Solving the Uncapacitated Multiple Allocation Hub Location Problem by means of a dual-ascent technique

Lázaro Cánovas, Sergio García*, Alfredo Marín

Departamento de Estadística e Investigación Operativa
Universidad de Murcia, 30100 Murcia, Spain
E-mails: lcanovas@um.es, sgarcia@um.es, amarin@um.es
* Corresponding author

Abstract

This paper deals with the uncapacitated multiple allocation hub location problem. The dual problem of a four-indexed formulation is considered and a heuristic method, based on a dual-ascent technique, is designed. This heuristic, which is reinforced with several specifical subroutines and does not require any external linear problem solver, is the core tool embedded in an exact branch-and-bound framework. Besides, the heuristic provides the branch-and-bound algorithm with good lower bounds for the nodes of the branching tree. The results of the computational experience (with the classical CAB and AP data sets) are included, showing the great effectiveness of this approach: instances with up to 120 nodes are solved.

Keywords: Location; Integer programming; Hub location; Dual-ascent technique
1. Introduction

In hub location problems there is a complete network of nodes and directed arcs. There are flows which need to be sent between every pair of nodes and there is a cost per unit of flow on each arc. Every flow must be routed via either one or two special nodes called hubs. The hubs act as terminal nodes by collecting the flows from the origins and redistributing them towards the destinations.

These problems usually arise in fields such as post service, transportation and computer networks. Since there is an increase in the traffic between hub nodes, the management of these linkages can be improved, resulting in smaller transportation costs, that is, shipments between hubs benefit from a discount factor. It is these scale economies along with the reduced number of linkages (compared to a fully interconnected network) that make interesting these hub-and-spoke systems.

According to the particular characteristics considered, there are several kinds of hub location problems: the number of hubs may or may not be established beforehand, the node allocation may be either single or multiple, there may be constraints on the capacities… A general review of the different problems can be found in the classical paper [2] or in the more recent and exhaustive survey [3].

This paper deals with the Uncapacitated Multiple Allocation Hub Location Problem (UMAHLP), where the following assumptions are considered:

- the number of hubs is undetermined beforehand, but there are fixed costs for establishing hubs on the nodes,

- the node allocation is multiple, that is, each origin-destination pair is allocated to the most suitable pair of hubs,
- and there are no capacities.

The UMAHLP was first formulated in [2]. A more robust formulation was stated in [14], which was later improved in [9] and [4], where the formulation that will be used in this paper is. Although all these formulations are four-indexed, that is, they use $O(n^4)$ variables, three-indexed formulations (that is, with $O(n^3)$ variables) also exist (see e.g. [1]).

Dual-ascent techniques for location problems were first used in [6], but it was in [10] where they were first applied to the UMAHLP. A superior performance to its algorithm was shown in [11]. However, the best results for this problem to date have been obtained in [1]. In this paper it is shown how the new algorithm presented is much more efficient than this latter one, solving much larger problems in much less time.

In Section 2 the notation is introduced and the dual problem is formulated. The heuristic core tool is developed in Section 3 and applied to the preprocessed dual problem (a simple preprocessing analysis halves the number of constraints, which comes in handy for the algorithm). This heuristic tool is based on a dual-ascent technique (reinforced with specifical subrou- titines such as the use of multiple horizons and slackness rectification) and it essentially provides a lower bound for the problem. A primal feasible solution is obtained from the dual heuristic solution by using slackness constraints. It is interesting to remark that this heuristic does not need any external LP-solver. Its implementation in a branch-and-bound framework is explained in Section 4 and the computational experience carried out with the AP and CAB data sets is included in Section 5. Finally, some conclusions are given in Section 6.
2. Dual formulation for the UMAHLP

A set of $n$ nodes (postal codes, computers), $N = \{1, \ldots, n\}$, is considered. There are amounts of flow $w_{ij} \geq 0$ (letters, information packages) which must be sent from the $i^{th}$ location to the $j^{th}$ one, $1 \leq i, j \leq n$. The cost for establishing a hub on node $k$ is $f_k \geq 0, 1 \leq k \leq n$.

Since every flow must be routed via the hubs using at least one and at most two, these are the possibilities:

i) if neither the origin nor the destination are hubs, the flow goes through one or two hubs;

ii) if either the origin or the destination (but not both of them) is a hub, then the flow may be sent directly or it may go through one intermediate hub;

iii) if both the origin and the destination are hubs, the flow is sent directly.

The transportation cost per unit of flow in arc $(i, j)$ is $c_{ij} \geq 0$. Costs $c_{ij}$ are distance-based costs, that is, these values are symmetric and satisfy the triangle inequality. The flows are sent in three steps: collection (from origin to hub), transfer (from hub to hub) and distribution (from hub to node). Each step has a cost factor $\chi, \alpha, \delta \geq 0$, respectively, and, because of inter-hub transport efficiencies, it is supposed to be $\chi, \delta > \alpha$. Therefore, the cost of route $i \rightarrow k \rightarrow m \rightarrow j$, where both $k$ and $m$ are hubs, is

$$C_{ijkm} = w_{ij} (\chi c_{ik} + \alpha c_{km} + \delta c_{mj}).$$

In initial works (such as [13]) only the discount factor $\alpha$ is considered. In later works (such as [2]) the other parameters are used.
The aim of the problem is to find the optimal location-allocation solution which minimizes the total cost.

