Front. Eng. Manag. 2018, 5(4): 451–465
https://doi.org/10.15302/J-FEM-2018038

RESEARCH ARTICLE

Fred GLOVER, Saïd HANAFI, Oualid GUEMRI, Igor CREVITS

A simple multi-wave algorithm for the uncapacitated facility location problem

Abstract The *multi-wave algorithm* (Glover, 2016) integrates tabu search and strategic oscillation utilizing repeated waves (nested iterations) of constructive search or neighborhood search. We propose a simple multi-wave algorithm for solving the Uncapacitated Facility Location Problem (UFLP) to minimize the combined costs of selecting facilities to be opened and of assigning each customer to an opened facility in order to meet the customers' demands. The objective is to minimize the overall cost including the costs of opening facilities and the costs of allocations. Our experimental tests on a standard set of benchmarks for this widely-studied class of problems show that our algorithm outperforms all previous methods.

Keywords discrete optimization, UFLP, multi-wave optimization, strategic oscillation, tabu search

1 Introduction

Recently, Glover (2016) introduced a new *multi-wave algorithm* (MWA) that embraces both (1) iterated constructive methods and (2) iterated neighborhood search methods (see Algorithm 1). An iterated constructive search starts from a null solution and employs moves that add elements until a complete solution is constructed, and an iterated neighborhood search starts from a given solution and employs improving moves to reach a local optimum.

In this paper, we focus on iterated improvement neighborhood search (which only accepts moves that are improving) as a natural analog of iterated constructive search.

To describe the method in overview, let $N(x)$ = the set of solutions that are neighbors of x, and $M(x)$ = the set of moves that lead to these neighbors; i.e. $M(x) = \{m = m(x) \in N(x)\}$. Let *CL* denote a candidate list of moves extracted from $M(x)$. For problems with large neighborhoods we assume $M(x)$ has been screened to reduce its size using a candidate list approach as described in Glover and Laguna (1997).

In the following, we use the term *boundary solution* to refer to both a completely feasible solution obtained by a constructive algorithm and to a local optimum obtained by a neighborhood search algorithm. We denote a best solution by x^*, which begins as a dummy solution with $f(x^*) = +\infty$, and $f(x)$ denotes the objective function to be minimized, as identified below the progression from a starting solution x° to a boundary solution x will be called a *wave* of the solution process. *MaxPass* and *MaxWave* refer to selected iteration limits. The reference to generating a null starting solution at the start of the "Pass loop" applies to constructive search. We assume *MaxPass* = 1 for this form of search, for reasons elaborated later.

Algorithm 1. Multi-Wave Algorithm (Overview)
1) $x^* = \varnothing$; $f(x^*) = +\infty$;
2) **For** *Pass* = 1 **to** *MaxPass* **do**
3) Generate a starting solution x°;
4) **For** *Wave* = 1 **to** *MaxWave* **do**
5) $x = x^\circ$;
6) **While** *x is not a boundary solution* **do** $x = m(x)$ where m is a move in *CL*;
7) $x^* = argmin\{f(x), f(x^*)\}$;
8) Update relevant *search* parameters for the next *Wave*;
9) EndFor

Received April 4, 2018; accepted July 17, 2018

Fred GLOVER (✉)
Leeds School of Business, University of Colorado, Boulder, CO 80309-0419, USA
E-mail: fred.glover@colorado.edu

Saïd HANAFI, Oualid GUEMRI, Igor CREVITS
LAMIH, CNRS UMR 8201, Université de Valenciennes, France

This work has been funded by ELSAT 2020 project, which is co-financed by the European Union with the European Regional Development Fund, the France state and the Hauts de France Region Council.

10) Update relevant *diversification* parameters for the next *Pass*;

11) EndFor

12) Return x^*

The Uncapacitated Facility Location Problem (UFLP).

Our following development focuses on the design and implementation of a simple multi-wave algorithm for solving the Uncapacitated Facility Location Problem (UFLP).

A variety of facility location problems arise in supply chain management and most of them are combinatorial in nature (Blum and Roli, 2003; Ardjmand et al., 2014; De Corte and Sörensen, 2015; Cerrone et al., 2017; De Armas et al., 2017). The location problems have several industrial applications. For example, determining the number of facilities to be installed and their locations has become an important issue for manufacturing and industrial companies (Melo et al., 2009; Chen et al., 2014). The determination of the best facilities to be implemented, by solving location problems, can significantly reduce the management cost and increase the performance of manufacturing and industrial companies (Cheung et al., 2001; Melo et al., 2009; Amin and Zhang, 2013).

The Uncapacitated Facility Location Problem (UFLP) is one of the most widely studied discrete location problems from this class, and consists of choosing a subset of facilities to be opened from a set of potential candidates, and of assigning each customer to an opened facility in order to meet the customers' demands. The objective is to minimize the overall cost which is the sum of the costs of opening facilities and allocating customers to them. An UFLP can be defined on a bipartite graph $G = (P \cup Q, A)$ with $p + q$ vertices and pq arcs where $p = |P|$ is the number customers and $q = |Q|$ is the number of candidate facilities. The cost of opening a facility is f_i and the cost of serving customer j from facility i is c_{ij}. The UFLP is to find a subset $O \subseteq Q$ of facilities to open to minimize the function

$$f(O) = \sum\nolimits_{i \in O} f_i + \sum\nolimits_{j \in P} \min_{i \in O} c_{i,j}.$$

Since the only combinatorial part of the problem is the selection of open facilities, it is natural to represent a solution by a binary vector $x \in \{0, 1\}^q$ where x_i equals 1 if facility i is open and equals 0 otherwise. We use the notation $Q^1 = Q^1(x) = \{i \in Q: x_i = 1\}$ to represent the facilities that are opened in a solution x. Hence, the optimization problem of interest may be expressed in the form

$$Minimize\, f(x) = \sum_{i \in Q^1(x)} f_i + \sum_{j \in P} \min_{i \in Q^1(x)} c_{i,j}: x \in \{0,1\}^q - \{0\}.$$

Note that all binary vectors in $\{0,1\}^q$ are feasible solutions except for 0.

In recent decades, many methods have been proposed for solving the UFLP, including heuristic methods, exact methods and hybrid methods. Several state-of-the-art surveys on location problems and their applications can be found in Hale and Moberg (2003), Klose and Drexl (2005), ReVelle and Eiselt (2005). Among exact algorithms that have been developed to solve UFLP problems optimally are the branch and bound algorithm of Khumawala (1972), a dual-based approach by Erlenkotter (1978) and its improved version by Körkel (1989), a three-stage algorithm by Galvão and Raggi (1989) and the branch and PEG algorithms of Goldengorin et al. (2003). In addition, Posta et al. (2014) recently introduced a cooperative exact method for solving UFLP integrating tabu search and branch and bound with the aim of finding an optimal solution.

A variety of methods have also been proposed based on Lagrangian relaxation. Within this class, Beasley (1993) developed several Lagrangian heuristics for solving location problems including the UFLP. Later, Beltran-Royo et al. (2012) and Jörnsten and Klose (2016) used Semi-Lagrangian relaxation to solve the UFLP drawing on the notion of Semi-Lagranian relaxation introduced in Beltran et al. (2006) for solving the p-median problem.

Of particular relevance to our current work, a wide range of algorithms have been developed for UFLP using a metaheuristic framework. Basu et al. (2015) present a review on the application of metaheuristics for solving discrete location problems including the UFLP, in which they focus on four methods: scatter search, tabu search, particle swarm optimization and genetic algorithms. Specifically addressing the UFLP, Ghosh (2003) studied several neighborhood search methods including tabu search and a complete local search with memory. Resende and Werneck (2006) proposed a multistart hybrid algorithm (called HYBRID) for UFLP which is based on the algorithm introduced in Resende and Werneck (2004) for solving the p-median problem, and which combined efficient procedures such as path-relinking (used in tabu search and scatter search) together with notions from genetic algorithms. In addition to these approaches, many different variants of tabu search have been successfully applied to UFLP (Al-Sultan and Al-Fawzan, 1999; Michel and Van Hentenryck, 2004; Sun, 2005; Sun, 2006) and Genetic Algorithms (Kratica et al., 2001; Homberger and Gehring, 2008; Tohyama et al., 2011). Other types of approaches include those of Greistorfer and Rego (2006)) who used a simple filter-and-fan method to deal with UFLP and Yigit et al. (2006) who proposed an evolutionary simulated annealing algorithm for solving UFLP. The study of Arostegui et al. (2006) presented a comparison study between simulated annealing, tabu search and genetic algorithms for location problems, in which tabu search obtained particularly promising results. A multi-population particle swarm optimization (MPSO) algorithm was applied by Wang et al. (2008) which included a parallel implementation. Tseng and Wu (2009a) developed a novel hybrid algorithm called multiple trajectory search (MTS) and the same authors also presented a multi-start

drop-add-swap heuristic to deal with UFLP in Tseng and Wu (2009b). Other multi-start approaches for the UFLP include those of Cura (2010) and Ardjmand et al. (2014). The latter multi-start heuristic was dubbed discrete unconscious search and obtained highly promising results. Employing a synthesis of several of the preceding ideas, De Armas et al. 2017 have proposed a method for the stochastic instance of UFLP (SUFLP), using a simheuristic (simulation optimization) approach that first embeds a fast heuristic inside the metaheuristic framework of iterated local search (ILS) for the deterministic version of UFLP and then integrates this procedure with Monte Carlo simulation techniques. Additional recent investigations of UFLP have been carried out by Albareda-Sambola et al. (2017), Atta et al. (2018), Galli et al. (2018), Sahman et al. (2017), Tsuya et al. (2017). There also exist several recent studies of variants and extensions of UFLP, including Akbaripour et al. (2017), Chalupa and Nielsen (2018), Han et al. (2018), Jiang et al. (2018), Pearce and Forbes (2018), Todosijević et al. (2017), An and Svensson (2017). Also quite recently, Ortiz-Astorquiza et al. (2017a, 2017b) respectively provide formulations and approximation algorithms for Multilevel Uncapacitated Facility Location and a comprehensive review on multi-level facility location problems that extend several classical facility location problems.

