A Hybrid Evolutionary Algorithm for Reliable Facility Location Problem

Han Zhang, Jialin Liu, and Xin Yao(✉)

Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
11849181@mail.sustech.edu.cn, {liujl,xiny}@sustech.edu.cn

Abstract. The reliable facility location problem (RFLP) is an important research topic of operational research and plays a vital role in the decision-making and management of modern supply chain and logistics. Through solving RFLP, the decision-maker can obtain reliable location decisions under the risk of facilities' disruptions or failures. In this paper, we propose a novel model for the RFLP. Instead of assuming allocating a fixed number of facilities to each customer as in the existing works, we set the number of allocated facilities as an independent variable in our proposed model, which makes our model more close to the scenarios in real life but more difficult to be solved by traditional methods. To handle it, we propose EAMLS, a hybrid evolutionary algorithm, which combines a memorable local search (MLS) method and an evolutionary algorithm (EA). Additionally, a novel metric called l3-value is proposed to assist the analysis of the algorithm's convergence speed and exam the process of evolution. The experimental results show the effectiveness and superior performance of our EAMLS, compared to a CPLEX solver and a Genetic Algorithm (GA), on large-scale problems.

Keywords: Reliable facility location problem · Integer programming · Hybrid algorithm · Evolutionary algorithm · Local search

1 Introduction

The facility location problem aims at finding the optimal locations for facilities from a set of candidate location nodes in order to minimize the cost such as the fixed facility cost and the transposition cost, or to maximize the total revenue. In general, there are also some constraints to be considered, such as satisfying all customers' demands, etc. It is an NP-hard optimization problem [1–3] and

This work was supported by the National Key R&D Program of China (Grant No. 2017YFC0804003), the National Natural Science Foundation of China (Grant No. 61976111, 61906083), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. JCYJ20190809121403553), the Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

T. Bäck et al. (Eds.): PPSN 2020, LNCS 12270, pp. 454–467, 2020.
https://doi.org/10.1007/978-3-030-58115-2_32

has attracted much attention from researchers in both the scientific community and engineering field due to its wide application in real world. The facilities could be hospitals, restaurants, post stations, bus stations, industrial plants, banks, warehouses, and distribution centers, etc. The facility location decision has high precedence in the whole logistics decisions and has a great influence on subsequent operation level decisions [4]. Daskin et al. [1] regards the location decisions as "the most critical and most difficult of the decisions needed to realize an efficient supply chain".

In RFLP, the facility is not always available all the time [1]. One or more of them may not work from time to time because of disruptions, examples include natural disasters, inclement weather, destruction of facilities by fire or flood, expiration of the contract, and any other force majeure factors. In such a situation, these are facility "failures". The failures of the facilities will result in excessive transportation costs because the customers that were considered to be served by them must be served by other, usually more distant, facilities [1]. Therefore, by solving RFLP, we can get a location decision which can ensure a certain level of reliability to guarantee customers can get service when facilities' failures occur.

Many models have been proposed for RFLP, in which all kinds of factors were taken into account and many of them are formulated for specific applications in real life. In addition, large-scale RFLP problems have rarely been considered. The algorithms studied in literature were mainly tested on problems of small size.

This paper focuses on two aspects: the problem formulation and the algorithm. Based on the work of [5,6], we propose a new reliable facility location-allocation problem (RFLP) formulation, which does not fix the number of allocated facilities to each customer as a constant and is more close to reality. The resulted model is a nonlinear 0–1 integer programming model which is more complicated for traditional methods. In this paper, a hybrid evolutionary algorithm called EAMLS is proposed to solve it. EAMLS combines a memorable local search method with an evolutionary algorithm, which has a good performance on both small-scale and large-scale problems considered in this paper. It is worth mentioning that the instances used in our experiments are much larger than the ones used in previous work. Furthermore, a convergence metric l3-value is proposed for analyzing the algorithm and observing the evolutionary process.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work of RFLP. In Sect. 3, our new RFLP formulation is introduced. We proposed a hybrid evolutionary algorithm EAMLS in Sect. 4. Section 5 presents computational studies, and Sect. 6 concludes.

2 Related Work

By solving a specific RFLP, decision-makers expect to get a robust location decision which is still economical when some facilities fail under various disruptions. The research can be divided into two categories according to the method used to handle facility failure or ensure reliability.

