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

© The Author(s) 2018. Published by Higher Education Press. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0)

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 \( x^0 \) 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) \( x^* = \emptyset ; f(x^*) = +\infty \);
2) For Pass = 1 to MaxPass do
3) Generate a starting solution \( x^0 \);
4) For Wave = 1 to MaxWave do
5) \( x = x^0 \);
6) While \( x \) is not a boundary solution do \( x = m(x) \) where \( m \) is a move in \( CL \);
7) \( x^* = \arg\min \{ f(x), f(x^*) \} \);
8) Update relevant search parameters for the next Wave;
9) EndFor
The cost of opening a facility is determined by the number of customers and the fixed cost of facilities that are opened in a solution. The decision on which customers to serve and which facilities to open can be modeled as a discrete location problem. The Uncapacitated Facility Location Problem (UFLP) is one of the most widely studied discrete location problems in operations research.

\[ f(O) = \sum_{i \in O} f_i + \sum_{j \in P} \min_{i \in O} c_{ij}. \]

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 \( Q^i = Q^i(x) = \{i \in Q : x_i = 1\} \) to represent the facilities that are opened in a solution \( x \). Hence, the optimization problem of interest may be expressed in the form

\[ \text{Minimize } f(x) = \sum_{i \in Q^i(x)} f_i + \sum_{j \in P} \min_{i \in Q^i(x)} c_{ij}; x \in \{0,1\}^q - \{0\}. \]

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 Drexel (2005), ReVelle and Eisele (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 Goldbergin 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-Lagrangian 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 (Kratika et al., 2001; Homberger and Gehring, 2008; Tohyama et al., 2011). Other types of approaches include those of Greistrofer 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 Arosteegui 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 AlbareS-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 \( CL = \{m_1, m_2, \ldots, m_s\} \) so that \( \Delta(m_1) \geq \ldots \geq \Delta(m_s) \geq \max\{\Delta(m) \mid m \in M(x) - CL\} \). Often the size \( s = |CL| \) is determined by a threshold \( T \) (Glover, 1989, 1995), i.e. \( CL = \{m \in M(x) \mid \Delta(m) \geq T\} \) where in the present work, the value of \( T \) is chosen as

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

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

and where \( \Delta_{\text{min}}, \Delta_{\text{mean}} \) and \( \Delta_{\text{max}} \) 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 \( \Delta_{\text{mean}} \) by defining \( T = \Delta_{\text{min}} + \alpha(\Delta_{\text{max}} - \Delta_{\text{min}}) \), 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 \( \Delta_{\text{mean}} \), then a rough correspondence between this latter candidate list and those based on \( \Delta_{\text{mean}} \) occurs by specifying \( \alpha = (\lambda + 1)/2 \) (hence \( \alpha \in [0.5, 1] \)) 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) based on 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 \( \beta \) 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 \( 1\text{-flip} \) 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 \( 1\text{-flip} \) move efficiently, for each customer \( j \), we maintain the closest facility and the second closest facility, i.e.,

\[
i'^{ij}_j(x) = \text{argmin}\{c_{ij} : i \in Q^j(x)\},
\]

\[
i'^{ij}_j(x) = \text{argmin}\{c_{ij} : i \in Q^j(x) - \{i'^{ij}_j(x)\}\}.
\]
Then the objective function can be rewritten as
\[ f(x) = \sum_{i \in Q} f_i + \sum_{j \in P} c_{ij}^P. \]
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
\[ f(x') = \sum_{i \in Q} f_i + f_k + \sum_{j \in P} c_{ij}^P, \]
where \( P_k = \{ j \in P : i_j = k \} \). The change \( \Delta_k^* \) of closing facility \( k \) is given by
\[ \Delta_k^* = f_k - \sum_{j \in P_k} (c_{ij}^P - c_{ij}^P). \]

