models, which are then solved using IBM CPLEX 12.6. To improve computational speed, the algorithm runs in PyPy7 (https://www.pypy.org). The experimental environment is an HP desktop with an Intel Core i7-6700 CPU at 3.40 GHz, 8GB RAM, and Windows 10. The algorithm can be downloaded from https://github.com/yfkong/unified.

All instances are tested for SSCFLP and CFLSAP. For instances zya1 and gya1, we also test SSCKFLP and CKFLSAP, with facility counts K = 13-22 for zya1 and K = 18-30 for gya1.

Two solution approaches are employed: exact methods and our matheuristic algorithm. The exact method constructs models according to our formulations and solves them using CPLEX 12.6, recording lower bounds, upper bounds, and computation times. The matheuristic algorithm uses parameter mloops = 100. Generally, larger values of this parameter require longer computation times. We record objective values, gaps to lower bounds, and computation times.

3.3 Computational Results

Results are presented in Tables 2 , 3 , 4 , and 5 . In these tables, LB represents the lower bound, UB the upper bound, and Gap the relative difference between LB and UB. When Gap = 0, CPLEX has found the optimal solution. The rightmost three columns show the objective value obtained by the matheuristic algorithm, its gap to the lower bound, and computation time.

Table 2 Detailed SSCFLP solutions

The SSCFLP results in Table 2 indicate that CPLEX can solve these instances, obtaining optimal or high-quality approximate solutions (MIPGap < 0.01%), though computation times vary significantly—from 2 minutes to over 2 hours without proving optimality. The matheuristic algorithm demonstrates high computational speed, requiring 1-6 minutes per instance, and achieves high solution quality: it finds optimal solutions for 12 instances, and for remaining instances except zyb1 and gyb3, the objective gaps are less than 0.01%.

Table 3 Detailed CFLSAP solutions

The CFLSAP results show that CPLEX obtains satisfactory results for all instances, with an average Gap of 0.20% and optimal solutions for 5 instances within 2 hours. Compared to Table 2, adding spatial contiguity constraints significantly increases CPLEX computation time. The matheuristic algorithm solves each instance in 1-5 minutes with an average Gap of 0.22%, finding 2 optimal solutions and delivering better solution quality than CPLEX for 12 instances.

Table 4 Detailed SSCKFLP solutions

The SSCKFLP results in Table 4 demonstrate that adding facility quantity constraints substantially reduces CPLEX computation time: optimal solutions are obtained for 20 instances, with the remaining 3 achieving high-quality solutions (Gap $<$ 0.02%), and 18 instances require only 2 minutes. The matheuristic algorithm achieves high solution quality, finding 11 optimal solutions, with one instance better than CPLEX, mean Gap of 0.00%, and maximum Gap of 0.04%. Computation times range from 1-6 minutes.

Table 5 Detailed CKFLSAP solutions

The CKFLSAP results show that even with spatial contiguity constraints, CPLEX can solve instances with high quality, though computation time increases significantly: 16 optimal solutions are found, with only 2 instances having Gap $>$ 0.10%. The matheuristic algorithm rapidly obtains high-quality solutions, finding 6 optimal solutions and outperforming CPLEX on 4 instances.

3.4 Comparative Analysis

Comparing SSCFLP and CFLSAP results reveals several characteristics: (1) Adding spatial contiguity constraints to SSCFLP slightly increases objective values by 0.09%-1.00% (average 0.39%) for ZY instances and 0.01%-0.36% (average 0.13%) for GY instances. (2) After adding spatial contiguity constraints, CPLEX computation time increases significantly while solution quality decreases, with MIPGap rising from 0.00% to 0.20%. (3) Spatial contiguity constraints have no noticeable effect on the matheuristic algorithm' s computation time. (4) Spatial contiguity constraints significantly impact facility locations and service areas. In our test instances, some cases select identical facility locations regardless of contiguity considerations, while others show significant differences between SSCFLP and CFLSAP solutions. For example, in instance zya4, Figure 2 [Figure 2: see original paper] illustrates the solutions. The objective difference is 0.39% and both use 19 facilities, but 3 facility locations differ; the SSCFLP solution has 4 non-contiguous service areas, while CFLSAP ensures all service areas are spatially contiguous.

Figure 2 [Figure 2: see original paper] SSCFLP (left) and CFLSAP (right) solutions on instance zya4

Facility quantity constraints significantly impact objective values. First, adding facility quantity constraints to either SSCFLP or CFLSAP substantially reduces exact algorithm computation time, likely because the constraint dramatically reduces the number of feasible facility combinations and shrinks the solution space. CPLEX results show that SSCKFLP is easier to solve to optimality than SSCFLP, and CKFLSAP is easier than CFLSAP. Second, SSCFLP and CFLSAP have optimal facility numbers; deviating from this optimal number increases the objective value. Figure 3 [Figure 3: see original paper] shows SSCKFLP objective values versus facility count K, with the upper plot for instance zya1 and lower for gya1, where the optimal facility counts in SSCFLP solutions are 19 and 20, respectively.