Some works [7–9] use a disruptive scenarios approach to describe facility failure. In this approach, scenarios contain facility failure information, e.g., simultaneously disrupted facility sites, modified customer demands, and facility costs, etc. The disruptive scenarios approach can describe the facility failure information well, but it usually requires plenty of scenarios to cover different disruptive situations, which implies large computational cost, especially for large-scale problems.

Another approach to ensure reliability is to allocate two or more facilities to serve each customer [5,6,10,11]. In this approach, the method for reliability is intuitive and easy to understand. Both a location decision (which contains how many facilities needed to build and where to build them) and an allocation decision (which shows how to allocate facilities to serve customers) are determined before the occurrences of facilities' disruptions/failures.

Some RFLP models have been proposed, e.g., models proposed by Li et al. [5] and Snyder and Darskin [6]. Table 1 summarizes the notations used in the models.

Table 1. Description of notations.

Notations	Description	Notations	Description
I	the set of customers, index by i;	m	# of facilities allocated for each customer;
J	the set of candidate location sites, index by j;	p	the facility failure probability;
NF	the set of candidate location sites that will not fail;	f_i	the fix cost of j;
F	the set of candidate location sites that may fail;	α	weighted parameter;
c_{ij}	the cost of per unit demand shipped from j to i;	h_i	the demands of customer i;

Besides, there are two sets of decision variables: location decision variables (**X**) and allocation decision variables (**Y**):

$$X_j = \begin{cases} 1, & \text{if candidate location site } j \text{ is selected;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$Y_{ijr} = \begin{cases} 1, & \text{if } j \text{ is allocated as the level-}r\text{ facility to serve } i\text{;} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In Eq. (2), the "level-r" facility j for customer i means the facility j will provide service only when the front r allocated facilities (from level-0 to level-(r-1)) fail.

A classical RFLP model in [6] is as follows.

$$\text{Min } \alpha w_1 + (1-\alpha) w_2 \tag{3}$$

Subject to:

$$w_1 = \sum_{j \in J} f_j X_j + \sum_{i \in I} \sum_{j \in J} h_i c_{ij} Y_{ij0} \tag{4}$$

$$w_2 = \sum_{i \in I} h_i \left[\sum_{j \in NF} \sum_{r=0}^{m-1} c_{ij} p^r Y_{ijr} + \sum_{j \in F} \sum_{r=0}^{m-1} c_{ij} p^r (1-p) Y_{ijr} \right] \tag{5}$$

$$\sum_{j \in J} Y_{ijr} + \sum_{j \in NF} \sum_{t=0}^{r-1} Y_{ijt} = 1 \quad \forall i \in I, r = 0, \ldots, m-1 \tag{6}$$

$$Y_{ijr} \leq X_j \quad \forall i \in I, j \in J, r = 0, \ldots, m-1 \tag{7}$$

$$\sum_{r=0}^{m-1} Y_{ijr} \leq 1 \quad \forall i \in I, \forall j \in J \tag{8}$$

$$m = |J| \tag{9}$$

$$X_u = 1 \tag{10}$$

$$X_j \in \{0, 1\} \quad \forall j \in J \tag{11}$$

$$Y_{ijr} \in \{0, 1\} \quad \forall i \in I; \forall j \in J; r = 0, \ldots, m-1 \tag{12}$$

In this model, there are two objectives in the objective function, w_1 is the operating cost and w_2 is the expected failure cost. The objective of the model is to minimize the weighted sum of the two objectives. Besides, there is an emergency facility u which will always be selected and not fail, and all customers can get service from it.

Several shortcomings are observed in the literature:

(1) The number of facilities allocated to each customer (i.e., m in Eq. (9)) is fixed in models of most literature, e.g., $m = 2$ (i.e., Y_{ij0} and Y_{ij1}) in [5] and $m = |J|$ in [6]. One issue of this allocation setting is the determination of an appropriate value of m. If m is bigger than the number of selected candidate location sites, i.e., $\sum_{j \in J} X_j$, it is not in line with the actual situation because we cannot allocate nonexistent facilities to customers. If we set the value of m smaller than $\sum_{j \in J} X_j$, the value of $\sum_{j \in J} X_j$ is changed during the exploration in solution space, therefore it is hard for us to set a suitable m value. If we set $m = 2$ directly, which means allocate just one primary facility and one backup facility to serve each customer, the reliability is a bit weak intuitively.