The objective function value after opening facility \( k \in Q^0 = Q^0(x) = \{ i \in Q : x_i = 0 \} \) is given by
\[ f(x') = \sum_{i \in Q} f_i + f_k + \sum_{j \in P} \min(c_{ij}^P - c_{ij}^P). \]
The change \( \Delta_k^+ \) of opening facility \( k \) is given by
\[ \Delta_k^+ = -f_k + \sum_{j \in P} \min\{0, c_{ij}^P - c_{ij}^P\} \]
\[ = -f_k + \sum_{j \in P} c_{ij}^P - c_{ij}^P, \]
where \( P_k = \{ j : c_{ij}^P < c_{ij}^P \} \). Hence, the change \( \Delta_k \) of flipping facility \( k \) can be expressed as
\[ \Delta_k = \begin{cases} \Delta_k^* & \text{if } x_k = 1 \\ \Delta_k^+ & \text{if } x_k = 0 \end{cases}. \]

Michel and Van Hentenryck (2004) and Sun (2006) provide an efficient procedure in \( O(p \log q) \) time to update the evaluation \( \Delta_k \) 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 \( f(x') = \sum_{i \in Q^0(x)} f_i - f_k + f_h + \sum_{j \in P \cap Q^0(x)} \min_{i \in Q^0(x) - k + h} c_{ij} \) \[ + \sum_{j \in P \cap Q^0(x) - k + h} c_{ij} \]
Note that
\[ \min_{i \in Q^0(x) - k + h} c_{ij} = \min\{c_{ij} : i \in Q^0(x) - k + h\} \]
\[ = \begin{cases} c_{ij} & \text{if } c_{ij} < \min\{c_{ji}, c_{ih}\} \\ c_{ij} & \text{if } c_{ij} = k \text{ and } c_{ij} < c_{ih} \\ c_{ih} & \text{otherwise} \end{cases}. \]
Let \( P^1 = \{ J \in P : c_{ij} < \min\{c_{ij}, c_{ih}\} \} \), \( P^2 = \{ J \in P : c_{ij} < c_{ih} \text{ and } c_{ij} < c_{ih} \} \) and \( P^3 = P - P^1 - P^2 \) be a partition of \( P \). Then the objective function change produced by swapping \( x_k \) and \( x_h \) is given by \( \Delta_{hk} = f(x) - f(x') \). More precisely, we have
\[ \Delta_{hk} = f_k - f_h + \sum_{j \in P} \left( \min_{i \in Q^0(x)} c_{ij} - \min_{i \in Q^0(x) - k + h} c_{ij} \right) \]
This last evaluation can be simplified as follows
\[ \Delta_{hk} = f_k - f_h + \sum_{j \in P} (c_{ij}^P - c_{ij}^P) \]
\[ + \sum_{j \in P} (c_{ij}^P - c_{ij}^P). \]
According to Resende and Werneck (2007), the objective function change can be rewritten in the following form
\[ \Delta_{hk} = \Delta_k^+ - \Delta_h^+ + \Delta_{hk}^+, \]
where
\[ \Delta_{hk}^+ = \sum_{j \in P} c_{ij}^P - \max(c_{ij}^P - c_{ij}^P), \]
and \( \Delta_{hk}^+ \) is called gain \( (k) \) which corresponds to the decrease in solution value due to the insertion of facility \( k \) (with no associated removal). Similarly, \( \Delta_{hk}^+ \) is called loss \( (h) \) corresponding to the increase in solution value due to the removal of facility \( h \) (with no associated insertion), while \( \Delta_{hk}^+ \) 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) \( \Delta_{hk}^+ \) is always positive or equal to 0; ii) If \( \Delta_{hk}^+ \) is negative, then there is an improvement when closing \( h \); iii) If \( \Delta_{hk}^+ \) is positive, then there is an improvement when opening \( h \).

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 \( \Delta_{hk}^+ > \Delta_{hk}^+ \) 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 \( m \)
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
to 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 \( d_k \) denoted 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 2 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) \( x = \emptyset ; x^* = x ; f(x^*) = \infty \);
2) For Pass = 1 to MaxPass do
3) Generate a starting solution \( x^o \) (null, randomly or by
post-analysis diversification);
4) For Wave = 1 to MaxWave do
5) \( x = x^o; AMR = \emptyset; k = 1; \)
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) \( x^* = \text{argmin}\{f(x), f(x^*)\} \);
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) \( x^* = \text{argmin}\{f(x), f(x^*)\} \);
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 \( \delta_k \) 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 \( \delta_k \) should satisfy \( \delta_k \geq d_k + 1 \). In the simplest case, we select a constant value \( \delta = \delta_k \geq 2 \), which therefore determines \( k^* = \lfloor r_e / \delta \rfloor \). This approach is useful for

![Fig. 1](vertical_phase.png) Vertical phase \( r = |AMR| < v_k \)

![Fig. 2](horizontal_phase.png) Horizontal phase \( r = |AMR| < v_k \)
densely populating the values \( v_k \) over the estimate \( r_c \), hence is suited to choosing \( k^\ast \) relatively large (in comparison to \( r_c \)).