The remainder of the paper is organized as follows. The next section presents the principles that underlie the Multi-Wave Algorithm and Section 3 exposes the general design of the algorithm. Our comparative experimental study is presented in Section 4 and concluding observations are provided in Section 5.

2 Multi-wave metaheuristic principles

The Multi-Wave Algorithm is based on combining tabu search and strategic oscillation to exploit three chief components: (1) a candidate list (*CL*) for exploring the neighborhood of the current solution; (2) an active move record (*AMR*) to capture the conditional effects of sequential decisions; (3) intensification and diversification strategies founded on the notion of persistent attractiveness. The following subsections present a brief description of each component, but for our present simple version of MWA we only make use of components (1) and (2). As shown in Section 4, even without component (3), our method outperforms all previous methods on the set of benchmark problems standardly used for comparative testing in the literature.

2.1 Candidate list

Let (m) to be the evaluation of the move m, i.e. the change in the objective function that results when performing m on x. We focus attention on a candidate list that contains improving moves with higher evaluations, hence $CL = \{m_1, m_2, ..., m_s\}$ so that $\Delta(m_1) \geqslant ... \geqslant \Delta(m_s) \geqslant max\{\Delta(m): m \in M(x) - CL\}$. Often the size $s = |CL|$ is determined by a threshold T (Glover, 1989, 1995), i.e. $CL = \{m \in M(x): \Delta(m) \geqslant T\}$ where in the present work, the value of T is chosen as

$$T = \Delta_{min} + \lambda(\Delta_{mean} - \Delta_{min}), \quad (1)$$

$$T = \Delta_{mean} + \lambda(\Delta_{max} - \Delta_{mean}), \quad (2)$$

and where Δ_{min}, Δ_{mean} and Δ_{max} refer to the min, mean and max evaluations $\Delta(m)$ over $M(x)$, and $\lambda \in [0, 1]$ is a parameter. Note that a threshold T is also used in GRASP Resende and Ribeiro (2003) which is computed without reference to Δ_{mean} by defining $T = \Delta_{min} + \alpha(\Delta_{max} - \Delta_{min})$, relative to a parameter $\alpha \in [0, 1]$. If the midpoint of the preceding representation, for $\alpha = 0.5$, is taken to be an approximation to Δ_{mean}, then a rough correspondence between this latter candidate list and those based on Δ_{mean} occurs by specifying $\alpha = (\lambda + 1)/2$ (hence $\alpha \in [.5, 1]$) to associate it with (2).

We assume that the evaluations have been scaled and translated, if necessary, so that they are all positive. In the present work, we choose a move m from CL based on the principle of probabilistic tabu search (P-TS) (Glover, 1989) by constructing a sample of size 1 from the uniform distribution with probability $Pr(m)$ given by

$$Pr(m) = \frac{\Delta(m)^{\beta}}{\sum_{m' \in CL} \Delta(m')^{\beta}}. \quad (3)$$

The exponent β can be selected greater than 1 to emphasize the differences between evaluations, or selected less than 1 to de-emphasize these differences. (Another early form of probabilistic tabu search generates probabilities by reference to the exponential function e^r for chosen values of the exponent r (Xu et al., 1997).) The completely random choice that results for $\beta = 0$ effectively corresponds to the approach subsequently proposed by GRASP for its candidate list. More recently, probabilistic variants of GRASP have been introduced in Wang et al. (2013) and Grasas et al. (2017) which adopt a number of ideas from probabilistic tabu search.

In the context of UFLP, let x and x' represent two binary solutions where x' is obtained from x by a 1-*flip move* which changes the value of a single variable from x_k to $x'_k = 1 - x_k$, so we have $x' = x + (1 - 2x_k)e^k$, where e^k is unit vector with all components zero except the k^{th} component. To evaluate a 1-flip move efficiently, for each customer j, we maintain the closest facility and the second closest facility, i.e.,

$$i_j^1 = i_j^1(x) = argmin\{c_{i,j} : i \in Q^1(x)\},$$

$$i_j^2 = i_j^2(x) = argmin\{c_{i,j} : i \in Q^1(x) - \{i_j^1(x)\}\}.$$

Then the objective function can be rewritten as $f(x)=\sum_{i\in Q^1}f_i+\sum_{j\in P}c_{i_j^1 j}$. Then the objective function change produced by flipping x_k is given by $\Delta_k=f(x)-f(x')$. The objective function value after closing facility $k\in Q^1$ is given by

$$f(x')=\sum_{i\in Q^1}f_i-f_k+\sum_{j\in P-P_k^=}c_{i_j^1 j}+\sum_{j\in P_k^=}c_{i_j^2 j},$$

where $P_k^=$ is the set of customers assigned to k, i.e., $P_k^==\{j\in P: i_j^1=k\}$. The change Δ_k^- of closing facility k is given by

$$\Delta_k^-=f_k-\sum_{j\in P_k}=(c_{i_j^2 j}-c_{i_j^1 j}). \tag{4}$$

The objective function value after opening facility $k\in Q^0=Q^0(x)=\{i\in Q: x_i=0\}$ is given by

$$f(x')=\sum_{i\in Q^1}f_i+f_k+\sum_{j\in P}min(c_{i_j^2 j}-c_{kj}).$$

The change Δ_k^+ of opening facility k is given by

$$\begin{aligned}\Delta_k^+&=-f_k+\sum_{j\in P}max\{0,c_{i_j^1 j}-c_{kj}\}\\&=-f_k+\sum_{j\in P_k^-}c_{i_j^1 j}-c_{kj},\end{aligned} \tag{5}$$

where $P_k^-=\{j: c_{kj}<c_{i_j^1 j}\}$. Hence, the change Δ_k of flipping facility k can be expressed as

$$\Delta_k=\begin{cases}\Delta_k^- & if \quad x_k=1\\ \Delta_k^+ & if \quad x_k=0\end{cases}.$$

Michel and Van Hentenryck (2004) and Sun (2006) provide an efficient procedure in $O(p\ log\ q)$ time to update the evaluation Δ_k for facility and warehouse location problems after each flip move.

Now we consider a swap move. Let x and x' represent two binary solutions where x' is obtained from x by swapping the value of two variables x_k and x_h with $h\in Q^1(x)$ and $h\in Q^0(x)$, so we have $Q^1(x')=Q^1(x)-k+h$. Then the objective function $f(x')=\sum_{i\in Q^1(x)}f_i-f_k+f_h+\sum_{j\in P}\min_{i\in Q^1(x)-k+h}c_{i,j}$. Note that

$$\min_{i\in Q^1(x)-k+h}c_{i,j}=\min\{c_{i,j}: i\in Q^1(x)-k+h\}$$

$$=\begin{cases}c_{i_j^1 j} if\, c_{i_j^1 j}<\min\{c_{kj},c_{hj}\}\\ c_{i_j^2 j} if\, i_j^1=k \quad and \quad c_{i_j^2 j}<c_{hj}\\ c_{hj} \qquad otherwise\end{cases}.$$

Let $P^1=\{J\in P: c_{i_j^1 j}<\min\{c_{kj},c_{hj}\}\}$, $P^2=\{J\in P: c_{i_j^1 j}<c_{kj}\ and\ c_{i_j^2}<c_{hj}\}$, and $P^3=P-P^1-P^2$ be a partition of P. Then the objective function change produced by swapping x_h and x_k is given by $\Delta_{hk}=f(x)-f(x')$. More precisely, we have

$$\Delta_{hk}=f_k-f_h+\sum_{j\in P}\left(\min_{i\in Q^1(x)}c_{ij}-\min_{i\in Q^1(x)-k+h}c_{ij}\right) \tag{6}$$

This last evaluation can be simplified as follows

$$\begin{aligned}\Delta_{hk}=&f_k-f_h+\sum_{j\in P^2}(c_{i_j^1 j}-c_{i_j^2 j})\\&+\sum_{j\in P^3}(c_{i_j^1 j}-c_{hj}).\end{aligned} \tag{7}$$

According to Resende and Werneck (2007), the objective function change can be rewritten in the following form

$$\Delta_{hk}=\Delta_k^+-\Delta_h^-+\Delta_{hk}^{\mp}, \tag{8}$$

where

$$\Delta_{hk}^{\mp}=\sum_{j\in P^2}c_{i_j^2 j}-\max(c_{i_j^1 j}-c_{kj}),$$

and Δ_k^+ is called *gain*(k) which corresponds to the decrease in solution value due to the insertion of facility k (with no associated removal). Similarly, Δ_h^- is called *loss*(h) corresponding to the increase in solution value due to the removal of facility h (with no associated insertion), while $\Delta_{hk}^{\mp}$ is called *extra*(k, h) corresponding to a positive correction term that accounts for the fact that the insertion of facility k and the removal of facility h may not be independent (a customer previously assigned to facility h may be reassigned to facility k). Note that i) $\Delta_{hk}^{\mp}$ is always positive or equal to 0; ii) If Δ_h^- is negative, then there is an improvement when closing h; iii) If Δ_k^+ is positive, then there is an improvement when opening k.