(2) To our best knowledge, there is a lack of research on the large-scale problem. The largest problem instance in the related research is 150-node and the optimization solver such as CPLEX can find near-optimal or even optimal solutions for the problem.

(3) There is a lack of research on the algorithm which can solve the large-scale problems efficiently as well.

Correspondingly, this paper:

(1) constructs a new formulation in which a non-fixed allocation setting, i.e., $m = \sum_{j \in J} X_j$, is used;

(2) proposes a hybrid evolutionary algorithm EAMLS which combines a local search method with an evolutionary algorithm and performs well on both small-scale and large-scale problems;

(3) performs experimental studies on large-scale problems whose scale is much larger than any related literature;

(4) proposes a convergence metric l3-value to help observe the evolutionary process, adjust parameters and further improve the algorithm.

3 Problem Formulation

We propose a new RFLP formulation in which we set the number of allocated facilities to each customer as an variable instead of a fixed constant.

The mathematical formulation of our model is as follows, formulated based on [5,6]. The decision variables are defined by Eqs. (1) and (2).

$$\text{Min} \sum_{j\in J} f_j X_j + \alpha \sum_{i\in I}\sum_{j\in J}\sum_{r=0}^{m-1} h_i c_{ij} p^r (1-p) Y_{ijr} \tag{13}$$

Subject to:

$$m = \sum_{j\in J} X_j \tag{14}$$

$$m \geq 2 \tag{15}$$

$$\sum_{j\in J} Y_{ijr} = 1 \quad \forall i \in I; r = 0, \ldots, m-1 \tag{16}$$

$$\sum_{r=0}^{m-1} Y_{ijr} \leq X_j \quad \forall i \in I, \forall j \in J \tag{17}$$

$$X_j \in \{0,1\} \quad \forall j \in J \tag{18}$$

$$Y_{ijr} \in \{0,1\} \quad \forall i \in I; \forall j \in J; r = 0, \ldots, m-1 \tag{19}$$

The objective function of the model is to minimize the total cost associate with facilities construction (i.e., the term $\sum_{j\in J} f_j X_j$) and transportation between the facilities and customers (i.e., the term $\sum_{i\in I}\sum_{j\in J}\sum_{r=0}^{m-1} h_i c_{ij} p^r (1-p) Y_{ijr}$).

Constraint (14) makes the number of facilities allocated to each customer (i.e., m) a variable and its value is related to location decision variables (i.e., $\mathbf{X}$). Constraint (15) represents at lease two facilities are constructed to ensure reliability. Constraint (16) assures only one facility can be the level-r supplier of customer i. Constraint (17) means candidate location site j can be allocated to customer as a supplier only when it is selected. Constraint (18) and (19) are standard integrality constraints.

Compared with classical models shown in Sect. 2, the significant difference in our model is the new non-fixed facility allocation setting, i.e., constraint (14). In our model, the value of m is not fixed but varies with decision variables $\mathbf{X}$, therefore it is more realistic, ensures reliability, but makes our model much more complex and difficult to solve by traditional methods as well.

4 A Hybrid Evolutionary Algorithm: EAMLS

This paper develops a new hybrid evolutionary algorithm EAMLS (Evolutionary Algorithm with Memorable Local Search) which combines a memorable local search method and an EA, and a convergence metric l3-value is proposed. In this section, the structure of EAMLS is explained first, then the design of operators of the Genetic Algorithm (GA) and EAMLS is introduced. Finally, the details of l3-value are described.

4.1 EAMLS

Algorithm 1 is the pseudo-code of EAMLS. Compared with the GA, the main characters of EAMLS contain: (1) no crossover operation; (2) population size self-adaptation; (3) the combination of a memorable local search (MLS) and EA; and (4) the adoption of convergence metric l3-value.

In Algorithm 1, variable $allNeighborInds$ stores all non-repeating neighborhood individuals generated by MLS before current generation and is updated at the end of every generation (Algorithm 1, Line 2 and Line 13). In the evolutionary process, a new population is generated from the current population after mutation, MLS, and survival selection (Algorithm 1, Lines 5–8), and convergence metric l3-value is calculated (Algorithm 1, Line 9). If l3-value is bigger than a pre-set threshold β, population size is increased by a pre-set step size p (Algorithm 1, Lines 10–12). The description of the l3-value will be shown in Sect. 4.3.