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

Note that with the first technique (1), the value of \( \delta \) determines the total number of interventions \( k^\ast \). 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 \( \delta_k \) increases after each \( \eta \) waves \( \eta \in \{1, \ldots, 4\} \) starting with small value of \( \delta_1 \in \{2, 3\} \) (i.e., \( \delta_{k+\eta} = \delta_k + \tau \) with \( \tau \in \{1, \ldots, 6\} \)). To implement the second rule (2), based on the estimation of \( r_c \), 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 \( k^\ast \) by 1 at each wave (\( k^\ast = k^\ast + 1 \)), starting with a small value (in our implementation, \( k^\ast = 5 \)). Hence, we have \( v_k = v_{k-1} + \delta \) where \( \delta = \bar{r}/k^\ast \).

3.2 Determining the number of drop moves \( d_k \)

The value \( d_k \) identifying the number of moves to be dropped during the \( k \)th 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 \( k \)th 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 \( k \)th 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 \( \lambda \) and \( \beta \) values. If \( \lambda \) is chosen sufficiently large, e.g., in the interval \([0.98, 1.0] \), then the value of \( \beta \) will be largely irrelevant. In our implementation, at each pass, the value of \( \beta \) starts from 1.5 and ends at 0.5 and at each wave decreases by stepwise \( 1/\text{MaxWave} \), while the value of \( \lambda \) starts from 0.8 and ends at 0 and at each wave decreases by stepwise \( 0.8/\text{MaxWave} \). Starting from large \( \lambda \) and \( \beta \) 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 \( h_0 = \min(|AMR|/d_0, \bar{h}) \), where \( \bar{h} \) is the maximum number of iterations we imposed on \( h_0 \). Consequently, if \( |AMR|/d_0 > \bar{h} \) 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:

2) CLM: The Complete Local search with Memory of Ghosh (2003).
7) MTS: Multiple Trajectory Search of Tseng and Wu (2009a).

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 costs. 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 \( f_i = \sqrt{y} \), 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>
<thead>
<tr>
<th>Instance</th>
<th>MWA</th>
<th>MultiS-FI</th>
<th>MultiS-BI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td></td>
<td>Avg</td>
<td>Avg</td>
</tr>
<tr>
<td>Nbr_it</td>
<td></td>
<td>best</td>
<td>best</td>
</tr>
<tr>
<td>500-10</td>
<td>798577.0</td>
<td>798577*</td>
<td>521.2</td>
</tr>
<tr>
<td>500-100</td>
<td>327068.8</td>
<td>327001</td>
<td>1191.7</td>
</tr>
<tr>
<td>500-1000</td>
<td>99175.8</td>
<td>99170</td>
<td>2767.3</td>
</tr>
<tr>
<td>1000-10</td>
<td>1434250.7</td>
<td>1434154*</td>
<td>442.6</td>
</tr>
<tr>
<td>1000-100</td>
<td>608008.3</td>
<td>607903</td>
<td>1407.0</td>
</tr>
<tr>
<td>1000-1000</td>
<td>220703</td>
<td>220668</td>
<td>2698.2</td>
</tr>
<tr>
<td>1500-10</td>
<td>2000839.6</td>
<td>2000801*</td>
<td>452.6</td>
</tr>
<tr>
<td>1500-100</td>
<td>867063.7</td>
<td>866829</td>
<td>1021.2</td>
</tr>
<tr>
<td>1500-1000</td>
<td>335579.5</td>
<td>335491</td>
<td>1893.3</td>
</tr>
<tr>
<td>2000-10</td>
<td>2558118.0</td>
<td>2558118*</td>
<td>533.1</td>
</tr>
<tr>
<td>2000-100</td>
<td>1123893.3</td>
<td>1123172</td>
<td>1121.6</td>
</tr>
<tr>
<td>2000-1000</td>
<td>438498.7</td>
<td>438295</td>
<td>1216.0</td>
</tr>
<tr>
<td>2500-10</td>
<td>3100172.0</td>
<td>3099907*</td>
<td>550.0</td>
</tr>
<tr>
<td>2500-100</td>
<td>1348372.0</td>
<td>1347790</td>
<td>996.7</td>
</tr>
<tr>
<td>2500-1000</td>
<td>535364.9</td>
<td>535132</td>
<td>1581.9</td>
</tr>
<tr>
<td>3000-10</td>
<td>3570904.8</td>
<td>3570766</td>
<td>534.2</td>
</tr>
<tr>
<td>3000-100</td>
<td>1604408.9</td>
<td>1604011</td>
<td>1006.8</td>
</tr>
<tr>
<td>3000-1000</td>
<td>644839.1</td>
<td>644718</td>
<td>1133.1</td>
</tr>
<tr>
<td>AVG</td>
<td>1170.47</td>
<td>928.72</td>
<td>8.22</td>
</tr>
</tbody>
</table>