Figure 3 [Figure 3: see original paper] SSCKFLP objectives on instances zya1 (upper) and gya1 (lower) versus the number of facilities

With fixed demand distribution and candidate facility locations, facility capacity and fixed costs influence location decisions to some extent. In our instances, scaling facility capacity uniformly typically changes facility selections, while uniformly scaling facility costs leaves selections unchanged in some cases. We observe identical facility selections across the following instance groups: zya1, zya2, and zya3; zyb1, zyb2, zyb3, zyb4, and zyb5; zyc1, zyc2, zyc3, zyc4, and zyc5; gya2 and gya3; gya4 and gya5; gyb2 and gyb3; gyc1, gyc2, and gyc3.

4 Conclusion

This paper extends the classical SSCFLP by adding spatial contiguity constraints for facility service areas and facility quantity constraints, yielding three variants: SSCKFLP, CFLSAP, and CKFLSAP. Leveraging facility locations and adopting network flow models [**?**], we construct CFLSAP and CKFLSAP models that incorporate spatial contiguity. While flow models for p-regions districting problems can only be solved exactly for very small instances, our CFLSAP and CKFLSAP models are relatively easier to solve because SSCFLP-selected facility locations can be directly used to establish flow models. Experiments show that CPLEX can solve instances with 1,276 units and 33 facilities, obtaining optimal or high-quality solutions (MIPGap < 0.44%). Practical SSCFLP applications may require either contiguous service areas or specified facility numbers, and our extended models address these needs.

We design a matheuristic algorithm to solve SSCFLP, SSCKFLP, CFLSAP, and CKFLSAP. The matheuristic is essentially a very large-scale neighborhood search: starting from an initial solution, it iteratively improves extra-large neighborhoods within the current solution using mathematical models until the solution cannot be further updated. Instance testing demonstrates that the matheuristic solves these problems with average gaps to lower bounds of 0.01%, 0.00%, 0.22%, and 0.08%, respectively—solution quality comparable to CPLEX but with substantially improved computational efficiency. These tests validate the effectiveness of our matheuristic algorithm.

Our case studies also reveal that adding spatial contiguity constraints to SSCFLP or SSCKFLP yields only modest objective value increases but may alter optimal facility locations. Adding facility quantity constraints to SSCFLP or CFLSAP significantly impacts objective values—the greater the deviation from the original optimal facility count, the larger the objective increase. With fixed demand distribution and candidate locations, facility capacity and fixed costs have some influence on location decisions.

References

[1] Aardal K, van den Berg P L, Gijswijt D, et al. Approximation algorithms for hard capacitated k-facility location problems. *European Journal of Operational Research*, 2015, 242(2):358-368.

[2] Agar M, Salhi S. Lagrangean heuristics applied to a variety of large capacitated plant location problems. *Journal of the Operational Research Society*, 1998, 49:1072–1084.

[3] Ahuja R K, Orlin J B, Sharma D. Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem. *Mathematical Programming*, 2001, 91:71–

[4] Avella P, Boccia M. A cutting plane algorithm for the capacitated facility location problem. *Computational Optimization and Applications*, 2009, 43 (1): 39-65.

[5] Barceló J, Casanovas J. A heuristic Lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 1984, 15(2):212–226

[6] Barceló J, Fernández E, Jörnsten K. Computational results from a new Lagrangean relaxation algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 1991, 53(1):38–45.

[7] Basu S, Sharma M & Ghosh P S. Metaheuristic applications on discrete facility location problems: a survey. *OPSEARCH*, 2015, 52, 530–561.

[8] Beasley J E. Lagrangian heuristics for location problems. *European Journal of Operational Research*, 1993, 65:383–399.

[9] Beasley J E. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 1990. 41:1069–1072.

[10] Boccia M, Sforza A, Sterle C, Vasilyev I. A cut and branch approach for the capacitated p-median problem based on Fenchel cutting planes. *Journal of Mathematical Modelling and Algorithms*, 2008, 7(1):43–58.

[11] Caserta M & Voß S. A general corridor method-based approach for capacitated facility location. *International Journal of Production Research*, 2020, 58:13: 3855-3880.

[12] Chen C.H., C.J. Ting. Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. *Transportation Research Part E: Logistics and Transportation Review*, 2008, 44 (6): 1099-1122.