Algorithm 2 is the pseudo-code of the memorable local search (MLS). First, we will introduce the definition of the neighborhood. The neighborhood of an individual is the set of individuals whose Hamming distance is 1 from that

Algorithm 1. Evolutionary Algorithm with Memorable Local Search.

```
Input: $G$: number of generations; $\mu$: population size; $l$: individual length; $m$: mutation
    rate; $\beta$: threshold of l3-value; $p$: step size of population self-adaptation;
Output: $bestSol$: the best individual in the final population;
 1: $initPop \leftarrow initializePop(\mu, l)$;
 2: $allNeighborInds \leftarrow$ an empty set;
 3: $pop \leftarrow evaluatePop(initPop)$;
 4: for $g = 1$ to $G$ do
 5:     $popAfterMutation \leftarrow mutation(pop, m)$;
 6:     $offspring \leftarrow evaluatePop(popAfterMuation)$;
 7:     $offspring_{LS} \leftarrow memorableLocalSearch(pop, offspring)$;
 8:     $pop \leftarrow survival(pop, offspring, offspring_{LS}, \mu)$;
 9:     l3-value $\leftarrow getl3Value(pop, allNeighborInds)$;
10:     if l3-value$> \beta$ then
11:         $\mu \leftarrow \mu + p$;
12:     end if
13:     add $offspring_{LS}$ to $allNeighborInds$;
14: end for
15: $bestSol \leftarrow selectBestIndividual(pop)$
16: Return $bestSol$
```

Algorithm 2. Memorable Local Search.

```
Input: $pop$: the parent population; $offspring$: the child population generated after
    mutation; $n$: # of individuals which need to check whether to do local search;
    $indLSed$:the set of individuals which have already down local search before this
    generation;
Output: $offspring_{LS}$: the population generated by local search;
 1: $offspring_{LS} \leftarrow$ an empty set;
 2: $parentPop \leftarrow$ combine $pop$ and $offspring$;
 3: $sortedParentPop \leftarrow$ sort $parentPop$ by fitness increasing order;
 4: $i \leftarrow 0$;
 5: for $j \leftarrow 1$ to $len(sortedParentPop)$ do
 6:     if $sortedParentPop[j]$ not in $indLSed$ then
 7:         $neighborInds \leftarrow generateNeighbor(sortedParentPop[j])$;
 8:         add $neighborInds$ to $offspring_{LS}$;
 9:         $i \leftarrow i + 1$;
10:         if $i > n$ then
11:             break;
12:         end if
13:     end if
14: end for
15: Return $offspring_{LS}$
```

individual. In MLS, sort $(\mu + \lambda)$ population (variable *sortedParentPop* in Algorithm 2) in decreasing order, i.e., good individuals are in the front. Then check individuals one by one in sorted $(\mu + \lambda)$ population whether it has been local-searched before this generation, and do local-search for those have not been local-searched (Lines 5–7 in Algorithm 2. It looks like that the algorithm remembers all local-searched individuals and that's why we name it Memorable Local Search). Exit the loop until the number of new individuals which have been local-searched in this generation reaches n (Lines 9–12 in Algorithm 2).

4.2 Operator Design of GA and EAMLS

In Sect. 5, we use a GA for comparison. Here some operators' design for GA and EAMLS is as follows[1]:

Representation. This paper uses binary representation. Every bit represents a location decision variable $X_j, j \in J$.

Population Initialization. Stochastic initialization is used in GA and EAMLS. Every gene of an individual takes 0 or 1 with equal probability.

Fitness Function. In general, the bigger the fitness value is, the better the individual will be. Therefore, the reciprocal of the objective value of the individual is used as the fitness function.

[1] If there is no special statement, that operator is adopted in both GA and EAMLS.

Selection Operator. In GA, roulette wheel selection is used to select parents to do crossover operation.

Crossover Operator. In GA, a one-point crossover operator is used. For two parent individuals selected by the selection operator, do crossover operation according to a pre-set crossover rate.

Mutation Operator. The bit-flipping mutation is used in GA and EAMLS. During mutation, every gene/bit of one individual mutates with a pre-set mutation rate.