As previously noted, this benchmark contains hard test problems. We compare the results of the MWA on these
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 \( q \in \{500, 1000, 1500, 2000, 2500, 3000\} \) and \( y = 10 \) the CPU-times of MWA are better than those of MDAS. Interestingly, for the indicated problems from the MED 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 2 MWA results versus literature results on GHOSH benchmark instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>TS</th>
<th>MDAS</th>
<th>GTS</th>
<th>CLM</th>
<th>HYB</th>
<th>MTS</th>
<th>US</th>
<th>MWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>250-sym-A</td>
<td>257805.0</td>
<td>257804.0</td>
<td>257832.6</td>
<td>257895.2</td>
<td>257806.2</td>
<td>257804.0</td>
<td>257804.0</td>
<td>257804.0</td>
</tr>
<tr>
<td>250-sym-B</td>
<td>276305.2</td>
<td>276305.2</td>
<td>276185.2</td>
<td>276352.2</td>
<td>276305.2</td>
<td>276305.2</td>
<td>276305.2</td>
<td>276305.2</td>
</tr>
<tr>
<td>250-sym-C</td>
<td>333671.6</td>
<td>333671.6</td>
<td>333820.0</td>
<td>333671.6</td>
<td>333671.6</td>
<td>333671.6</td>
<td>333671.6</td>
<td>333671.6</td>
</tr>
<tr>
<td>250-asm-A</td>
<td>257917.8</td>
<td>257917.8</td>
<td>257978.4</td>
<td>258032.6</td>
<td>257923.4</td>
<td>257917.8</td>
<td>257917.8</td>
<td>257917.8</td>
</tr>
<tr>
<td>250-asm-B</td>
<td>276053.2</td>
<td>276053.2</td>
<td>276467.2</td>
<td>276184.2</td>
<td>276053.2</td>
<td>276053.2</td>
<td>276053.2</td>
<td>276053.2</td>
</tr>
<tr>
<td>250-asm-C</td>
<td>332897.2</td>
<td>332897.2</td>
<td>333237.6</td>
<td>333058.4</td>
<td>332897.2</td>
<td>332897.2</td>
<td>332897.2</td>
<td>332897.2</td>
</tr>
<tr>
<td>500-sym-A</td>
<td>511180.4</td>
<td>511181.2</td>
<td>511383.6</td>
<td>511487.2</td>
<td>511196.4</td>
<td>511188.8</td>
<td>511180.4</td>
<td>511180.4</td>
</tr>
<tr>
<td>500-sym-B</td>
<td>537912.0</td>
<td>537912.0</td>
<td>538480.4</td>
<td>538685.8</td>
<td>537912.0</td>
<td>537912.0</td>
<td>537912.0</td>
<td>537912.0</td>
</tr>
<tr>
<td>500-sym-C</td>
<td>621059.2</td>
<td>621059.2</td>
<td>621107.2</td>
<td>621172.8</td>
<td>621059.2</td>
<td>621059.2</td>
<td>621059.2</td>
<td>621059.2</td>
</tr>
<tr>
<td>500-asm-A</td>
<td>511140.0</td>
<td>511136.4</td>
<td>511251.6</td>
<td>511393.4</td>
<td>511145.0</td>
<td>511137.8</td>
<td>511137.8</td>
<td>511137.8</td>
</tr>
<tr>
<td>500-asm-B</td>
<td>537847.6</td>
<td>537847.6</td>
<td>538144.0</td>
<td>538421.0</td>
<td>537863.4</td>
<td>537847.6</td>
<td>537847.6</td>
<td>537847.6</td>
</tr>
<tr>
<td>500-asm-C</td>
<td>621463.8</td>
<td>621463.8</td>
<td>621811.8</td>
<td>621990.8</td>
<td>621463.8</td>
<td>621463.8</td>
<td>621463.8</td>
<td>621463.8</td>
</tr>
<tr>
<td>750-sym-A</td>
<td>763693.4</td>
<td>763684.8</td>
<td>763830.8</td>
<td>763978.0</td>
<td>763706.6</td>