The following variables are defined to formulate the problem:

\[ x_{ijkm} = \text{fraction of } w_{ij} \text{ which is sent from origin } i \text{ to destination } j \]
\[ \text{via hubs } k \text{ and } m \text{ (in this same order)}, \]

\[ y_k = \begin{cases} 1 & \text{if node } k \text{ is a hub,} \\ 0 & \text{otherwise,} \end{cases} \]

with \( 1 \leq i, j, k, m \leq n \)

Particularly, variables \( x_{ijkk} \) represent shipments via one single intermediate hub \( k \).

From now on, unless otherwise explicitly stated, all the indices will be referred from 1 to \( n \). Thus, using the previous decision variables along with the formulation appeared in [4], the problem to be solved is:

\[
(\text{P}) \begin{cases}
\text{Min} & \sum_{i,j,k,m} C_{ijkm}x_{ijkm} + \sum_k f_k y_k \\
\text{s.t.} & \sum_{k,m} x_{ijkm} = 1 \quad \forall i, j, \\
& \sum_m x_{ijkm} + \sum_{m,m \neq k} x_{ijmk} - y_k \leq 0 \quad \forall i, j, k, \\
& y_k \in \{0, 1\} \quad \forall k, \\
& x_{ijkm} \geq 0 \quad \forall i, j, k, m.
\end{cases}
\]

Constraints (2) specify that all the flow is sent between every pair of nodes, while constraints (3) ensure that flow is only sent via open hubs. Remark that the upper bounds \( x_{ijkm} \leq 1 \) have been omitted in (5) because they are implicitly given by constraints (2). Observe as well that, since there are no capacities, then variables \( x_{ijkm} \) are binary in the optimal solution, taking the value 1 only when representing the cheapest route from \( i \) to \( j \).
Now the linear relaxation problem of \((P)\) is considered:

\[
(LP) \begin{cases}
\text{Min} & (1) \\
\text{s.t.} & (2), (3), (5), \\
y_k \geq 0 & \forall k.
\end{cases}
\]

Notice that variables \(y_k\) take values between 0 and 1 in the optimal solutions because variables \(x_{ijkm}\) are 0-1 in the optimal points, values \(C_{ijkm}\) and \(f_k\) are all positive and the problem is a minimization one. This is the reason why upper bounds \(y_k \leq 1\) have been dropped out.

Dual variables \(u_{ij}\) are next associated to constraints (2) and variables \(\tilde{v}_{ijk}\) to (3), obtaining the dual problem:

\[
(D_0) \begin{cases}
\text{Max} & \sum_{i,j} u_{ij} \\
\text{s.t.} & u_{ij} + \tilde{v}_{ijk} \leq C_{ijkk} & \forall i, j, k, \\
 & u_{ij} + \tilde{v}_{ijk} + \tilde{v}_{ijm} \leq C_{ijkm} & \forall i, j, k, m, k \neq m, \\
 & - \sum_{i,j} \tilde{v}_{ijk} \leq f_k & \forall k, \\
 & u_{ij} \text{ free} & \forall i, j, \\
 & \tilde{v}_{ijk} \leq 0 & \forall i, j, k.
\end{cases}
\]

Observe that only variables \(u_{ij}\) appear in the objective function. As a result, once the change \(v_{ijk} = -\tilde{v}_{ijk}\) has been made, the following rewriting of \(D_0\) points to a dual ascent method as a potential method to solve it:

\[
(D) \begin{cases}
\text{Max} & \sum_{i,j} u_{ij} \\
\text{s.t.} & u_{ij} \leq C_{ijkk} + v_{ijk} & \forall i, j, k, \\
 & u_{ij} \leq C_{ijkm} + v_{ijk} + v_{ijm} & \forall i, j, k, m, k \neq m, \\
 & \sum_{i,j} v_{ijk} \leq f_k & \forall k, \\
 & u_{ij} \geq 0 & \forall i, j, \\
 & v_{ijk} \geq 0 & \forall i, j, k.
\end{cases}
\]
It is clear that variables \( u_{ij} \) will be positive in any optimal dual solution because values \( C_{ijkm} \) are all positive. Therefore, \((D_0)\) and \((D)\) have the same optimal values.

3. The dual heuristic algorithm

A dual ascent algorithm will be used to solve problem \((D)\). The aim is to increase variables \( u_{ij} \) as much as possible and as quick as possible.

The initial status is:

\[
  u_{ij} = \min \{ C_{ijkm} \}_{k,m} \forall i, j, \quad v_{ijk} = 0 \forall i, j, k.
\]

3.1. Previous considerations

First, an initial preprocessing is carried out. Additionally, some further notation is explained.

Initial preprocessing

As a rule, it is very efficient to do a preprocessing analysis prior to solving any problem -and the one considered here is no exception.