Survival Selection Strategy. We adopt $(\mu + \lambda)$ strategy to select next generation population from $(\mu + \lambda)$ population, i.e., the mixed population of the current generation population and the offspring.

Repair Strategy. Repair strategy is working when there are individuals which do not satisfy the constraint (15). For an individual needed repair, check every gene in ascending order of fixed cost and change the gene with 0-value to 1 until the individual satisfies the constraint (15).

How to Determine$\mathbf{Y}$. For one customer, the selected candidate locations (i.e., locations whose $X_j = 1$) are allocated to it in ascending order of distance, which has been proved the optimal allocation pattern under a certain solution $\mathbf{X}$ [6] and can satisfy the constraints (12), (13), and (15).

4.3 Convergence Metric l3-Value

In order to observe the evolutionary process, a convergence metric l3-value is proposed.

Algorithm 3 is the pseudo-code of the calculation method of l3-value. The new population generated after survival selection is checked, and the number of individuals which also belong to the set *allNeighborInds* is counted (Lines 2–6 in Algorithm 3). Then we calculate the proportion of these individuals in the population as l3-value (Line 7 in Algorithm 3). l3-value can be used to measure the convergence during the evolutionary process. The bigger the l3-value is, the stronger the evolution converges.

Algorithm 3. Function $getl3Value()$.

```
Input: pop: the new population after survival selection; allNeighborInds: the set of
    all individuals generated by memorable local search before this generation;
Output: l3-value;
1: $num \leftarrow 0$;
2: for $ind \in pop$ do
3:     if $ind \in allNeighborInds$ then
4:         $num \leftarrow num + 1$;
5:     end if
6: end for
7: l3-value $\leftarrow num/len(pop)$;
8: Return $l3\text{-}value$
```

5 Computational Studies

Because this paper proposes a new problem, and there are not any algorithms like EAMLS can be used to compare directly, we compare EAMLS with a GA and CPLEX (a commercial optimization solver of IBM) on two models: m = 2 and m = $\sum_{j\in J} X_j$ models. The difference between the two models is the allocation setting. In the m = 2 model, the number of facilities allocated to each customer, i.e. m, is fixed to 2, which is adopted in much literature. The m = $\sum_{j\in J} X_j$ model is proposed by us in this paper and m varies with decision variables $\mathbf{X}$ during the search process. Section 5.1 shows the experimental design, including instances generation, parameters setting, and experimental environment. The experiments and results of the m = 2 and m = $\sum_{j\in J} X_j$ models are presented in Sect. 5.2. Analyses and discussions are given in Sect. 5.3.

5.1 Experimental Design

Instance Generation. This paper generates problem instances uniformly at random on different scales. The parameters used to generate instances are shown in Table 2. There are eight 10-node instances, eight 50-node instances, eight 100-node instances, and four 600-node instances.

Table 2. Parameters used in instances generation

Parameters	Ranges
Candidate location coordinate	[0,1]
Customer demands	{0,1,...,1000}
Fixed cost of facility	{500,501,...,1500}
Facility failure probability	0.05

Parameter Setting of Algorithms. Some parameters' values of GA and EAMLS are shown in Table 3. Table 4 presents the generation number and population size of GA and EAMLS, which associate with the scale of problem instances. The values of parameters in Tables 3 and 4 are chosen arbitrarily on the basis of meeting the following conditions: (1) EAMLS converges at the end of evolution; (2) the number of fitness evaluations (FEs) of GA is not lower than EAMLS. Besides, the default parameters of CPLEX are used.

Experimental Environment. The algorithms are implemented in Python 3.7 and run on Dell R370 server which has 2x Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20 GHz CPU, 128G RAM, and CentOS 7.6 operating system.

Statistical Test. We use the Wilcoxon sign rank test to determine whether the results between EAMLS and other methods have statistically significant differences. The Wilcoxon sign rank test is a non-parameter test which is suitable for two related or matched samples and compares data in pair, hence it is suitable to use here.