<td>763708.8</td>
<td>763684.8</td>
<td>763684.8</td>
</tr>
<tr>
<td>750-sym-B</td>
<td>796571.8</td>
<td>796571.8</td>
<td>796919.0</td>
<td>797173.4</td>
<td>796632.2</td>
<td>796571.8</td>
<td>796576.8</td>
<td>796571.8</td>
</tr>
<tr>
<td>750-asm-A</td>
<td>900158.6</td>
<td>900158.6</td>
<td>900158.6</td>
<td>900158.6</td>
<td>900158.6</td>
<td>900158.6</td>
<td>900158.6</td>
<td>900158.6</td>
</tr>
<tr>
<td>750-asm-B</td>
<td>763717.0</td>
<td>763716.4</td>
<td>763836.8</td>
<td>764019.4</td>
<td>763731.2</td>
<td>763735.8</td>
<td>763712.4</td>
<td>763712.4</td>
</tr>
<tr>
<td>#BKS</td>
<td>14/18</td>
<td>16/18</td>
<td>0/18</td>
<td>1/18</td>
<td>8/18</td>
<td>14/18</td>
<td>15/18</td>
<td>18/18</td>
</tr>
</tbody>
</table>
## Table 3  MWA CPU-times versus literature CPU-times on GHOSH benchmark instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>TS</th>
<th>MDAS</th>
<th>CLM</th>
<th>HYB</th>
<th>MTS</th>
<th>US</th>
<th>MWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Find</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>250-sym-A</td>
<td>2.828</td>
<td>7.36</td>
<td>86.842</td>
<td>4.328</td>
<td>5.59</td>
<td>0.843</td>
<td>1.94</td>
</tr>
<tr>
<td>250-sym-B</td>
<td>5.628</td>
<td>5.88</td>
<td>34.634</td>
<td>7.774</td>
<td>7.73</td>
<td>0.618</td>
<td>0.75</td>
</tr>
<tr>
<td>250-sym-C</td>
<td>9.878</td>
<td>5.73</td>
<td>69.458</td>
<td>8.702</td>
<td>8.49</td>
<td>0.63</td>
<td>0.435</td>
</tr>
<tr>
<td>250-asym-A</td>
<td>2.618</td>
<td>7.32</td>
<td>86.506</td>
<td>4.636</td>
<td>5.62</td>
<td>0.993</td>
<td>1.84</td>
</tr>
<tr>
<td>250-asym-B</td>
<td>5.79</td>
<td>5.92</td>
<td>33.688</td>
<td>8.082</td>
<td>7.33</td>
<td>0.6</td>
<td>0.75</td>
</tr>
<tr>
<td>250-asym-C</td>
<td>9.196</td>
<td>5.67</td>
<td>89.99</td>
<td>7.776</td>
<td>8.51</td>
<td>0.586</td>
<td>0.39</td>
</tr>
<tr>
<td>500-sym-A</td>
<td>15.616</td>
<td>35.87</td>
<td>946.028</td>
<td>27.644</td>
<td>27.4</td>
<td>8.201</td>
<td>14.72</td>
</tr>
<tr>
<td>500-sym-B</td>
<td>31.432</td>
<td>26.72</td>
<td>294.656</td>
<td>34.196</td>
<td>32.7</td>
<td>4.619</td>
<td>4.29</td>
</tr>
<tr>
<td>500-sym-C</td>
<td>71.106</td>
<td>24.18</td>
<td>437.462</td>
<td>40.376</td>
<td>36.72</td>
<td>3.46</td>
<td>2.27</td>
</tr>
<tr>
<td>500-asym-A</td>
<td>13.76</td>
<td>35.97</td>
<td>921.208</td>
<td>20.232</td>
<td>27.43</td>
<td>7.736</td>
<td>11.81</td>
</tr>
<tr>
<td>500-asym-B</td>
<td>34.748</td>
<td>26.73</td>
<td>311.344</td>
<td>31.3</td>
<td>32.91</td>
<td>4.751</td>
<td>3.95</td>
</tr>
<tr>
<td>500-asym-C</td>
<td>72.064</td>
<td>26.60</td>
<td>388.21</td>
<td>47.79</td>
<td>36.61</td>
<td>3.601</td>
<td>2.31</td>
</tr>
<tr>
<td>AVG CPU</td>
<td>56.10</td>
<td>35.95</td>
<td>928.87</td>
<td>41.33</td>
<td>39.09</td>
<td>7.80</td>
<td>8.73</td>
</tr>
</tbody>
</table>