To begin with, the number of constraints of \((D)\) can be nearly halved by taking

\[
  C_{ijkm}^c = \min \{ C_{ijkm}, C_{ijmk} \} \forall i, j, k, m, \quad k < m,
\]

which results in reducing constraint family \((6)\) to

\[
  u_{ij} \leq C_{ijkm}^c + v_{ijk} + v_{ijm} \forall i, j, k, m, \quad k < m.
\]

Additionally, let us define \( C_{ijkk}^c = C_{ijkk} \forall i, j, k. \)

If \( R(i, j) \) denotes all the constraints of \((D)\) involving variable \( u_{ij} \) (after this preprocessing), then \(|R(i, j)|\) has been decreased from \( n^2 \) to \( n(n + 1)/2 \).
Two additional preprocessing rules, whose proofs can be found in [1], are considered:

**PR 1**
*If \((x^*, y^*)\) is an optimal solution for \((P)\), then*

\[ x_{ikm}^* = 0 \quad \forall i, k, m, \ k \neq m. \]

**PR 2**
*Let \((x^*, y^*)\) be an optimal solution for \((P)\). If \(i, j, k, m \in N, k \neq m\), and*

\[ C_{ijkm} > \min \{C_{ijkk}, C_{ijmm}\}, \]

*then*

\[ x_{ijkm}^* = 0. \]

*Equivalently:*

- *if \(i, k, m \in N\) and \(\chi c_{ik} + \alpha c_{km} > \chi c_{im}\), then*

\[ x_{ijkm}^* = 0 \quad \forall j; \]

- *if \(j, k, m \in N\) and \(\alpha c_{km} + \delta c_{mj} > \delta c_{kj}\), then*

\[ x_{ijkm}^* = 0 \quad \forall i. \]

**Further notation**

The following notation will be used throughout the text.

- *\(PL\) is the pair list: it contains all the variables \(u_{ij}\) which can still be increased. Initially*

\[ PL = \{(i, j) : i, j \in N\}. \]
• $EL$ is the erased list: it contains all the variables $u_{ij}$ which can no longer be increased. Initially $EL = \emptyset$.

• $r_k$ is the $k^{th}$ remaining resource. Initially $r_k = f_k \forall k$.

• The value of quartern $(i, j, k, m)$ is

$$val(i, j, k, m) = \begin{cases} 
C_{ijkm}^k + v_{ijk}, & k = m, \\
C_{ijkm}^k + v_{ijk} + v_{ijm}, & k \neq m. 
\end{cases}$$

• Throughout the heuristic process, values $\{val(i, j, k, m)\}_{(k,m) \in R(i,j)}$ are considered in increasing order for each pair $(i, j) \in PL$:

$$val^{(1)}_{ij} < val^{(2)}_{ij} < \ldots < val^{(d)}_{ij} < val^{(d+1)}_{ij} := M,$$

where $M$ is a value much larger than the data of the problem and

$$\begin{align*}
val^{(1)}_{ij} &= \min \{val(i, j, k, m)\}_{(k,m) \in R(i,j)}, \\
val^{(t+1)}_{ij} &= \min \{val(i, j, k, m) > val^{(1)}_{ij}, \ldots, val^{(t)}_{ij}\}_{(k,m) \in R(i,j)}, \ t \geq 1.
\end{align*}$$

• The $t^{th}$ level of pair $(i, j)$ is defined as

$$L_t(i, j) = \{(k, m) \in R(i, j) : val(i, j, k, m) = val^{(t)}_{ij}\}$$

and $val^{(t)}_{ij}$ is the value of the $t^{th}$ level of pair $(i, j)$, that is, every pair belonging to the $t^{th}$ level has value $val^{(t)}_{ij}$.

• The $k^{th}$ increment of pair $(i, j)$ is

$$\Delta_k(i, j) = val^{(k+1)}_{ij} - val^{(k)}_{ij},$$

the amount required for ascending from $k^{th}$ level to the $(k + 1)^{th}$ one.
3.2. A detailed iteration

Next, a general iteration of the algorithm is fully described.

**Pair selection**

A pair \((\bar{i}, \bar{j}) \in PL\) is chosen according to the following criteria:

i) Choose \((\bar{i}, \bar{j})\) such as

\[
|L_1(\bar{i}, \bar{j})| = \min \{|L_1(i, j)|\}_{(i,j) \in PL}.
\]

ii) If there is a tie, then it is additionally required

\[
\Delta_1(\bar{i}, \bar{j})) = \max \{\Delta_1(i, j)\}_{(i,j)}.
\]

iii) If the tie still persists, it is also required

\[
(\bar{i}, \bar{j}) = \text{lex min}\{(i, j)\},
\]

where lex min means the lexicographical minimum.

Hence the pair \((\bar{i}, \bar{j})\) with the minimum cardinal first level and the maximum potential increment is chosen.

**The basic covering set problem**

Now we consider the set

\[
L_1(\bar{i}, \bar{j}) = \{(k_1, m_1), \ldots, (k_s, m_s)\}
\]

and \(\Delta_1(\bar{i}, \bar{j})\) the gap between \(val_{ij}^{(1)}\) and \(val_{ij}^{(2)}\).