In our implementation, the candidate list contains *Drop*, *Add* and *Swap* moves. To keep the size of the candidate list moderate, only the *Swap*(k, h) moves that satisfy the condition $\Delta_{hk}>0$ and $\Delta_{hk}^{\mp}>\Delta_h^-$ are candidates to belong to $CL(x)$. This condition assures by Eq. (8) that we only consider *Swap* moves that dominate *Add* or *Drop* moves.

2.2 Conditional effect of sequential decisions and active record move

Starting from an initial solution, the algorithm repeatedly performs improving moves until a boundary solution is reached. In this sequence of decisions, the moves applied in early steps of the MWA affect the evaluation and the choice of the decision in the current step because the information underlying the current decision depends on these earlier moves. (This is the so-called *conditional effect of sequential decisions*.) Furthermore, the evaluation of current decisions is able to take advantage of more refined

information than is available to early decisions, conditional upon decisions already made, due to the fact that the problem is now reduced, with the implication that later decisions are likely to be better in a conditional sense than earlier decisions. Consequently, as observed in Glover (2000, 2016), it becomes advantageous to re-examine earlier decisions (which are likely to be bad) at intermediate stages during the search process in order to amend them and thus create paths to higher quality boundary solutions.

In its general design, the Multi-Wave Algorithm exploits this notion of the conditional effect of sequential decisions using an *Active Move Record* (*AMR*) whose form is described below. Specifically, the algorithm starts with a *vertical phase* by performing constructive or improving moves, and then after a given number of iterations the moves (decisions) executed earlier are re-examined a *horizontal phase* to see if additional improvement results by replacing them with other moves that now are evaluated to be better. An alternation between vertical and horizontal phases which are triggered by parameters of the search continues until a boundary solution is reached, whereon updates are made and the process begins again.

We refer to both constructive and improving moves as *forward moves*, and refer to destructive moves and moves that complement (or "cancel") previous improving moves as *reverse moves*. For a constructive algorithm, the operation of dropping and reversing a move m is often very simple, consisting only of removing the move m from *AMR*. We will understand the reference to a reverse move $\overline{m}$ in this case to involve just the dropping operation. We also refer both to the (arbitrary) beginning solution that launches an improving phase of neighborhood search and to the null construction that launches a constructive phase, as the *initial solution*.

More precisely, the *Active Move Record* identifies the forward moves selected so far that have not been cancelled by reverse moves. It is convenient to represent this record by $AMR = (m_1, m_2, \ldots, m_r)$, where $r = |AMR|$, and where the indexing sequence indicates that move m_i was executed before move m_{i+1} for $i \in [1, r-1]$. It should be noted that $M(x)$ and *CL* can include moves that reverse moves in *AMR*.

At any stage, the effect of applying the moves in *AMR* to the initial solution yields a current solution x and in the case of a constructive algorithm, we assume that the objective function value $f(x)$ can be evaluated for a partial solution as well as a complete solution. We first sketch the general form of the Multi-Wave algorithm and then describe the ways that *AMR* and *CL* are used within it.

3 General design of the multi-wave algorithm

Each wave is divided into alternating vertical and horizontal components, where vertical steps consist of forward moves that enlarge the *AMR* and horizontal steps consist of combined forward and reverse moves designed to leave the cardinality of *AMR* unchanged. Here we handle the horizontal phase by the simpler approach of dropping moves from *AMR* by reversing them, and then adding an equal number of forward moves to *AMR*. The key steps for carrying this out involve: (1) deciding when a horizontal phase is to be launched, (2) selecting the number of moves to be executed during this phase, and (3) identifying the particular moves drawn from *AMR* to drop and the associated forward moves from *CL* that make up the phase.

We index the successive intervention points by $k = 1, 2, \ldots, k^*$. The parameter v_k denotes the number of vertical steps (forward moves) executed before launching intervention k, which therefore occurs when $|AMR| = v_k$. Finally, h_k denotes the number of horizontal steps executed during this intervention and d_k denotes the number of moves dropped on each of these steps. The values of these parameters are interrelated and we give simple rules to determine them. The condition for reaching an intervention point and executing a horizontal phase may be expressed as follows.

If $|AMR| = v_k$, perform h_k horizontal steps each consisting of dropping (reversing) d_k moves from the start of *AMR* and sequentially adding d_k new moves to *AMR* from *CL* (where *CL* has access to the moves dropped from *AMR*). This sequential selection implies that the move evaluations may change from one step to the next. If no forward move exists to replace a given reverse move (as where no improving moves currently exist), a boundary solution is reached and the wave terminates.

In short, all steps of adding moves to *AMR* during the horizontal phase have exactly the same form as the steps of choosing and executing forward moves during the vertical phase.

Algorithm 2 describes in detail the main steps used in the proposed Multi-Wave Algorithm.

Figure 1 shows the evolution of the *AMR* in the Vertical Phase when $r = |AMR| < v_k$ which consists of adding moves to *AMR* until it reaches the size $r = v_k$. Figure 2 illustrates the steps of the Horizontal Phase that performs d_k Drop / Add moves h_k times. Note that the Concluding Horizontal Phase can be also illustrated by Fig. 2 where the process iterates h_0 times by dropping d_0 moves from *AMR* and adding moves until reaching a boundary solution.

Algorithm 2. Multi-Wave Algorithm

1) $x = \varnothing$; $x^* = x$; $f(x^*) = \infty$;

2) **For** *Pass* = 1 **to** *MaxPas* **do**

3) Generate a starting solution x° (*null, randomly or by post-analysis diversification*);

4) **For** *Wave* = 1 **to** *MaxWave* **do**

5) $x = x^\circ$; $AMR = \varnothing$; $k = 1$;

6) **While** *x is not a boundary solution* **do**

7) **If** $|AMR| = v_k$ **then // Horizontal Phase**

8) **For** $h = 1$ **to** h_k **do**
9) Reverse the first d_k moves and remove them from *AMR*
10) Perform d_k forward moves and add them to *AMR*
11) EndFor
12) $k \leftarrow k + 1$;
13) Else // Vertical Phase
14) Perform one forward move and add it to *AMR*
15) EndIf
16) EndWhile
17) $x^* = argmin\{f(x), f(x^*)\}$;
18) **For** $h = 1$ **to** h_0 **do // Concluding Horizontal Phase**
19) Reverse the first d_0 moves and remove them from *AMR*
20) **While** *x is not a boundary solution* **do**
21) Perform a forward move and add it to *AMR*
22) EndWhile
23) $x^* = argmin\{f(x), f(x^*)\}$;
24) **EndFor**
25) Update v_k, h_k and d_k for all k;
26) EndFor
27) EndFor
28) Return x^*

A variety of rules are possible for determining the parameters of the Multi-Wave Algorithm, and particularly those of the horizontal phase. We focus here on rules that are easy to execute provided in subsequent sections.

3.1 Determining the intervention parameter values v_k and k^*

The v_k values which identify the values of $|AMR|$ at which interventions occur can be conveniently established by using an estimate r_e of the final $|AMR|$ value upon reaching a boundary solution. Such an estimate can be determined by executing a preliminary wave without any interventions (as insured by making v_1 large) and setting $r_e = |AMR|$ when the wave terminates. This r_e estimate can be updated after each subsequent wave to equal the final $|AMR|$ value obtained during this wave or the average of the final $|AMR|$ values obtained so far.