## Table 4  MWA results versus literature results on MED test problems

<table>
<thead>
<tr>
<th>Instance</th>
<th>OPT</th>
<th>ILS</th>
<th>CPU</th>
<th>OPT</th>
<th>ILS</th>
<th>CPU</th>
<th>OPT</th>
<th>ILS</th>
<th>CPU</th>
<th>AVG CPU</th>
<th>MWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>500-10</td>
<td>798577</td>
<td>799077.1</td>
<td>46.89</td>
<td>798577.0*</td>
<td>28.0</td>
<td>798577*</td>
<td>33.2</td>
<td>798577.0*</td>
<td>1.488</td>
<td>8.22</td>
<td></td>
</tr>
<tr>
<td>500-100</td>
<td>326790</td>
<td>327007.6</td>
<td>18.54</td>
<td>326922.375</td>
<td>25.6</td>
<td>326805.4</td>
<td>32.7</td>
<td>326798</td>
<td>4.619</td>
<td>9.05</td>
<td></td>
</tr>
<tr>
<td>500-1000</td>
<td>99169</td>
<td>99172.6</td>
<td>14.57</td>
<td>99194.34</td>
<td>22.4</td>
<td>99194.34</td>
<td>22.4</td>
<td>99194.34</td>
<td>3.46</td>
<td>2.27</td>
<td></td>
</tr>
<tr>
<td>750-10</td>
<td>1434154</td>
<td>1435918.6</td>
<td>54.94</td>
<td>1434171</td>
<td>130.7</td>
<td>1434185.4</td>
<td>173.9</td>
<td>1434154*</td>
<td>11.37</td>
<td>4.1</td>
<td></td>
</tr>
<tr>
<td>750-100</td>
<td>607878</td>
<td>608202.3</td>
<td>160.76</td>
<td>607992.563</td>
<td>106.5</td>
<td>607880.4</td>
<td>148.8</td>
<td>607883.9</td>
<td>3.95</td>
<td>0.53</td>
<td></td>
</tr>
<tr>
<td>750-1000</td>
<td>220560</td>
<td>221260.9</td>
<td>175.71</td>
<td>220626.563</td>
<td>84.4</td>
<td>220560.9</td>
<td>141.7</td>
<td>220585.2</td>
<td>14.54</td>
<td>5.03</td>
<td></td>
</tr>
<tr>
<td>AVG CPU</td>
<td>56.10</td>
<td>35.95</td>
<td>928.87</td>
<td>41.33</td>
<td>39.09</td>
<td>7.80</td>
<td>8.73</td>
<td>3.99</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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).