Table 3. Some parameters of GA and EAMLS

Parameters	Value
Crossover rate for GA, c	0.9
Mutation rate, m	0.1
# Local search individual, n	10
l3-value threshold, β	0.8
Step size of population self-adaption, p	100

Table 4. Parameters associate with instance size

Instance scale (# nodes)	GA		EAMLS	
	# Generation	Population size	# Generation	Population size
10	60	30	10	20
50	200	200	20	20
100	400	200	50	100
600	4600	200	250	200

5.2 Experiments on the $\mathbf{m = 2}$ and $\mathbf{m = \sum_{j \in J} X_j}$ Models

For the m = 2 model, We compare EAMLS with the GA and CPLEX on small-scale (10-node), mid-scale (100-node), and large-scale (600-node) instances. There are 30 runs on small and mid-scale instances and 10 runs on large-scale instances because of time. The computational results are shown in Table 5.

For the $\mathrm{m} = \sum_{j \in J} X_j$ model, we compare EAMLS with the GA and CPLEX on 50 and 100-node instances, and there are 30 runs on each instance. Table 6 is the computational results.

5.3 Analyses and Discussions

We compare GA, CPLEX, and EAMLS on different scale (10, 100, and 600-node) problem instances for m = 2 model whose allocation setting is often used in literature, and the experimental results are shown in Table 5. Experimental results on 50 and 100-node instances of the new complicated $\mathrm{m} = \sum_{j \in J} X_j$ model are presented in Table 6.

For m = 2 model, from Table 5, we can see that CPLEX performs the best on both solution quality and time for small and mid-scale (10 and 100-node) instances. EAMLS can find solutions as good as CPLEX but need more time. Although CPLEX can solve small and mid-scale instances fast, it needs more RAM space as the problem scale increases. For large-scale problem (600-node) instances, EAMLS can find better solutions in less time compared with GA, while the CPLEX cannot find a solution.

The new $\mathrm{m} = \sum_{j \in J} X_j$ model is more complicated to solve, especially for CPLEX. Table 6 demonstrates that the performance of EAMLS is better than GA and CPLEX on both solution quality and time.

Table 5. Computational results on m = 2 model 10 (30 runs),100 (30 runs), and 600 (10 runs)-node instances. AOV is Average Objective Value. OR is the Optimal Rate and calculated by (# runs which finding the optimal solution)/(# all runs). Gap is calculated by (AOV(other method)-AOV(EAMLS))/AOV(EAMLS). When Gap is positive, the performance of other methods is worse than EAMLS, otherwise better. The symbol "*" in AOV represents the results between EAMLS and that method have statistically significant differences. The symbol "-" represents CPLEX cannot solve the instance or the optimal solution is unknown so no results can be given.

Instance No.	GA				CPLEX				EAMLS			
	AOV	Gap (%)	OR	Time	AOV	Gap (%)	OR	Time	AOV	Gap (%)	OR	Time
10-1	**2463.19**	0.00	**1.00**	1.52	**2463.19**	0.00	1.00	**0.46**	**2463.19**	0.00	1.00	6.14
10-2	**2874.03**	0.00	**1.00**	1.51	**2874.03**	0.00	1.00	**0.41**	**2874.03**	0.00	1.00	5.46
10-3	**2623.35**	0.00	**1.00**	1.74	**2623.35**	0.00	1.00	**0.66**	**2623.35**	0.00	1.00	5.41
10-4	**2323.92**	0.00	**1.00**	1.93	**2323.92**	0.00	1.00	**0.48**	**2323.92**	0.00	1.00	5.86
10-5	**2917.87**	0.00	**1.00**	2.48	**2917.87**	0.00	1.00	**0.50**	**2917.87**	0.00	1.00	5.71
10-6	**3149.31**	0.00	**1.00**	2.72	**3149.31**	0.00	1.00	**0.41**	**3149.31**	0.00	1.00	5.59
10-7	**3324.98**	0.00	**1.00**	2.39	**3324.98**	0.00	1.00	**0.58**	**3324.98**	0.00	1.00	5.64
10-8	**3165.87**	0.00	**1.00**	2.10	**3165.87**	0.00	1.00	**0.52**	**3165.87**	0.00	1.00	4.58
100-1	13029.83*	22.28	0.00	2374.86	**10645.89**	–0.10	1.00	**14.30**	10656.11	0.00	0.87	1431.17
100-2	13166.44*	20.95	0.00	2375.71	**10885.43**	0.00	1.00	**14.31**	**10885.43**	0.00	1.00	1387.01
100-3	12982.37*	16.90	0.00	2396.42	**11105.21**	0.00	1.00	**14.76**	11105.39	0.00	0.93	1514.12
100-4	13379.41*	16.66	0.00	2388.67	**11468.64**	0.00	1.00	**14.42**	**11468.64**	0.00	1.00	1382.91
100-5	14563.46*	16.39	0.00	2398.34	**12505.51**	–0.05	1.00	**14.80**	12512.29	0.00	0.90	1415.63
100-6	13189.74*	17.29	0.00	2402.44	**11245.55**	0.00	1.00	**14.00**	**11245.55**	0.00	1.00	1447.11
100-7	12841.37*	16.11	0.00	1696.85	**11043.70**	–0.15	1.00	**15.49**	11059.89	0.00	0.90	1326.41
100-8	13886.78*	18.30	0.00	1242.25	**11732.46**	–0.05	1.00	**14.94**	11738.83	0.00	0.87	1180.91
600-1	144896.91*	281.04	–	655420.67	–	–	–	–	**38026.65**	0.00	–	**564432.00**
600-2	145508.23*	293.12	–	656832.50	–	–	–	–	**37013.71**	0.00	–	**572568.34**
600-3	141486.28*	283.41	–	654632.01	–	–	–	–	**36902.36**	0.00	–	**568824.96**
600-4	141256.35*	282.80	–	656656.21	–	–	–	–	**36900.52**	0.00	–	**568546.96**