Given r_e, a desired spacing between successive interventions can be selected based on either:

(1) The separation δ_k between successive values v_k at which successive interventions occur, which yields $v_k = v_{k-1} + \delta_k$ for $k = 0$ to k^*, where by convention $v_0 = 0$. The value δ_k should satisfy $\delta_k \geqslant d_k + 1$. In the simplest case, we select a constant value $\delta = \delta_k \geqslant 2$, which therefore determines $k^* = \lfloor r_e / \delta \rfloor$. This approach is useful for

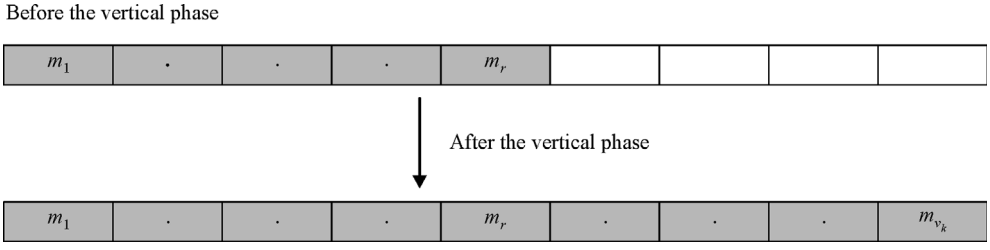

Fig. 1 Vertical phase $r = |AMR| < v_k$

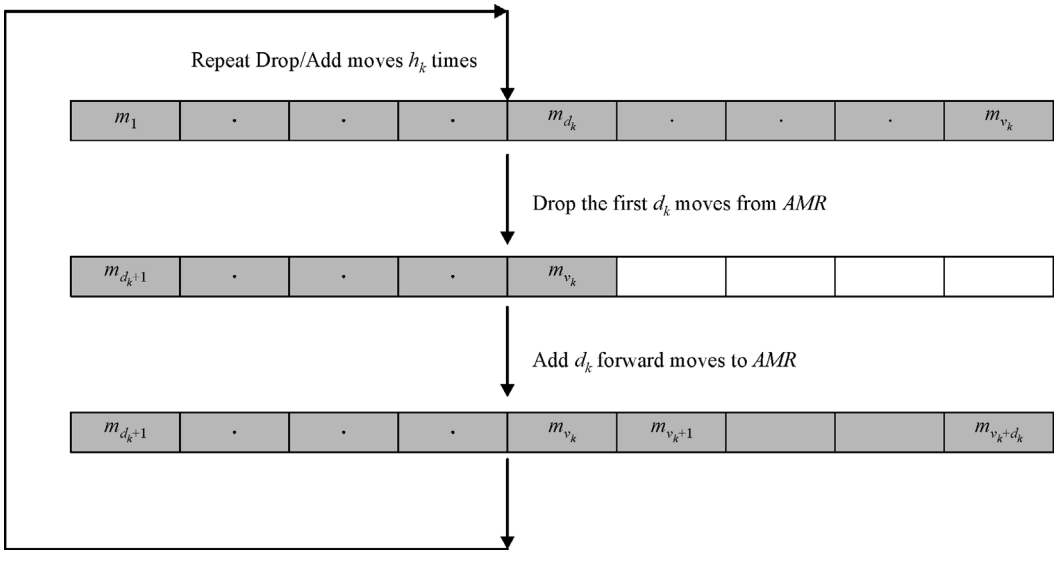

Fig. 2 Horizontal phase $r = |AMR| < v_k$

densely populating the values v_k over the estimate r_e, hence is suited to choosing k^* relatively large (in comparison to r_e).

(2) The chosen total number of interventions k^*. In the simplest case, we divide the interval $[1, r_e]$ into equal parts and we consider values of k^* ranging from 1 to a selected upper limit (5 for small value), then for any value of k^*, let $v_k = [k\, r_e/(k^* + 1)]$ or $v_k = [(k + 1)r_e/(k^* + 2)]$ for each $k =$ 1 to k^*. This approach is useful for sparsely populating the values v_k over the estimate r_e, and it is suited to choosing k^* relatively small.

Note that with the first technique (1), the value of δ determines the total number of interventions k^*. We have tested the simplest version where $\delta_k = \delta$ is a constant value, but this simplest rule worked well only for some instances of UFLP. Hence we implement a new rule where δ_k increases after each η waves $\eta \in \{1, \ldots, 4\}$ starting with small value of $\delta_1 \in \{2, 3\}$ (i.e., $\delta_{k+h} = \delta_k + \tau$ with $\tau \in \{1, \ldots, 6\}$). To implement the second rule (2), based on the estimation of r_e, we keep a record at each *pass* of the average number of moves performed up to the current wave. Accompanying this, we increase the number of interventions k^* by 1 at each wave ($k^* = k^* + 1$), starting with a small value (in our implementation, $k^* = 5$). Hence, we have $v_k = v_{k-1} + \delta$ where $\delta = \frac{\tilde{r}_e}{k^*}$.

3.2 Determining the number of drop moves d_k

The value d_k identifying the number of moves to be dropped during the kth intervention point should be chosen, as previously noted, so that $d_k < \delta_k = v_k - v_{k-1}$. Again, an equal spacing option gives a possible starting point, giving each d_k a value $d_k = d > 1$ for all k. Alternatively, d_k may be selected randomly at each intervention point to lie in the interval $[1, d]$. Another option is to let d_k be larger on the first step of the horizontal phase and then decrease it to a smaller value on subsequent steps. In the latter case, two alternatives of interest are to decrease d_k immediately to 1 and to decrease it by a single unit at each step until it reaches 1. In our implementation for UFLP we determined the number of drop moves by setting $d_k = 2$ for all k, adopting the equal spacing option because of its simplicity and effectiveness.

3.3 Determining the number of horizontal steps h_k

We determine the number of horizontal steps h_k as a function of the number of dropped moves d_k. To assure that every move in *AMR* is dropped on one of these steps performed on the kth intervention, we select $h_k = \lceil |AMR|/ d_k \rceil$ in our current implementation, where $|AMR|$ refers here to the size of the active move record when the kth intervention is launched. In particular, each time a move or a block of moves is selected from *AMR*: (a) the moves dropped from *AMR* are made available to become members of *CL*; (b) the rule of probabilistic tabu search is employed for successively choosing moves from *CL* to be executed; and (c) $M(x)$ and hence *CL* are updated after each of these moves. It is reasonable to expect that the horizontal interventions will contribute to an intensification/diversification tradeoff in a way that will enable the P-TS choice rules to focus primarily on intensification, as by using relatively large λ and β values. If λ is chosen sufficiently large, e.g., in the interval [.98, 1.0], then the value of β will be largely irrelevant. In our implementation, at each *pass*, the value of β starts from 1.5 and ends at 0.5 and at each *wave* decreases by stepwize 1/*MaxWave*, while the value of λ starts from 0.8 and ends at 0 and at each *wave* decreases by stepwize 0.8/*MaxWave*. Starting from large λ and β values allows intensification of the search while slowly reducing those values at each pass permits an intensification/diversification tradeoff.

3.4 Concluding horizontal phase

The goal behind the concluding horizontal phase is to iron out imperfections produced after the last intervention. This procedure is applied to the *AMR* produced by the current wave. The difference between the concluding horizontal phase and the horizontal phase performed during the wave is: after dropping d_0 moves from *AMR* and reversing them, the concluding horizontal phase performs forward moves until a boundary solution is reached, rather than stopping after executing d_0 forward moves, as done in the horizontal phase of the wave. From this fact, the initial values of h_0 and d_0 are chosen to depend on the value $|AMR|$. Choosing a large value of d_0 can make a significant change in the solution x and leads us to start a new search from scratch, as in a simple multi-start search, and this will be done h_0 times. On the other hand, choosing a small value of d_0 allows us to iron out early imperfections, but if $|AMR|$ is large then h_0 will receive a large value, and this will negatively affect the search procedure and its CPU-time. To achieve the goal of the concluding horizontal phase without inhibiting the effectiveness of the search process, we fix h_0 and d_0 by setting $d_0 = 2$ and $h_0 = min(\lceil |AMR|/d_0 \rceil, \overline{h})$, where $\overline{h}$ is the maximum number of iterations we imposed on h_0. Consequently, if $\lceil |AMR|/d_0 \rceil > \overline{h}$ the procedure will not examine all the moves in *AMR* and in such a case only early moves in *AMR* are reversed. We are motivated to use this technique based on the notion of the conditional effect of sequential decisions.

4 Computational results

Our simple version of the multi-wave algorithm was coded in JAVA and all experiments were conducted on a computer with Intel Xeon E3-1505M CPU @ 2.80 GHz and 16 GB RAM. To test the performance of our MWA, we

use four of the most widely referenced and challenging benchmark data sets of UFLP from the literature including: The benchmark of Ghosh (2003) (***GHOSH***), the data set of Kratica et al. (2001) (***M****), the large-scale data set used in Barahona and Chudak (1999) (***MED***) and the test problems from the OR-Library of Beasley (1990) (***ORLIB***). These data sets can be uploaded from http://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/. In addition, the MWA is compared against nine of the most efficient methods developed for the UFLP. These state-of-the-art methods are:

1) LAG: The Lagrange-method from Sun (2006) which is implemented in Sun (2005).

2) CLM: The Complete Local search with Memory of Ghosh (2003).

3) GTS: The Tabu Search of Ghosh (2003).

4) HYB: The Hybrid multistart method of Resende and Werneck (2006).

5) TS: The Tabu Search of Sun (2006).

6) MDAS: The Multi-start Drop Add Swap heuristic of Tseng and Wu (2009b).

7) MTS: Multiple Trajectory Search of Tseng and Wu (2009a).

8) US: The discrete Unconscious Search of Ardjmand et al. (2014).

9) ILS: The Iterated Local Search method of De Armas et al. (2017).

Our experiments are divided into two parts. In the first part, we present a comparison between the proposed MWA and two multi-start algorithms we have implemented with the aim to identify performance differences between the MWA and classical multi-start approaches. In the second part of our experimentation, the MWA results are compared to the outcomes from the 9 state-of-the-art algorithms identified above. In all experiments, we report CPU-times for the MWA in two columns: Column ***All*** which refers to the full CPU-time of the execution and column ***Find*** which refers to the CPU-time needed to find the best solution in the execution. In what follows, first we give a brief description of the benchmark data sets used, then we present the two parts of the experiments.