### 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.

<table>
<thead>
<tr>
<th>Instance</th>
<th>OPT</th>
<th>Best</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>500-10</td>
<td>798577</td>
<td>798577</td>
<td>798577*</td>
</tr>
<tr>
<td>500-100</td>
<td>326790</td>
<td>326894</td>
<td>326790*</td>
</tr>
<tr>
<td>500-1000</td>
<td>99169</td>
<td>99169*</td>
<td>99169*</td>
</tr>
<tr>
<td>1000-10</td>
<td>1434154</td>
<td>1434154*</td>
<td>1434154*</td>
</tr>
<tr>
<td>1000-100</td>
<td>607878</td>
<td>607939</td>
<td>607878*</td>
</tr>
<tr>
<td>1000-1000</td>
<td>220560</td>
<td>220959</td>
<td>220560*</td>
</tr>
<tr>
<td>1500-10</td>
<td>2000801</td>
<td>2001920</td>
<td>2000801*</td>
</tr>
<tr>
<td>1500-100</td>
<td>866454</td>
<td>866769</td>
<td>866454*</td>
</tr>
<tr>
<td>1500-1500</td>
<td>334962</td>
<td>337452</td>
<td>335002</td>
</tr>
<tr>
<td>2000-10</td>
<td>2558118</td>
<td>2558125</td>
<td>2558118*</td>
</tr>
<tr>
<td>2000-100</td>
<td>1122748</td>
<td>1124572</td>
<td>1122505</td>
</tr>
<tr>
<td>2000-1000</td>
<td>437686</td>
<td>442136</td>
<td>437693</td>
</tr>
<tr>
<td>2500-10</td>
<td>3099907</td>
<td>3102719</td>
<td>3099907*</td>
</tr>
<tr>
<td>2500-100</td>
<td>1347516</td>
<td>1349510</td>
<td>1347516*</td>
</tr>
<tr>
<td>2500-1000</td>
<td>534405</td>
<td>539485</td>
<td>534506</td>
</tr>
<tr>
<td>3000-10</td>
<td>3570766</td>
<td>3570930</td>
<td>3570766*</td>
</tr>
<tr>
<td>3000-100</td>
<td>1602154</td>
<td>1606424</td>
<td>1602345</td>
</tr>
<tr>
<td>3000-1000</td>
<td>643463</td>
<td>650777</td>
<td>643797</td>
</tr>
</tbody>
</table>