Let

\[
Q = \{k_1, m_1, k_2, m_2, \ldots, k_s, m_s\}
\]
be the hubs of the first level without possible multiplicities, that is, repeated elements are only considered once. A subset \( \hat{Q} \) of \( Q \) must be chosen in such a way that by doing

\[
v_{ijk} = v_{ijk} + \Delta_1(\bar{i}, \bar{j}) \quad \forall k \in \hat{Q},
\]
\[
r_k = r_k - \Delta_1(\bar{i}, \bar{j}) \geq 0 \quad \forall k \in \hat{Q},
\]

the value of the first level reaches the value of the second one. Should this latter value be impossible to reach, the aim will be to approach to it as much as possible.

There are four possible cases depending on the kind of increment:

1) *(Unrestricted) full increment:* next level is reached.
   
   An increment of \( \Delta_1(\bar{i}, \bar{j}) \) units can be done and every pair of \( Q \) is eligible.
   
   \( \land \Delta_1(\bar{i}, \bar{j}) \leq r_k \forall k \in Q. \)

2) *Restricted full increment:* next level is reached.

   A full increment is possible, although not all the hubs are eligible for doing it.
   
   \( \land \Delta_1(\bar{i}, \bar{j}) \leq \max \{r_k, r_m\}, 1 \leq t \leq s. \)
   \( \lor \exists t \in \{1, \ldots, s\} : \Delta_1(\bar{i}, \bar{j}) > \min \{r_k, r_m\}. \)

3) *Partial increment:* next level is not reached.

   It is possible to do an increment, although one of less than \( \Delta_1(\bar{i}, \bar{j}) \) units.
   
   \( \land \Delta_1(\bar{i}, \bar{j}) > \min \{\max \{r_k, r_m\}\}_{t=1}^s > 0. \)

4) Null increment: next level is not reached.
This is a blocked pair case. Since the real increment is null, pair \((i, j)\) is
removed from \(PL\) and aggregated to \(EL\). A new pair \((\bar{i}, \bar{j})\) is selected.
If \(PL = \emptyset\), then the heuristic iterates no longer.

\[ \diamond \exists \bar{t} \in \{1, \ldots, s\} : \quad r_{k_t} = r_{m_t} = 0. \]

Although by space reasons only the first (and the most usual) case will
be explained, variants from the standard solution method here given can be
easily derived to solve the other three cases.

As the purpose is finding a feasible subset \(\hat{Q}\) which needs as little resource
as possible, the problem is

\[
\begin{align*}
\text{(CP)} \quad \text{Min} & \quad \sum_{a \in Q} g_a \\
\text{s.t.} & \quad g_{k_1} + g_{m_1} \geq 1, \quad \quad (CPC_1) \\
& \quad \vdots \quad \vdots \\
& \quad g_{k_s} + g_{m_s} \geq 1, \quad \quad (CPC_s) \\
& \quad g_a \in \{0,1\}, \quad a \in Q.
\end{align*}
\]

Observe that a first heuristic decision has already been taken when dis-
cretizing variables \(g_a\): in every constraint, either one hub or the other one
(even both of them perhaps) is chosen, being mixed solutions given up. Part-
icularly, this consideration draws \((CP)\) near to a classical set covering prob-
lem.

Notice that a hub \(k\) will always be chosen if there is a pair with both
components identical, that is, \((k_t, m_t) = (k, k)\). This is the reason why
constraints of the form \(g_k + g_k \geq 1\) can be used instead of \(g_k \geq 1\) and hub \(k\)
can be computed indistinctly with either one or two multiplicities in the rule
described below (although this latter decision, which is the criterium here
chosen, is better because it reduces the number of ties).
Algorithm $CP$:

1. Let $\mu(a)$, $a = 1, \ldots, s$, be the multiplicity of $g_a$ in the constraints of problem $(CP)$ –that is, how many times $g_a$ appears– and let $\hat{Q} = \emptyset$.

2. Start with constraint $(CP_{C1})$: if $\mu(k_1) \geq \mu(m_1)$, then $\hat{Q} = \{k_1\}$; otherwise, $\hat{Q} = \{m_1\}$.

3. Consider now the other constraints (one by one and following the ascending order $t = 2, \ldots, s$):
   - if $\mu(k_t) + \mu(m_t) \geq 1$, then constraint $(CP_{Ct})$ is already covered and no additional hub needs to be chosen for this constraint;
   - otherwise, a new hub is chosen and aggregated to $\hat{Q}$ according to the same rule used for $(CP_{C1})$.