Table 6. Computational results on $m = \sum_{j \in J} X_j$ model 50 and 100-node instances, 30 runs. AOV is Average Objective Value. Gap is calculated by ((AOV(other method)-AOV(EAMLS))/AOV(EAMLS). When Gap is positive, the performance of other methods is worse than EAMLS, otherwise better. The symbol "*" in AOV represents the results between EAMLS and that method have statistically significant differences.

Instance No.	GA			CPLEX			EAMLS		
	AOV	Gap (%)	Time	AOV	Gap (%)	Time	AOV	Gap (%)	Time
50-1	7053.71*	0.68	719.76	12589.41*	79.69	4715.56	**7006.23**	0.00	**91.52**
50-2	7154.93	–0.13	720.49	15734.80*	119.63	4488.26	**7164.20**	0.00	**90.75**
50-3	6890.54*	0.75	713.25	12656.13*	85.06	5219.33	**6838.95**	0.00	**91.10**
50-4	7166.63	0.04	698.45	12147.92*	69.58	4702.01	**7163.42**	0.00	**90.04**
50-5	6929.29	0.03	714.86	11946.35*	72.46	5281.27	**6926.95**	0.00	**87.42**
50-6	6575.09	0.29	696.80	13284.69*	102.64	4836.45	**6555.87**	0.00	**90.59**
50-7	7162.83	0.07	685.04	12441.00*	73.81	4495.41	**7157.76**	0.00	**81.19**
50-8	7175.89*	0.26	629.19	14433.41*	101.67	4522.35	**7156.99**	0.00	**70.07**
100-1	12895.10*	20.47	3976.97	113781.45*	963.00	19451.62	**10703.78**	0.00	**2266.50**
100-2	13093.80*	19.77	3820.11	110441.89*	910.18	17159.57	**10932.89**	0.00	**2168.54**
100-3	13082.38*	17.21	2719.68	114576.21*	926.52	35836.91	**11161.59**	0.00	**2337.35**
100-4	13484.69*	17.04	2551.55	99484.65*	763.50	35129.74	**11521.11**	0.00	**2217.00**
100-5	14484.70*	15.22	2579.12	111338.15*	785.68	17249.10	**12570.86**	0.00	**2279.35**
100-6	13360.41*	18.20	2626.97	99397.46*	779.39	19433.87	**11302.96**	0.00	**2288.82**
100-7	12810.60*	15.20	2553.83	105460.22*	848.32	17271.95	**11120.78**	0.00	**2036.58**
100-8	13809.12*	17.05	2548.32	112170.76*	850.82	18667.88	**11797.25**	0.00	**1792.08**

According to the observation of computational results, we can get three features of EAMLS: (1) For small- and mid-scale problems, the solutions found by EAMLS are comparable to those found by other methods; (2) For large-scale problems, EAMLS significantly outperforms other methods; (3) EAMLS especially performs well on (a) the new complicated model and (b) large-scale problems. So why is EAMLS effective? Through combining MLS with EA and using l3-value to guide the population size to grow gradually, EAMLS performs a full local search while performing a global search, maintains good population diversity, as well as speeds up the convergence.