## 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

<table>
<thead>
<tr>
<th>Instance</th>
<th>TS</th>
<th>CPU</th>
<th>LAG</th>
<th>CPU</th>
<th>US</th>
<th>CPU</th>
<th>HYB</th>
<th>CPU</th>
<th>MWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MO1</td>
<td>1305.95</td>
<td>0.68</td>
<td>1305.95</td>
<td>0.69</td>
<td>1305.95</td>
<td>0.07</td>
<td>1305.95</td>
<td>0.98</td>
<td>1305.95</td>
</tr>
<tr>
<td>MO2</td>
<td>1432.26</td>
<td>0.76</td>
<td>1445.70</td>
<td>0.73</td>
<td>1432.26</td>
<td>0.07</td>
<td>1432.26</td>
<td>1.03</td>
<td>1432.26</td>
</tr>
<tr>
<td>MO3</td>
<td>1516.77</td>
<td>0.80</td>
<td>1535.77</td>
<td>0.79</td>
<td>1516.77</td>
<td>0.07</td>
<td>1516.77</td>
<td>0.96</td>
<td>1516.77</td>
</tr>
<tr>
<td>MO4</td>
<td>1442.24</td>
<td>0.77</td>
<td>1442.24</td>
<td>0.79</td>
<td>1442.24</td>
<td>0.06</td>
<td>1442.24</td>
<td>0.89</td>
<td>1442.24</td>
</tr>
<tr>
<td>MO5</td>
<td>1408.77</td>
<td>0.80</td>
<td>1408.77</td>
<td>0.78</td>
<td>1408.77</td>
<td>0.06</td>
<td>1408.77</td>
<td>0.81</td>
<td>1408.77</td>
</tr>
<tr>
<td>MP1</td>
<td>2686.48</td>
<td>4.97</td>
<td>2722.66</td>
<td>3.40</td>
<td>2686.48</td>
<td>0.24</td>
<td>2686.48</td>
<td>3.69</td>
<td>2686.48</td>
</tr>
<tr>
<td>MP2</td>
<td>2904.86</td>
<td>5.00</td>
<td>2914.42</td>
<td>2.81</td>
<td>2904.86</td>
<td>0.28</td>
<td>2904.86</td>
<td>4.12</td>
<td>2904.86</td>
</tr>
<tr>
<td>MP3</td>
<td>2623.71</td>
<td>4.95</td>
<td>2623.71</td>
<td>3.06</td>
<td>2623.71</td>
<td>0.30</td>
<td>2623.71</td>
<td>3.50</td>
<td>2623.71</td>
</tr>
<tr>
<td>MP4</td>
<td>2938.75</td>
<td>5.82</td>
<td>2958.80</td>
<td>3.23</td>
<td>2938.75</td>
<td>0.32</td>
<td>2938.75</td>
<td>3.88</td>
<td>2938.75</td>
</tr>
<tr>
<td>MP5</td>
<td>2932.33</td>
<td>5.49</td>
<td>2946.03</td>
<td>3.06</td>
<td>2932.33</td>
<td>0.25</td>
<td>2932.33</td>
<td>4.16</td>
<td>2932.33</td>
</tr>
<tr>
<td>MQ1</td>
<td>4091.01</td>
<td>20.84</td>
<td>4091.01</td>
<td>3.96</td>
<td>4091.01</td>
<td>0.84</td>
<td>4091.01</td>
<td>8.91</td>
<td>4091.01</td>
</tr>
<tr>
<td>MQ2</td>
<td>4028.33</td>
<td>17.82</td>
<td>4096.13</td>
<td>4.34</td>
<td>4028.33</td>
<td>0.79</td>
<td>4028.33</td>
<td>7.80</td>
<td>4028.33</td>
</tr>
<tr>
<td>MQ3</td>
<td>4275.43</td>
<td>15.46</td>
<td>4373.08</td>
<td>3.78</td>
<td>4275.43</td>
<td>0.84</td>
<td>4275.43</td>
<td>9.50</td>
<td>4275.43</td>
</tr>
<tr>
<td>MQ4</td>
<td>4235.15</td>
<td>16.24</td>
<td>4274.68</td>
<td>4.06</td>
<td>4235.15</td>
<td>0.74</td>
<td>4235.15</td>
<td>9.84</td>
<td>4235.15</td>
</tr>
<tr>
<td>MQ5</td>
<td>4080.74</td>
<td>17.66</td>
<td>4138.50</td>
<td>3.87</td>
<td>4080.74</td>
<td>0.90</td>
<td>4080.74</td>
<td>10.81</td>
<td>4080.74</td>
</tr>
<tr>
<td>MR1</td>
<td>2608.15</td>
<td>71.33</td>
<td>2634.71</td>
<td>11.67</td>
<td>2608.15</td>
<td>3.04</td>
<td>2608.15</td>
<td>27.22</td>
<td>2608.15</td>
</tr>
<tr>
<td>MR2</td>
<td>2654.73</td>
<td>74.02</td>
<td>2704.66</td>
<td>11.37</td>
<td>2654.73</td>
<td>2.98</td>
<td>2654.73</td>
<td>27.46</td>
<td>2654.73</td>
</tr>
<tr>
<td>MR3</td>
<td>2788.25</td>
<td>83.13</td>
<td>2874.46</td>
<td>11.62</td>
<td>2788.25</td>
<td>3.20</td>
<td>2788.25</td>
<td>26.41</td>
<td>2788.25</td>
</tr>
<tr>
<td>MR4</td>
<td>2756.04</td>
<td>72.84</td>
<td>2774.38</td>
<td>12.54</td>
<td>2756.04</td>
<td>2.91</td>
<td>2756.04</td>
<td>27.59</td>
<td>2756.04</td>
</tr>
<tr>
<td>MR5</td>
<td>2505.05</td>
<td>75.91</td>
<td>2526.32</td>
<td>11.57</td>
<td>2505.05</td>
<td>3.08</td>
<td>2505.05</td>
<td>26.98</td>
<td>2505.05</td>
</tr>
<tr>
<td>MS1</td>
<td>5283.76</td>
<td>629.31</td>
<td>5434.11</td>
<td>50.19</td>
<td>5283.76</td>
<td>10.56</td>
<td>5283.76</td>
<td>113.39</td>
<td>5283.76</td>
</tr>
<tr>
<td>MT1</td>
<td>10069.80</td>
<td>6173.04</td>
<td>10172.55</td>
<td>189.69</td>
<td>10069.80</td>
<td>120.58</td>
<td>10069.80</td>
<td>701.167</td>
<td>10069.80</td>
</tr>
<tr>
<td>Avg</td>
<td>-</td>
<td>2.24</td>
<td>1.82</td>
<td>1.38</td>
<td>0.41</td>
<td>0.29</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7  MWA results versus literature results on ORLIB test problems

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

References