Example 1

If

$$L_1(\bar{i}, \bar{j}) = \{(1, 2), (1, 1), (2, 3), (1, 3), (3, 4), (1, 5)\},$$

then

$$\mu(1) = 5, \mu(2) = 2, \mu(3) = 3, \mu(4) = 1, \mu(5) = 1;$$

therefore

$$\hat{Q} = \{1, 3\}.$$

Now let us define $\Delta = \Delta'_1$, where $\Delta'_1 \leq \Delta_1(\bar{i}, \bar{j})$ is the real increment done, and variables are updated:

$$v_{ijk} = v_{ijk} + \Delta \quad \forall k \in \hat{Q},$$

$$r_k = r_k - \Delta \quad \forall k \in \hat{Q}.$$
Observe that it might be $\Delta^f_1 < \Delta_1(\tilde{i}, \tilde{j})$ (in case of non-full increment, cases 3 and 4).

However, this update operation will not be done yet. The reason is next explained.

**Horizon use**

First of all, an example is given which will motivate the ideas explained later in this section.

**Example 2**

Consider this full increment case, where $r_3 \geq 50$:

<table>
<thead>
<tr>
<th>$k_t$</th>
<th>$m_t$</th>
<th>$val(\tilde{i}, \tilde{j}, k_t, m_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>60</td>
</tr>
</tbody>
</table>

We have $\Delta_1(\tilde{i}, \tilde{j}) = 8$; thus, pair $(\tilde{i}, \tilde{j})$ has been selected, with $|L(\tilde{i}, \tilde{j})| = 1$ and 8 the highest increment among all the first levels with length one. Besides, $\hat{Q} = \{3\}$.

According to what have already been explained, once the iteration is made and variables have been updated, the situation becomes

<table>
<thead>
<tr>
<th>$k_t$</th>
<th>$m_t$</th>
<th>$val(\tilde{i}, \tilde{j}, k_t, m_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>60</td>
</tr>
</tbody>
</table>

In next iteration pair $(\tilde{i}, \tilde{j})$ will be chosen again and the increment will be also the same (since now $r_3 \geq 42$).
This behavior will keep on going on until something “new” happens: a full increment is no longer possible, the second level value overtakes the third one... The important fact is that useless iterations (meaning by useless that they are identical to the previous ones) are being done: even six in this example.

The concept of horizon is introduced to deal with this situation. The \( k^{th} \) horizon is defined as the \((k+1)^{th}\) level, \( k \geq 1 \). Therefore, in the previous section, the increment of the first horizon has been explained.

The basic idea is to avoid useless iterations by considering simultaneously (that is, in the same iteration) multiple horizons. To begin with, values of \( \{\bar{v}_{ij}k\} \) and \( \{r_k\} \) are not updated until the horizon process is finished.

If a full increment cannot be done in the first horizon, the horizon process stops. Otherwise, \( \Delta_2(\bar{i}, \bar{j}) \) and a similar problem to \((CP)\) (but with the constraints associated with \( L_2(\bar{i}, \bar{j}) \)) are considered.

This new problem is solved in the same way that \((CP)\) save for the following particularities:

- **Overlapping restriction**: only the hubs of \( L_1(\bar{i}, \bar{j}) \) are eligible, that is, \( \mu(k) \) is fixed to zero for all \( k \notin L_1(\bar{i}, \bar{j}) \). Moreover, observe that no mention of updating values \( \mu(k) \) with the new constraints has been done.

- **Pair pseudorectification**: if \( (k, m) \in L_1(\bar{i}, \bar{j}) \) and one of these two hubs belongs to \( \hat{Q} \), but not both of them, then the non-belonging one is eligible in this new covering problem.

The first condition keeps the number of possible choices controlled, preventing the expanded problem from becoming extremely difficult. The second one is very handy in order to solve problems which, although not solved with
the current set $\hat{Q}$, can be solved with a “near” one, that is, by using hubs from the initial covering problem.

Of course, these new covering problems can be given in any of the four variants before stated.

Once the second horizon is solved, we do $\Delta = \Delta + \Delta f^2$, where $\Delta f^2 \leq \Delta^2(\bar{i}, \bar{j})$ is the real increment for this second horizon.

If $\Delta f^2 = \Delta^2(\bar{i}, \bar{j})$, then we try to do the increment corresponding to the third horizon. This process is repeated until a horizon is reached where no full increment can be done (values $r_k$ are limited) or a constraint appears where none of the two hubs can be chosen (because $\mu$ is zero for both of them).

When this horizon process ends, values are finally updated:

\begin{align*}
v_{ijk} &= v_{ijk} + \Delta \quad \forall k \in \hat{Q}, \\
r_k &= r_k - \Delta \quad \forall k \in \hat{Q}.
\end{align*}

The following example illustrates this process.

**Example 3**

*Forget again any limitation due to $r_k$ values and consider the following situation:*

<table>
<thead>
<tr>
<th>$k_t$</th>
<th>$m_t$</th>
<th>$val(\bar{i}, \bar{j}, k_t, m_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>25</td>
</tr>
</tbody>
</table>

*According to what has been exposed:*

$\hat{Q} = \{2, 3\}$,
\[ \Delta = \Delta_1' + \Delta_2' + \Delta_3' = 5 + 3 + 1 = 9, \]

\begin{align*}
&k_t \quad m_t \quad \text{val}(\bar{i}, \bar{j}, k_t, m_t) \\
&2 \quad 3 \quad 28 = 10 + 9 + 9 \\
&2 \quad 4 \quad 24 = 15 + 9 \\
&3 \quad 8 \quad 27 = 18 + 9 \\
&4 \quad 9 \quad 19 \\
&1 \quad 5 \quad 25
\end{align*}

Three simple iterations have been done in just one single step.