4.1 Benchmarks for UFLP

We use the following four benchmark data sets to evaluate the MWA:

• ***GHOSH***: In this data set from Ghosh (2003), the number of customers p and the number of facilities q are set to be $(p, q) \in \{(250, 250), (500, 500), (750, 750)\}$ with 30 problem instances of each size. These instances are divided into three groups A, B and C in which the facility setup costs f_i, are chosen uniformly from intervals [100, 200], [1000, 2000] and [10000, 20000] respectively for low, medium and high fixed cost values. In all instances, the assignment cost c_{ij} is chosen at random from [1000, 2000]. In addition, each set of test problems of the same size and from the same group are further divided into two categories, where in 5 instances c_{ij} is symmetric (i.e., $c_{ij} = c_{ji}$) and in the other 5 instances c_{ij} is asymmetric.

• ***MED***: This benchmark contains 18 large-scale problems, originally introduced for the p-median problem by Ahn et al. (1988) and used later by Barahona and Chudak (1999) in the context of UFLP. We name each instance as q-y where the number of facilities $q \in \{500, 1000, 1500, 2000, 2500, 3000\}$ (each facility is a candidate for opening and represents a customer at the same time) and the parameter $y \in \{10, 100, 1000\}$. Each combination of q and y is included to yield a total of 18 instances. The parameter y is used to calculate the setup cost $f_i = \frac{\sqrt{q}}{y}$, which is the same for all facilities.

• ***M****: The third benchmark is the data set of Kratica et al. (2001). According to the authors, these problems are generated to closely approximate real-life cases. In these problems the number of customers is equal to the number of facilities and varies from 100 to 2000 (i.e., $p = q \in \{100, 200, 300, 500, 1000, 2000\}$).

• ***ORLIB***: The fourth test problem set is taken from the OR-Library of Beasley (1990). These instances were initially proposed for the capacitated facility location problem and then used for the uncapacitated facility location problem case by ignoring the capacity of facilities. This data set contains 15 instances, in which the numbers of facilities and customers vary from $((q, p) = (16, 50)$ to $(q, p) = (100, 1000))$.

4.2 Comparison between basic multi-start and the MWA

In this subsection, we present a comparison between two basic multi-start methods and the MWA. The aim behind this comparison is to identify the contribution provided by techniques used in our simple MWA that differentiate it from multi-start methods: (1) the use of the *AMR* and the division of a wave into horizontal and vertical phases in order to exploit the conditional effects of the sequential decisions, and (2) the manner in which we exploit the candidate list in order to select the most appropriate moves to be applied at each step of the algorithm. To achieve this aim, we compare the results returned by the MWA to those obtained by the two basic multi-start methods using the same CPU times.

The basic multi-start algorithms we implemented may be viewed as simple instances of the GRASP approach and consist of two sequential phases: (i) a Greedy Randomized Procedure (GRP) in which a new starting solution is generated, and (ii) a local search which improves the solution returned from the first phase. These steps are performed until the time limit stopping criterion is met. In traditional GRASP methods the GRP procedures are used to construct complete and/or feasible solutions but in our case, a UFLP-solution with only one facility is a complete and feasible solution. In fact, in our multi-start methods, the GRP procedure is used to find starting solutions with

good tradeoff between the assignment cost and facilities opening cost. To do this, the GRP starts from a solution with one facility (i.e., with low opening cost even that generated a large transportation cost). To establish a good tradeoff between the opening cost and the assignment cost (that minimizes the overall cost) we search for the best number of facilities to be opened by starting with one facility and continue to add facilities using the greedy randomized technique until no improving add moves are available. Thus, the method performs improving add moves which both improve the solution quality and converge to a good tradeoff between the assignment and opening costs. After that, the method performs a local search procedure in which the *Add*, *Drop* and *Swap* moves are considered.

In this study, the two versions of the multi-start approach are examined: MultiS-FI, in which the local search adopts the first improvement choice strategy and MultiS-BI, in which the local search adopts the best improvement choice strategy. Table 1 shows the results of the three algorithms MultiS-FI, MultiS-BI and MWA: the first column presents the instances, then in columns from two to seven, for the two multi-start methods, we present the average solution cost, the best solution cost and the average number of iterations out of 10 runs. In columns eight and nine we show the MWA results, and finally, in column, ten we present the CPU times of the methods. These results are obtained as follows: First, we run the MWA algorithm on these instances by setting the *MaxPas* = 10 and *MaxWave* = 10 (so 100 waves are performed). Then we run the two multi-start methods using the same amount of CPU-times consumed by the MWA.

As shown in Table 1, the MWA algorithm clearly proves more effective than the two multi-start methods. The MWA obtains higher quality best solutions for all MED instances and in most instances, the average solution cost obtained by the MWA is better than the best solution cost obtained by the multi-start methods. In addition, we observe that the average number of iterations for MultiS-FI equals 1170 and for MultiS-BI equals 928. So, on average, around 1000 solutions are investigated by these methods in one execution, whereas the 100 solutions generated by executing only 100 waves of the MWA turn out to include much better outcomes. Hence, we conclude that the new techniques used in our simple MWA method allow it to uncover better regions in the search space where high-quality solutions are available.

4.3 Comparison with state-of-the-art algorithms

We now present the results of the MWA on each benchmark and compare these results to those obtained by the most effective methods from the literature.

4.3.1 Results on GHOSH test problems

As previously noted, this benchmark contains hard test problems. We compare the results of the MWA on these

Table 1 MWA results versus multi-start results

	MultiS-FI			MultiS-BI			MWA		
Instance	**Avg**	**best**	**Nbr_it**	**Avg**	**best**	**Nbr_it**	**Avg**	**Best**	**CPU**
500-10	798577.0	798577*	521.2	798577.0	798577*	514.8	798577.0*	798577*	8.22
500-100	327068.8	327001	1191.7	327017.9	326901	917.2	326798	326790*	20.49
500-1000	99175.8	99170	2767.3	99172.9	99169*	2632.9	99170.9	99169*	47.16
1000-10	1434250.7	1434154*	442.6	1434301.7	1434154*	430.4	1434154*	1434154*	38.75
1000-100	608008.3	607903	1407.0	608028.8	607903	1007.6	607883.9	607878*	134.62
1000-1000	220703	220668	2698.2	220725.3	220680	2322.2	220585.2	220560*	273.47
1500-10	2000839.6	2000801*	452.6	2000839.5	2000801*	433.8	2000828.8	2000801*	111.23
1500-100	867063.7	866829	1021.2	867451.8	867163	764.3	866490.1	866454*	314.24
1500-1000	335579.5	335491	1893.3	335571.8	335487	1297.9	335059.1	335002	702.70
2000-10	2558118.0	2558118*	533.1	2558126.2	2558118*	492.8	2558119.4	2558118*	277.50
2000-100	1123893.3	1123172	1121.6	1124056.8	1123188	765.4	1122854.5	1122805	660.59
2000-1000	438498.7	438295	1216.0	438544.6	438304	503.3	437813.6	437693	1201.25
2500-10	3100172.0	3099907*	550.0	3100439.2	3099907*	570.4	3100155.9	3099907*	531.12
2500-100	1348372.0	1347790	996.7	1348954.5	1348594	695.4	1347566.8	1347516*	1036.96
2500-1000	535364.9	535132	1581.9	535271.4	535091	1186.9	534593.6	534506	2737.16
3000-10	3570904.8	3570766	534.2	3570988.5	3570766*	562.6	3570766*	3570766*	876.77
3000-100	1604408.9	1604011	1006.8	1605408.1	1604748	709.7	1602512.2	1602345	1711.03
3000-1000	644839.1	644718	1133.1	644746.6	644574	909.4	643885.3	643797	3389.68
AVG			1170.47			928.72			

Table 2 MWA results versus literature results on GHOSH benchmark instances

Instance	TS	MDAS	GTS	CLM	HYB	MTS	US	MWA
250-sym-A	257805.0	**257804.0**	257832.6	257895.2	257806.2	**257804.0**	**257804.0**	**257804.0**
250-sym-B	**276035.2**	**276035.2**	276185.2	276352.2	**276035.2**	**276035.2**	**276035.2**	**276035.2**
250-sym-C	**333671.6**	**333671.6**	333820.0	**333671.6**	**333671.6**	**333671.6**	**333671.6**	**333671.6**
250-asym-A	**257917.8**	**257917.8**	257978.4	258032.6	257923.4	**257917.8**	**257917.8**	**257917.8**
250-asym-B	**276053.2**	**276053.2**	276467.2	276184.2	**276053.2**	**276053.2**	**276053.2**	**276053.2**
250-asym-C	**332897.2**	**333897.2**	333237.6	333058.4	**332897.2**	**333897.2**	**332897.2**	**332897.2**
500-sym-A	**511180.4**	511181.2	511383.6	511487.2	511196.4	511188.8	**511180.4**	**511180.4**
500-sym-B	**537912.0**	**537912.0**	538480.4	538685.8	**537912.0**	**537912.0**	**537912.0**	**537912.0**
500-sym-C	**621059.2**	**621059.2**	621107.2	621172.8	**621059.2**	**621059.2**	**621059.2**	**621059.2**
500-asym-A	511140.0	**511136.4**	511251.6	511393.4	511145.0	511137.8	511137.4	**511136.4**
500-asym-B	**537847.6**	**537847.6**	538144.0	538421.0	537863.4	**537847.6**	**537847.6**	**537847.6**
500-asym-C	**621463.8**	**621463.8**	621811.8	621990.8	**621463.8**	**621463.8**	**621463.8**	**621463.8**
750-sym-A	763693.4	**763684.8**	763830.8	763978.0	763706.6	763708.8	**763684.8**	**763684.8**
750-sym-B	**796571.8**	**796571.8**	796919.0	797173.4	796632.2	**796571.8**	796576.8	**796571.8**
750-sym-C	**900158.6**	**900158.6**	901158.4	900785.2	900272.0	**900158.6**	**900158.6**	**900158.6**
750-asym-A	763717.0	763716.4	763836.6	764019.4	763731.2	763735.8	763712.4	***763711.6****
750-asym-B	**796374.4**	**796374.4**	796859.0	796754.2	796396.8	**796374.4**	**796374.4**	**796374.4**
750-asym-C	**900193.2**	**900193.2**	900514.2	900349.8	**900193.2**	**900193.2**	**900193.2**	**900193.2**
#BKS	**14/18**	**16/18**	**0/18**	**1/18**	**8/18**	**14/18**	**15/18**	**18/18**