Our algorithm EAMLS performs well on large-scale problem instances of both $\mathrm{m} = 2$ and $\mathrm{m} = \sum_{j \in J} X_j$ models, and its advantage will become more apparent as the problem scale increases. However, the larger the problem, the greater the number of FEs needed for EAMLS to converge.

6 Conclusion

This paper proposes a new RFLP formulation in which the number of facilities allocated to each customer (i.e., m) is not fixed but varies with decision variables $\mathbf{X}$. This non-fixed allocation setting makes the model more close to scenarios in real life.

A hybrid evolutionary algorithm EAMLS (which can also be viewed as a memetic algorithm) is proposed to solve the model. Combining a memorable local search method and EA, EAMLS performs well on the new complicated model and large-scale problems considered in this paper, and its advantage will become more obvious as the problem scale increases. Besides, a convergence metric l3-value is proposed to analyze the algorithm's convergence speed and exam the evolutionary process.

Finally, we explore the large-scale problems of the two models. Under what conditions is a problem a large-scale problem? It is related to the model and whether the problem can be solved by the exact algorithm efficiently. For the $\mathrm{m} = 2$ model which allocates a fixed number of facilities to each customer as in the existing research, we solve large-scale problem instances (600-node) whose scale is much larger than other literature. For the new complicated $m = \sum_{j \in J} X_j$ model, 100-node instances can be treated as large-scale problems because the exact algorithm or optimization solver cannot solve them effectively. And our algorithm EAMLS has good performance on large-scale problems considered in this paper.

In the future, the model which integrates various factors should be studied, and more complicated FLPs, such as dynamic FLP and FLP under uncertain environments, should be focused. Furthermore, effective meta-heuristic algorithms for large-scale problems should be studied as well.

References

1. Daskin, M.S., Snyder, L.V., Berger, R.T.: Facility location in supply chain design. In: Langevin, A., Riopel, D. (eds.) Logistics Systems: Design and Optimization, pp. 39–65. Springer, Boston (2005). https://doi.org/10.1007/0-387-24977-X_2
2. Farahani, R.Z., Hekmatfar, M.: Facility Location: Concepts, Models, Algorithms and Case Studies. Springer, New York (2009). https://doi.org/10.1007/978-3-7908-2151-2
3. Owen, S.H., Daskin, M.S.: Strategic facility location: a review. Eur. J. Oper. Res. **111**(3), 423–447 (1998)
4. Riopel, D., Langevin, A., Campbell, J.F.: The network of logistics decisions. In: Langevin, A., Riopel, D. (eds.) Logistics Systems: Design and Optimization, pp. 1–38. Springer, Boston (2005). https://doi.org/10.1007/0-387-24977-X_1
5. Li, Q., Zeng, B., Savachkin, A.: Reliable facility location design under disruptions. Comput. Oper. Res. **40**(4), 901–909 (2013)
6. Snyder, L.V., Daskin, M.S.: Reliability models for facility location: the expected failure cost case. Transp. Sci. **39**(3), 400–416 (2005)
7. Peng, P., Snyder, L.V., Lim, A., Liu, Z.: Reliable logistics networks design with facility disruptions. Transp. Res. Part B Methodological **45**(8), 1190–1211 (2011)
8. Jabbarzadeh, A., Jalali Naini, S.G., Davoudpour, H., Azad, N.: Designing a supply chain network under the risk of disruptions. Math. Prob. Eng. **2012**, 23 pages (2012). https://doi.org/10.1155/2012/234324. Article ID 234324
9. Du, B., Zhou, H., Leus, R.: A two-stage robust model for a reliable p-center facility location problem. Appl. Math. Model. **77**, 99–114 (2020)
10. Li, Q., Savachkin, A.: A fast tabu search algorithm for the reliable P-median problem. In: Gao, D., Ruan, N., Xing, W. (eds.) Advances in Global Optimization, vol. 95, pp. 417–424. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-08377-3_41
11. Afify, B., Ray, S., Soeanu, A., Awasthi, A., Debbabi, M., Allouche, M.: Evolutionary learning algorithm for reliable facility location under disruption. Expert Syst. Appl. **115**, 223–244 (2019)