**Slackness rectification**

The horizon process can be improved still by adding a debugging phase to treat situations like the one shown in the following example (which, by the way, are very common).

The name *slackness rectification* is due to the fact that in this phase slackness values are tightened (reduced), but without reducing the objective value.

**Example 4**

\begin{align*}
&k_t \quad m_t \quad C_{ijkt} \quad v_{ijk} \quad v_{iijmt} \quad \text{val}(\bar{i}, \bar{j}, k_t, m_t) \\
&1 \quad 2 \quad 10 \quad 4 \quad 2 \quad 16 \\
&2 \quad 2 \quad 14 \quad 2 \quad - \quad 16 \\
&2 \quad 3 \quad 12 \quad 2 \quad 4 \quad 18 \\
&1 \quad 4 \quad 11 \quad 4 \quad 4 \quad 19 \\
&3 \quad 4 \quad 12 \quad 4 \quad 4 \quad 20 \\
&2 \quad 4 \quad 20 \quad 2 \quad 4 \quad 26
\end{align*}

We observe that some \( v_{ijk} \) may perhaps be reduced without affecting the optimality of the situation, that is, without reducing the current heuristic
The resources so recovered may be used to increase the objective function even more.

Given a pair \((i, j)\), variable \(u_{ij}\) is bounded by the lowest value of

\[
\min \left\{ C_{ijkk}^< + v_{ijk} \right\}_k, \quad \min \left\{ C_{ijkm}^< + v_{ijk} + v_{ijm} \right\}_{k<m}.
\]

Hence, it would be interesting to lower the right hand sides of the non-binding constraints as much as possible, because it would increase \(r_k\) values without changing the current dual objective value. Therefore, the basic idea is reducing variables \(v_{ijk}\) as much as possible and adding the retrieved resources to variables \(r_k\).

The current feasible situation comes given by the constraints

\[
\begin{align*}
C_{ijkk}^< + v_{ijk} & \geq val_{ij}^{(1)}, \quad (k, k) \in R(\bar{i}, \bar{j}), \\
C_{ijkm}^< + v_{ijk} + v_{ijm} & \geq val_{ij}^{(1)}, \quad (k, m) \in R(\bar{i}, \bar{j}), \ k < m,
\end{align*}
\]

The (multiobjective) problem to solve is

\[
\begin{align*}
\text{Max } \{s_1, \ldots, s_n\} \\
\text{s.t. } C_{ijkk}^< (v_{ijk} - s_k) & \geq val_{ij}^{(1)}, \quad (k, k) \in R(\bar{i}, \bar{j}), \\
C_{ijkm}^< (v_{ijk} - s_k) + (v_{ijm} - s_m) & \geq val_{ij}^{(1)}, \quad (k, m) \in R(\bar{i}, \bar{j}), \ k < m,
\end{align*}
\]

\[0 \leq s_k \leq v_{ijk}, \quad 1 \leq k \leq n.
\]

Obviously, variables \(\{s_k\}\) are required to be positive and to be upperly bounded by \(\{v_{ijk}\}\).

Complementing \(t_k = v_{ijk} - s_k\), the reduction problem is rewritten as:

\[
\begin{align*}
\text{Min } \{t_1, \ldots, t_n\} \\
\text{s.t. } t_k & \geq val_{ij}^{(1)} - C_{ijkk}^<, \quad (k, k) \in R(\bar{i}, \bar{j}), \\
t_k + t_m & \geq val_{ij}^{(1)} - C_{ijkm}^<, \quad (k, m) \in R(\bar{i}, \bar{j}), \ k < m,
\end{align*}
\]

\[0 \leq t_k \leq v_{ijk}, \quad 1 \leq k \leq n.
\]

The heuristic process designed to solve \((RP)\) is next exposed:
1. Trivial preprocessing:

1.1 Variable trivial elimination

Let $\mathcal{K} = \{1, \ldots, n\}$.

For each $k \in \mathcal{K}$, if $v_{ijk} = 0$, then $t^*_k$ is fixed to zero and $k$ is removed from $\mathcal{K}$.

1.2 Constraint trivial elimination

If $val_{ij}^{(1)} \cdot C_{ijkm}^< < 0$, the associated constraint is removed from $(RP)$.

2. Initialization:

For each pair $(k, m) \in \mathcal{K} \times \mathcal{K}$, $1 \leq k \leq m \leq n$, if $val_{ij}^{(1)} = val(i, j, k, m)$, then $t^*_k$ and $t^*_m$ are fixed to $v_{ijk}$ and $v_{ijm}$, respectively, and $k$ and $m$ are removed from $\mathcal{K}$ because neither of these variables can be reduced.

Finally, all the remaining variables of $\mathcal{K}$ are initially set to zero:

$$t^*_k = 0 \ \forall k \in \mathcal{K}.$$ 

Nevertheless, this first solution may be unfeasible.