problems to the results of: TS, CLM, GTS, HYB, MTS, US and MDAS. Table 2 presents the solution costs (objective function values) obtained by each method and Table 3 presents the CPU-times of each method. The results for TS, GTS, HYB, MTS, CLM and US are taken from Sun (2006) and Ardjmand et al. (2014). The results of MDAS are not reported in Ardjmand et al. (2014) and we take them directly from the original paper of Tseng and Wu (2009b).

As shown in Table 2, the MWA was able to reach all best-known solutions (which were found by at least one of the most effective methods in the literature) and to obtain a new best known solution for the instance 750-*asym-A*. By contrast, none of the previously proposed methods was able to find all the best solutions previously known, disregarding the new best solution we have found. Table 3 additionally shows that our simple MWA has a competitive computing time compared to the other methods. The MWA performs particularly well on the type *C* problems where it is able to reach the best solution of two large-scale instances in less than 2 s (750-*sym-C* and 750-*asym-C*) and obtain best solutions for two of the medium size instances in less than 1 s (500-*sym-C* and 500-a*sym-C*). In addition, on average, our algorithm needs only half the CPU-time required by the fastest method to obtain its best results (comparing the averages of these CPU-times).

4.3.2 Results on MED test problems

The second comparison is made on the MED test problems. In Table 4, we compare the four methods that have results on this data set: ILS, MDAS, HYB and MWA. In the first and the second columns we present the instance name and the optimal solution found by the solver Gurobi, then for each method, we report the average solutions cost and the average CPU-times out of 30 runs for ILS and HYB, 50 runs for MDAS and 10 runs for the MWA. The results of HYB and ILS are taken from Resende and Werneck (2006) and De Armas et al. (2017) and the results of MDAS are taken from Tseng and Wu (2009b). Note that the results for MDAS are not reported in De Armas et al. (2017).

Comparing MWA to ILS and MDAS, the MWA obtains better solutions for all problem instances with the exception of one problem in which MDAS and MWA obtain the same optimal solutions over all executions (500-10). Compared to ILS, the MWA method finds better solutions to all problem instances and requires smaller average CPU-time (778,23 versus 931,41). Although the average CPU-time for MDAS is slightly smaller than the average CPU-time for MWA, we observe that in all instances where $q \in \{500, 1000, 1500, 2000, 2500, 3000\}$-$y = 10$ the CPU-times of MWA are better than those of MDAS. Interestingly, for the indicated problems from the MED test set, in contrast to the problem from the GHOSH test set, the HYB method is much closer to matching the MWA performance overall – obtaining better results than MWA in 7 instances, while the MWA method obtains better results than HYB in 10 instances (and in one

Table 3 MWA CPU-times versus literature CPU-times on GHOSH benchmark instances

Instance	TS	MDAS	CLM	HYB	MTS	US	MWA	
							All	***Find***
250-sym-A	2.828	7.36	86.842	4.328	5.59	0.843	1.94	0.18
250-sym-B	5.628	5.88	34.634	7.774	7.73	0.618	0.75	0.1
250-sym-C	9.878	5.73	69.458	8.702	8.49	0.63	0.435	0.05
250-asym-A	2.618	7.32	86.506	4.636	5.62	0.993	1.84	0.78
250-asym-B	5.79	5.92	33.688	8.082	7.33	0.6	0.75	0.08
250-asym-C	9.196	5.67	89.99	7.776	8.51	0.586	0.39	0.04
500-sym-A	15.616	35.87	946.028	27.644	27.4	8.201	14.72	7.98
500-sym-B	31.432	26.72	294.656	34.196	32.7	4.619	4.29	0.95
500-sym-C	71.106	24.18	437.462	40.376	36.72	3.46	2.27	0.47
500-asym-A	13.76	35.97	921.208	20.232	27.43	7.736	11.81	7.28
500-asym-B	34.748	26.73	311.344	31.3	32.91	4.751	3.95	0.53
500-asym-C	72.064	26.60	388.21	47.79	36.61	3.601	2.31	0.45
750-sym-A	39.812	93.68	3650.662	49.214	67.63	25.903	34.41	20.31
750-sym-B	93.352	68.83	1583.17	92.886	77.67	14.601	11.37	4.1
750-sym-C	229.914	56.31	1194.534	113.64	88.17	10.888	7.117	1.31
750-asym-A	39.65	92.60	3658.588	59.13	67.91	26.914	40.58	20.49
750-asym-B	95.43	64.71	1606.778	73.322	78.06	14.542	11.17	5.03
750-asym-C	236.902	56.95	1325.812	112.994	87.17	10.827	7.12	1.76
AVG CPU	**56.10**	**35.95**	**928.87**	**41.33**	**39.09**	**7.80**	**8.73**	**3.99**

Table 4 MWA results versus literature results on MED test problems

Instance	Gurobi	ILS		MDAS		HYB		MWA		
	OPT	AVG	CPU	AVG	CPU	AVG	CPU	AVG	CPU (***Find***)	CPU(***All***)
500-10	798577	799077.1	46.89	798577.0*	28.0	798577*	33.2	798577.0*	1.488	8.22
500-100	326790	327007.6	18.54	326922.375	25.6	326805.4	32.9	326798	12.402	20.49
500-1000	99169	99172.6	14.57	99196.0	22.4	99169*	23.6	99170.9	18.13	47.16
1000-10	1434154	1435918.6	54.94	1434171.0	130.7	1434185.4	173.9	1434154*	15.41	38.75
1000-100	607878	608202.3	160.76	607992.563	106.5	607880.4	148.8	607883.9	73.21	134.62
1000-1000	220560	221260.9	175.71	220626.563	84.4	220560.9	141.7	220585.2	159.42	273.47
1500-10	2000801	2002874.5	178.66	2000854.14	331.1	2001121.7	347.8	2000828.8	70.89	111.23
1500-100	866454	867654.7	385.46	867149.69	293.6	866493.2	378.7	866490.1	196.11	314.24
1500-1000	334962	338046.1	381.83	335400.813	218.7	334973.2	387.2	335059.1	361.28	702.70
2000-10	2558118	2559611.3	224.77	2558121.5	687.4	2558120.8	717.5	2558119.4	93.39	277.50
2000-100	1122748	1125471.9	980.89	1123936.5	562.3	1122861.9	650.8	1122854.5	311.26	660.59
2000-1000	437686	443025.7	997.64	438263.0	425.9	437690.7	760	437813.6	527.60	1201.25
2500-10	3099907	3107032.5	710.01	3100174.5	1116.7	3100224.7	1419.5	3100155.9	259.50	531.12
2500-100	1347516	1350446.6	1903.79	1348713.25	870.5	1347577.6	1128.2	1347566.8	711.44	1036.96
2500-1000	534405	540365.2	1940.58	535134.938	675.3	534426.6	1309.4	534593.6	1723.54	2737.16
3000-10	3570766	3579295.7	2605.87	3570820.75	1667.0	3570818.8	1621.1	3570766*	287.43	876.77
3000-100	1602154	1607502.6	2987.58	1605083.63	1349.5	1602530.9	1977.6	1602512.2	999.13	1711.03
3000-1000	643463	652092.7	2996.92	644376.25	1017.3	643541.8	2081.4	643885.3	2022.41	3389.68

instance the two methods obtain the same average solution value). The two methods require approximately the same average CPU-times for this problem set.

Table 5 compares the best solutions for problems from the MED test set, comparing MWA to ILS, where MWA executes 10 runs and ILS executes 30 runs. Here we present only the results for ILS and MWA because, unfortunately, the best solutions returned by MDAS and HYB are not reported for these particular problems. Table 5 shows that the best solutions found by MWA are better than the best solutions found by ILS in 15 out of 18 instances, and the two methods found the same optimal solutions for 3 instances.

From Table 4, we observe that: (1) For all instance q-10 ($q \in \{500, 1000, 1500, 2000, 2500, 3000\}$) the MWA performs very well by reaching the best average solution cost in all executions while requiring less CPU-time than the state-of-the-art methods from the literature. (2) For instances q-1000 ($q \in \{500, 1000, 1500, 2000, 2500, 3000\}$), the MWA provides high-quality solutions and outperforms the two state-of-the-art methods ILS and MDAS. We observe also that the CPU-times of MWA in these instances are slightly worse than the other methods. Finally, Table 5 shows that the MWA was able, out of 10 runs, to find the optimal solutions for 12 out of 18 large-scale instances (with up to 3000 customers and 3000 facilities).