[13] Contreras I A, Díaz J A. Scatter search for the single source capacitated facility location problem. *Annals of Operations Research*, 2008, 157 (1):73-89.

[14] Cornuejols G, Sridharan R, Thizy J. A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Research*, 1991, 50(3):280–297.

[15] Cortinhal M J, Captivo M E. Upper and lower bounds for the single source capacitated location problem. *European Journal of Operational Research*, 2003, 151(2): 333-351.

[16] Duque J C, Church R L, Middleton R S, et al. The p-Regions problem. *Geographical Analysis*, 2011, 43(1): 104-126.

[17] Delmaire H, Díaz J A, Fernández E, Ortega M. Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. *Infor-Information Systems and Operational Research*, 1999, 37(3): 194-225.

[18] Díaz J, Fernández E. A branch-and-price algorithm for the single source capacitated plant location problem. *Journal of the Operational Research Society*, 2002, 53:728–740.

[19] Filho V J M F, Galvao R D. A tabu search heuristic for the concentrator location problem. *Location Science*, 1998, 6: 189–209.

[20] Gadegaard S L, Klose A, Nielsen L R. An improved cut-and-solve algorithm for the single-source capacitated facility location problem. *EURO Journal on Computational Optimization*, 2018, 6: 1–27.

[21] Guastaroba G, Speranza M G. A heuristic for BILP problems: The Single Source Capacitated Facility Location Problem, *European Journal of Operational Research*, 2014, 238 (2): 438-450.

[22] Hindi H, Pieńkosz K. Efficient solution of large scale, single-source, capacitated plant location problems. *Journal of the Operational Research Society*, 1999, 50:268–274

[23] Holmberg K, Rönnqvist M, Yuan D. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, 1999, 113 (3): 544-559.

[24] Holt J, Ronnqvist M, Tragantalerngsak S. A repeated matching heuristic for the single-source capacitated facility location problem. *European Journal of*

Operational Research, 1999, 116:51–68.

[25] Klincewicz J., Luss H. A Lagrangean relaxation heuristic for capacitated facility location with single-source constraints. *Journal of the Operational Research Society*, 1986, 37:495–

[26] Klose A, Görtz S. A branch-and-price algorithm for the capacitated facility location problem. *European Journal of Operational Research*, 2007, 179(3):1109–1125.

[27] Kong Y F, Zhu Y F, Wang Y J. A hybrid metaheuristic algorithm for the school districting problem. *Acta Geographica Sinica*, 2017, 72(02):256-268. [孔云峰, 朱艳芳, 王玉璟. 学校分区问题混合元启发算法研究 [J]. 地理学报, 2017, 72(2): 256-268.]

[28] Neebe A W, Rao M R. An algorithm for the fixed-charge assigning users to sources problem *The Journal of the Operational Research Society*, 1983, 34:1107–1113.

[29] Oliveira Ó, Matos T, Gamboa D. A RAMP Algorithm for Large-Scale Single Source Capacitated Facility Location Problems. In: Matsatsinis N, Marinakis Y, Pardalos P. (eds) *Learning and Intelligent Optimization. LION 2019. Lecture Notes in Computer Science*, 2020, vol 11968, Springer, Cham.

[30] Rönnqvist M, Tragantalerngsak S, Holt J. A repeated matching heuristic for the single-source capacitated facility location problem. *European Journal of Operational Research*, 1999, 116(1):51–68.

[31] Shirabe T. Districting modeling with exact contiguity constraints. *Environment and Planning B: Planning and Design*, 2009, 36(6): 1053-1066.

[32] Sridharan R. A Lagrangian heuristic for the capacitated plant location problem with single source constraints. *European Journal of Operational Research*, 1993, 66(3):305–312.

[33] Tran T H, Scaparra M P, O' Hanley J R. A hypergraph multi-exchange heuristic for the single-source capacitated facility location problem, *European Journal of Operational Research*, 2017, 263(1): 173-187.

[34] Ulukan Z, Demircioğlu E. A Survey of Discrete Facility Location Problems. *International Journal of Industrial and Manufacturing Engineering*, 2015, 9(7): 2487-2492.

[35] Wang, Y., Xu, D., Du, D. et al. An approximation algorithm for k-facility location problem with linear penalties using local search scheme. *J Comb Optim* 36, 264–279 (2018). https://doi.org/10.1007/s10878-016-0080-2

[36] WANG W. A Local Search Algorithm for Hard Uniform Capacitated k-facility Location Problem. Nanjing Normal University, 2017.

[37] Yang Z, Chu F, Chen H. A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research*, 2012, 221 (3): 521-