3. Feasibility search

After these previous steps, the problem is:

$$\begin{align*}
\text{Min} \ \{t_k\}_{k \in \mathcal{K}} \\
\text{s.t.} \quad & t_k \geq val_{ij}^{(1)} - C_{ijkk}^<, \quad k \in \mathcal{K}, \\
& t_k \geq val_{ij}^{(1)} - C_{ijkm}^< - v_{ijm}, \quad k \in \mathcal{K}, \ m \notin \mathcal{K}, \ k < m, \\
& t_k \geq val_{ij}^{(1)} - C_{ijkm}^< - v_{ijk}, \quad m \in \mathcal{K}, \ k \notin \mathcal{K}, \ k < m, \\
& t_k + t_m \geq val_{ij}^{(1)} - C_{ijkm}^<, \quad (k, m) \in \mathcal{K} \times \mathcal{K}, \ k < m, \quad (7) \\
& 0 \leq t_k \leq v_{ijk}, \quad k \in \mathcal{K}.
\end{align*}$$

Again, any constraint with a negative right hand side is removed.
3.1 Subproblem:

We solve the subproblem obtained by removing constraints (7) from \((RP)\). The solution \(\{t^*_k\}_{k \in K}\) for this problem is immediate.

3.2 Pair adjustment:

Let \(\mathcal{L} = \emptyset\).

For each pair \((k, m) \in K \times K, k < m\), if

\[
t^*_k + t^*_m \geq \text{val}^{(1)}_{ij} - C_{ijkm},
\]

then the constraint is satisfied. Otherwise, the solution is infeasible for that constraint and pair \((k, m)\) is aggregated to \(\mathcal{L}\).

Therefore, problem \((RP)\) has been reduced to

\[
\begin{aligned}
\text{Min} & \quad \{t_k\}_{k \in K} \\
\text{s.t.} & \quad t_k + t_m \geq \text{val}^{(1)}_{ij} - C_{ijkm}, \quad (k, m) \in \mathcal{L}, \\
& \quad t_k \geq t^*_k, \quad k \in K.
\end{aligned}
\]

Starting with the first constraint, we proceed as follows: let

\[
\lambda_1 = v_{ijk} - t^*_k, \quad \lambda_2 = \text{val}^{(1)}_{ij} - C_{ijkm} - t^*_k - t^*_m.
\]

If \(\lambda_2 \leq \lambda_1\), then

\[
t^*_k = t^*_k + \lambda_2.
\]

Otherwise,

\[
t^*_k = v_{ijk}, \quad t^*_m = t^*_m + \lambda_2 - \lambda_1.
\]

That is, we try to add all the difference to \(t_k\). If this leads to an unfeasible solution (due to exceeding its upper bound), then \(t_k\) achieves this upper value and the rest of the difference is added to \(t_m\).

Since variables \(t_k\) are updated after each step, there may be newly satisfied constraints which will not need to be analyzed.
4. **Ending:**

Finally,

\[ r_k = r_k + v_{ijk} - t^*_k, \quad v_{ijk} = t^*_k \quad 1 \leq k \leq n. \]

If \( r_k \) was null prior to this analysis and it is now positive, all the pairs of \( EL \) which were deleted because of the \( r_k \) being empty are recovered: they are aggregated to \( PL \) and removed from \( EL \).

A last remark on the preprocessing phase: given a quartern \((i, j, k, m)\), with \( k \neq m \), as

\[ C_{ijkm} < \min \{C_{ijkk}, C_{ijmm}\} \]

because of the initial preprocessing for \((D)\), then constraint

\[ C_{ijkm} + t_k + t_m \geq \text{val}_{ij}^{(1)} \]

is not redundant with

\[ C_{ijkk} + t_k \geq \text{val}_{ij}^{(1)}, \]

\[ C_{ijmm} + t_m \geq \text{val}_{ij}^{(1)}. \]

This is the reason why this case of redundancy has not been considered in the preprocessing phase for solving \((RP)\).

**Example 5**

*Take up again the situation exposed in Example 4 and solve the associated problem.*

\( K = \{1, 2, 3, 4\} \).

**Step 1:** Constraint \((2, 4)\) is deleted because

\[ 16 = \text{val}_{ij}^{(1)} < C_{ij24} = 20. \]
Step 2: Fix $t_1^* = 4$, $t_2^* = 2$.

$\mathcal{K} = \{3, 4\}$.

Set $t_3^* = t_4^* = 0$.

Step 3: The problem is

$$\begin{align*}
\text{Min} & \quad \{t_3, t_4\} \\
\text{s.t.} & \quad t_3 \geq 2, \\
& \quad t_4 \geq 1, \\
& \quad t_3 + t_4 \geq 4, \\
& \quad 0 \leq t_3, t_4 \leq 4.
\end{align*}$$

Omitting constraint $t_3 + t_4 \geq 4$, the optimal solution is

$t_3^* = 2$, $t_4^* = 1$.