Table 5 MWA best solutions versus ILS best solutions on MED test problems

	Gurobi	ILS	MWA
Instance	OPT	Best	Best
500-10	798577	798577*	798577*
500-100	326790	326894	326790*
500-1000	99169	99169*	99169*
1000-10	1434154	1434154*	1434154*
1000-100	607878	607939	607878*
1000-1000	220560	220959	220560*
1500-10	2000801	2001920	2000801*
1500-100	866454	866769	866454*
1500-1000	334962	337452	335002
2000-10	2558118	2558125	2558118*
2000-100	1122748	1124572	1122805
2000-1000	437686	442136	437693
2500-10	3099907	3102719	3099907*
2500-100	1347516	1349510	1347516*
2500-1000	534405	539485	534506
3000-10	3570766	3570930	3570766*
3000-100	1602154	1606424	1602345
3000-1000	643463	650777	643797

4.3.3 Results on M* and ORLIB test problems

The results of MWA versus the literature results on M* and ORLIB benchmarks are presented in Table 6 and Table 7 respectively. In Table 6 (for the M* test problems) we compare MWA to TS, LAG, US and HYB and for each instance show the solution costs and CPU-times reported by each method. In Table 7 (for the ORLIB test problems) we present the name of the instance and the optimal solution in columns one and two, then present the CPU-times of TS, LAG and HYB in columns three, four, and five respectively. The CPU-times of MWA are presented in columns six and seven. These tables show that the M* and ORLIB problems were easy to solve.

Table 6 shows that TS, US, and HYB, along with our MWA approach, were all able to reach every best-known solution for the M* problems in competitive CPU-times compared to the other state-of-the-art methods. The ORLIB problems turn out to be even easier to solve, since all state-of-the-art methods, without exception, are able to obtain optimal solutions to all problems. For this reason, Table 7 shows only the CPU-times. It may be noted that our MWA method requires significantly less average computation time for these problems than the other methods, demonstrating its efficiency in solving these problems as well as in the case of the more complex problems, where in addition it obtains better solution results.

5 Conclusions

Our simple version of the multi-wave algorithm for the uncapacitated facility location problem proves highly effective, as determined by computational tests comparing our method to nine state-of-the-art methods from the literature. The efficacy of our MWA algorithm derives from strategies for exploiting the conditional effects of sequential decisions, embodied in two alternating phases, a vertical phase that successively augments or improves a current solution, and a horizontal phase that modifies the current solution by replacing moves stored in an active move record (AMR). The alternation of vertical and horizontal phases continues until reaching a boundary solution, after which parameters are updated based on performance history and the method repeats. Our findings motivate an investigation of more advanced strategies of the general multi-wave algorithm that additionally exploit other conditional effects, including particularly the effect of persistent attractiveness.

Our findings suggest that the ideas we have incorporated in our multi-wave implementation for UFLP may likewise prove valuable in application to other combinatorial optimization problems. It has been beyond the scope of this study to investigate the full set of strategies available to the multi-wave algorithm, and consequently, our findings

Table 6 MWA results versus literature results on M* test problems

Instance	TS		LAG		US		HYB		MWA		
	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU *All*	***Find***
MO1	1305.95	0.68	1305.95	0.69	1305.95	0.07	1305.95	0.988	1305.95	0.24	0.08
MO2	1432.26	0.76	1445.70	0.73	1432.26	0.07	1432.26	1.030	1432.26	0.27	0.10
MO3	1516.77	0.80	1535.77	0.79	1516.77	0.07	1516.77	0.960	1516.77	0.30	0.12
MO4	1442.24	0.77	1442.24	0.79	1442.24	0.06	1442.24	0.892	1442.24	0.27	0.09
MO5	1408.77	0.80	1408.77	0.78	1408.77	0.06	1408.77	0.815	1408.77	0.34	0.09
MP1	2686.48	4.97	2722.66	3.40	2686.48	0.24	2686.48	3.695	2686.48	0.72	0.33
MP2	2904.86	5.00	2914.42	2.81	2904.86	0.28	2904.86	4.125	2904.86	0.78	0.47
MP3	2623.71	4.95	2623.71	3.06	2623.71	0.30	2623.71	3.500	2623.71	0.61	0.33
MP4	2938.75	5.82	2958.80	3.23	2938.75	0.32	2938.75	3.887	2938.75	0.70	0.34
MP5	2932.33	5.49	2946.03	3.06	2932.33	0.25	2932.33	4.169	2932.33	0.75	0.42
MQ1	4091.01	20.84	4091.01	3.96	4091.01	0.84	4091.01	8.919	4091.01	1.64	0.91
MQ2	4028.33	17.82	4096.13	4.34	4028.33	0.79	4028.33	7.802	4028.33	1.65	1.15
MQ3	4275.43	15.46	4373.08	3.78	4275.43	0.84	4275.43	9.508	4275.43	1.63	0.97
MQ4	4235.15	16.24	4274.68	4.06	4235.15	0.74	4235.15	9.834	4235.15	1.84	0.92
MQ5	4080.74	17.66	4138.50	3.87	4080.74	0.90	4080.74	10.813	4080.74	1.87	0.98
MR1	2608.15	71.33	2634.71	11.67	2608.15	3.04	2608.15	27.221	2608.15	8.05	4.03
MR2	2654.73	74.02	2704.66	11.37	2654.73	2.98	2654.73	27.646	2654.73	6.02	3.86
MR3	2788.25	83.13	2874.46	11.62	2788.25	3.20	2788.25	26.417	2788.25	4.74	4.10
MR4	2756.04	72.84	2774.38	12.54	2756.04	2.91	2756.04	27.595	2756.04	7.70	3.96
MR5	2505.05	75.91	2526.32	11.57	2505.05	3.08	2505.05	26.989	2505.05	7.80	4.54
MS1	5283.76	629.31	5434.11	50.19	5283.76	10.56	5283.76	113.395	5283.76	31.34	11.12
MT1	10069.80	6173.04	10172.55	189.69	10069.80	120.58	10069.80	701.167	10069.80	180.33	78.47
#BKS	22/22		5/22		22/22		**22/22**		22/22		

Table 7 MWA results versus literature results on ORLIB test problems

Instance	Optimal	TS	LAG	HYB	MWA *All*	*Find*
Cap71	932 615.75	0.05	0.07	0.034	0.060	0.020
Cap72	977 799.40	0.05	0.08	0.039	0.060	0.025
Cap73	1 010 641.45	0.05	0.08	0.053	0.045	0.045
Cap74	1 034 976.97	0.04	0.07	0.049	0.050	0.030
Cap101	796 648.44	0.07	0.07	0.055	0.095	0.055
Cap102	854 704.20	0.06	0.12	0.056	0.085	0.030
Cap103	893 782.11	0.05	0.10	0.072	0.080	0.050
Cap104	928 941.75	0.06	0.06	0.077	0.075	0.055
Cap131	793 439.56	0.11	0.20	0.105	0.145	0.055
Cap132	851 495.32	0.12	0.19	0.097	0.115	0.055
Cap133	893 076.71	0.13	0.28	0.131	0.130	0.040
Cap134	928 941.75	0.13	0.20	0.140	0.125	0.085
Capa	17 156 454.48	13.02	7.97	7.380	1.545	1.190
Capb	12 979 071.58	10.38	8.69	6.245	1.687	1.492
Capc	11 505 594.33	9.23	9.19	6.148	1.805	1.190
Avg	-	2.24	1.82	1.38	0.41	0.29

also motivate an exploration of other multi-wave components.

References

Ahn S, Cooper C, Cornuéjols G, Frieze A (1988). Probabilistic analysis of a relaxation for the k-median problem. Mathematics of Operations Research, 13(1): 1–31

Akbaripour H, Masehian E, Roostaei A (2017). Landscape analysis and scatter search metaheuristic for solving the uncapacitated single allocation hub location problem. International Journal of Industrial and Systems Engineering, 26(4): 425–459

Al-Sultan K S, Al-Fawzan M A (1999). A tabu search approach to the uncapacitated facility location problem. Annals of Operations Research, 86: 91–103

Albareda-Sambola M, Fernández E, Saldanha-da-Gama F (2017). Heuristic solutions to the facility location problem with general bernoulli demands. INFORMS Journal on Computing, 29(4): 737–753

Amin S H, Zhang G (2013). A multi-objective facility location model for closed-loop supply chain network under uncertain demand and return. Applied Mathematical Modelling, 37(6): 4165–4176

An H C, Svensson O (2017). Recent developments in approximation algorithms for facility location and clustering problems. In: Fukunaga T, Kawarabayashi K, eds. Combinatorial Optimization and Graph Algorithms, 1–19. Singapore: Springer

Ardjmand E, Park N, Weckman G, Amin-Naseri M R (2014). The discrete unconscious search and its application to uncapacitated facility location problem. Computers & Industrial Engineering, 73: 32–40

Arostegui M A Jr, Kadipasaoglu S N, Khumawala B M (2006). An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems. International Journal of Production Economics, 103(2): 742–754

Atta S, Mahapatra P R S, Mukhopadhyay A (2018). Deterministic and randomized heuristic algorithms for uncapacitated facility location problem. Information and Decision Sciences, 205–216. Singapore: Springer

Barahona F, Chudak F (1999). Near-optimal Solutions to Large Scale Facility Location Problems. Technical Report RC21606, IBM, USA

Basu S, Sharma M, Ghosh P S (2015). Metaheuristic applications on discrete facility location problems: A survey. OPSEARCH, 52(3): 530–561

Beasley J E (1990). OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society, 41(11): 1069–1072

Beasley J E (1993). Lagrangean heuristics for location problems. European Journal of Operational Research, 65(3): 383–399

Beltran C, Tadonki C, Vial J Ph (2006). Solving the *p*-median problem with a semi-Lagrangian relaxation. Computational Optimization and Applications, 35(2): 239–260

Beltran-Royo C, Vial J P, Alonso-Ayuso A (2012). Semi-lagrangian relaxation applied to the uncapacitated facility location problem. Computational Optimization and Applications, 51(1): 387–409

Blum C, Roli A (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, 35 (3): 268–308

Cerrone C, Cerulli R, Golden B (2017). Carousel greedy: A generalized greedy algorithm with applications in optimization and statistics. Computers & Operations Research, 85: 97–112

Chalupa D, Nielsen P (2018). Instance scale, numerical properties and design of metaheuristics: A study for the facility location problem. arXiv preprint arXiv:1801.03419

Chen L, Olhager J, Tang O (2014). Manufacturing facility location and sustainability: A literature review and research agenda. International Journal of Production Economics, 149: 154–163

Cheung B S, Langevin A, Villeneuve B (2001). High performing evolutionary techniques for solving complex location problems in industrial system design. Journal of Intelligent Manufacturing, 12(5): 455–466

Cura T (2010). A parallel local search approach to solving the uncapacitated warehouse location problem. Computers & Industrial Engineering, 59(4): 1000–1009

De Armas J, Juan A A, Marques J M, Pedroso J P (2017). Solving the deterministic and stochastic uncapacitated facility location problem: From a heuristic to a simheuristic. Journal of the Operational Research Society, 68(10): 1161–1176

De Corte A, Sörensen K (2015). An iterated local search algorithm for water distribution network design optimization. Networks, 67(3): 187–198

Erlenkotter D (1978). A dual-based procedure for uncapacitated facility location: General solution procedures and computational experience. Operations Research, 26(6): 992–1009

Galli L, Letchford A N, Miller S J (2018). New valid inequalities and facets for the simple plant location problem. European Journal of Operational Research, 269(3): 824–833

Galvão R D, Raggi L A (1989). A method for solving to optimality uncapacitated location problems. Annals of Operations Research, 18 (1): 225–244

Ghosh D (2003). Neighborhood search heuristics for the uncapacitated facility location problem. European Journal of Operational Research, 150(1): 150–162

Glover F (1989). Tabu search—Part I. ORSA Journal on Computing, 1 (3): 190–206

Glover F (1995). Tabu thresholding: Improved search by nonmonotonic trajectories. ORSA Journal on Computing, 7(4): 426–442

Glover F (2000). Multi-start and strategic oscillation methods—principles to exploit adaptive memory. In: Laguna M, Gonzales-Velarde J L, eds. Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, 1–24. Berlin: Kluwer Academic Publishers

Glover F (2016). The multi-wave algorithm for metaheuristic optimization. Journal of Heuristics, 22(3): 331–358

Glover F, Laguna M (1997). Tabu Search. Boston: Kluwer Academic Publishers

Goldengorin B, Ghosh D, Sierksma G (2003). Branch and peg algorithms for the simple plant location problem. Computers & Operations Research, 30(7): 967–981

Grasas A, Juan A A, Faulin J, de Armas J, Ramalhinho H (2017). Biased randomization of heuristics using skewed probability distributions: A survey and some applications. Computers & Industrial Engineering, 110: 216–228

Greistorfer P, Rego C (2006). A simple filter-and-fan approach to the facility location problem. Computers & Operations Research, 33(9): 2590–2601

Hale T S, Moberg C R (2003). Location science research: A review. Annals of Operations Research, 123(1/4): 21–35

Han L, Xu D, Du D, Zhang D (2018). A local search approximation algorithm for the uniform capacitated k-facility location problem. Journal of Combinatorial Optimization, 35(2): 409–423

Homberger J, Gehring H (2008). A two-level parallel genetic algorithm for the uncapacitated warehouse location problem. In: Proceedings of Hawaii international conference on system sciences

Jiang Y, Xu D, Du D, Wu C, Zhang D (2018). An approximation algorithm for soft capacitated k-facility location problem. Journal of Combinatorial Optimization, 35(2): 493–511

Jörnsten K, Klose A (2016). An improved Lagrangian relaxation and dual ascent approach to facility location problems. Computational Management Science, 13(3): 317–348

Khumawala B M (1972). An efficient branch and bound algorithm for the warehouse location problem. Management Science, 18(12): 718–731

Klose A, Drexl A (2005). Facility location models for distribution system design. European Journal of Operational Research, 162(1): 4–29

Körkel M (1989). On the exact solution of large-scale simple plant location problems. European Journal of Operational Research, 39(2): 157–173

Kratica J, Tošic D, Filipović V, Ljubić I (2001). Solving the simple plant location problem by genetic algorithm. Operations Research, 35(1): 127–142

Melo M T, Nickel S, Saldanha-Da-Gama F (2009). Facility location and supply chain management–A review. European Journal of Operational Research, 196(2): 401–412

Michel L, Van Hentenryck P (2004). A simple tabu search for warehouse location. European Journal of Operational Research, 157(3): 576–591

Ortiz-Astorquiza C, Contreras I, Laporte G (2017a). Formulations and approximation algorithms for multilevel uncapacitated facility location. INFORMS Journal on Computing, 29(4): 767–779

Ortiz-Astorquiza C, Contrerasn I, Laporte G (2017b). Multi-level facility location problems. European Journal of Operational Research, 267(3): 791–805

Pearce R H, Forbes M (2018). Disaggregated Benders decomposition and branch-and-cut for solving the budget-constrained dynamic uncapacitated facility location and network design problem. European Journal of Operational Research, 270(1): 78–88

Posta M, Ferland J A, Michelon P (2014). An exact cooperative method for the uncapacitated facility location problem. Mathematical Programming Computation, 6(3): 199–231

Resende M G, Werneck R F (2004). A hybrid heuristic for the p-median problem. Journal of Heuristics, 10(1): 59–88

Resende M G C, Ribeiro C C (2003). Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G, eds. Handbook of Metaheuristics, 219–249. Berlin: Kluwer Academic Publishers

Resende M G C, Werneck R F (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. European Journal of Operational Research, 174(1): 54–68

Resende M G C, Werneck R F (2007). A fast swap-based local search procedure for location problems. Annals of Operations Research, 150(1): 205–230

ReVelle C S, Eiselt H A, ReVelleC S (2005). Location analysis: A synthesis and survey. European Journal of Operational Research, 165(1): 1–19

Sahman M A, Altun A A, Dündar A O (2017). The binary differential search algorithm approach for solving uncapacitated facility location problems. Journal of Computational and Theoretical Nanoscience, 14(1): 670–684

Sun M (2005). A tabu search heuristic procedure for the uncapacitated facility location problem. In: Rego C, Alidaee B, eds. Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search, 191–211. Boston: Kluwer Academic Publishers

Sun M (2006). Solving the uncapacitated facility location problem using tabu search. Computers & Operations Research, 33(9): 2563–2589

Todosijević R, Urošević D, Mladenović N, Hanafi S (2017). A general variable neighborhood search for solving the uncapacitated r-allocation p-hub median problem. Optimization Letters, 11(6): 1109–1121

Tohyama H, Ida K, Matsueda J (2011). A genetic algorithm for the uncapacitated facility location problem. Electronics and Communications in Japan, 94(5): 47–54

Tseng L Y, Wu C S (2009a). Multiple trajectory search for uncapacitated facility location problems. In: Proceedings of the Second International Joint Conference on Computational Sciences and Optimization, Sanya, China. 2: 965–968

Tseng L Y, Wu C S (2009b). The multistart drop-add-swap heuristic for the uncapacitated facility location problem. In: Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics, Intelligent Control Systems and Optimization, Milan, Italy. 21–28

Tsuya K, Takaya M, Yamamura A (2017). Application of the firefly algorithm to the uncapacitated facility location problem. Journal of Intelligent & Fuzzy Systems, 32(4): 3201–3208

Wang D, Wu C H, Ip A, Wang D, Yan Y (2008). Parallel multipopulation particle swarm optimization algorithm for the uncapacitated facility location problem using OpenMP. In: Proceedings of 2008 IEEE Congress on Evolutionary Computation, IEEE World Congress on Computational Intelligence. 1214–1218

Wang Y, Lu Z, Glover F, Hao J K (2013). Probabilistic GRASP-tabu search algorithms for the UBQP problem. Computers & Operations Research, 40(12): 3100–3107

Xu J, Chiu S Y, Glover F (1997). Probabilistic tabu search for telecommunications network design. Combinatorial Optimization: Theory and Practice, 1(1): 69–94

Yigit V, Aydin M E, Turkbey O (2006). Solving large-scale uncapacitated facility location problems with evolutionary simulated annealing. International Journal of Production Research, 44(22): 4773–4791