Since $t_3 + t_4 < 4$, a final rectification is needed:

$t_3^* = 3$, $t_4^* = 1$.

Thus, $(t_1^*, t_2^*, t_3^*, t_4^*) = (4, 2, 3, 1)$ and the final status is

<table>
<thead>
<tr>
<th>$k_t$</th>
<th>$m_t$</th>
<th>$val(\bar{i}, \bar{j}, k_t, m_t)$</th>
<th>old value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>23</td>
<td>26</td>
</tr>
</tbody>
</table>

The objective value of the heuristic does not change and the situation is now more advantageous than before because $r_3$ and $r_4$ have been increased by 1 and 3 units, respectively.
3.3. Stop criterium

The heuristic algorithm stops if:

- either $PL = \emptyset$ (every pair is blocked)
- or $r_k = 0 \ \forall k \in N$ (exhausted resources).

It must be remarked that it is not possible that the algorithm ends with $PL = \emptyset$ and $r_k > 0 \ \forall k \in N$ (whenever all values $r_k$ are strictly positive, it is possible to keep on increasing the dual function). This observation will be important when described below the process to obtain a primal feasible solution from the dual heuristic solution.

To sum up, the general scheme of the described heuristic is the following:

**Heuristic tool overview:**

1. $PL = \{(i, j) : i, j \in N\}$.
2. While $PL \neq \emptyset$, do:
   2.1 Choose the best pair $(\bar{i}, \bar{j}) \in PL$.
   2.2 Solve the related problem $(CP)$ with the aid of the multiple horizon strategy.
   2.3 Apply the slackness rectification process.
   2.4 If $r_k = 0 \ \forall k \in N$, stop.

4. The branch-and-bound algorithm

The dual heuristic described in the previous section will be the basic tool in a classical branch-and-bound algorithm.
Variable selection

The branching variable is chosen in the following way:

Let

\[ N = \{1, \ldots, n\}, \]
\[ F_0 = \{k \in N : y_k \text{ is fixed to } 0\}, \]
\[ F_1 = \{k \in N : y_k \text{ is fixed to } 1\}, \]
\[ F_2 = N \setminus (F_0 \cup F_1), \]
\[ F_2^0 = \{k \in F_2 : r_k = 0\}, \]
\[ F_2^+ = F_2 \setminus F_2^0. \]

From now on, we consider a node from \( F_0 \cup F_1 \) and look for a variable \( \bar{k} \) to be fixed such as:

i) \( f_{\bar{k}} = \min \{f_k\}_{k \in F_2^0}. \)

ii) If \( F_2^0 = \emptyset \), then

\[ \frac{r_{\bar{k}}}{f_{\bar{k}}} = \min \left\{ \frac{r_k}{f_k} \right\}_{k \in F_2^+}. \]

Any tie is undone by choosing the lexicographical minimum.

From dual to primal

A primal feasible solution \((\hat{x}, \hat{y})\) is obtained from the dual solution by considering the following slackness constraint:

\[ y_k \left( \sum_{i,j} v_{ijk} - f_k \right) = 0 \quad \forall k. \]

The criterium used is:

- if \( r_k = 0 \), then \( \hat{y}_k \) is set to one,

- if \( r_k = 1 \), then \( \hat{y}_k \) is set to zero; notice that this condition is not a consequence from the slackness constraint but an arbitrary rule;
- $\hat{x}_{ijkm} = 1$ if

$$C_{ijkm} = \min \left\{ C_{ijkm} : \hat{y}_k = \hat{y}_m = 1 \right\}_{k,m}.$$ 

Observe that at least one hub will be open, since it was proved before that it is not possible for the heuristic to stop with $r_k > 0 \forall k$.

Once this initial heuristic primal solution is determined, a simple post-optimization process is carried out: any surplus hub because its being open is not useful is closed. Providing that two or more hubs have been opened, open hubs are checked from the one with the highest $f_k$ to the one with the lowest opening cost; if the primal value resulting from closing this hub is lower than the one from keeping it open, then this hub is closed and any route which used this hub is adapted to the remaining hubs.

The objective value of this primal solution is an upper bound for the node and the value of the dual heuristic solution is a lower bound.

**Node selection**

The node with the lowest upper bound is the one chosen for branching and the correspondent upper and lower bounds for the two son nodes are obtained.

**4.1. Node treatment**

The following considerations provide more efficiency to the algorithm.

**Extra preprocessing**

If the current node is not the root, additional preprocessing tests can be checked ([1]):
PR 3
If \( y_k = y_m = 0, k, m \in N \), then
\[
x_{ijkm} = 0 \forall i, j.
\]

PR 4
If \( y_i = 1, i \in N \), then
\[
x_{ijkm} = 0 \forall j, k, m, k \neq i;
\]
\[
x_{jikm} = 0 \forall j, k, m, m \neq i.
\]

PR 5
If \( y_k = 1 \) and \( \chi c_{ik} + \alpha c_{km} < \chi c_{im}, i, k, m \in N \), then
\[
x_{ijmh} = 0 \forall j, h \in N.
\]
If \( y_m = 1 \) and \( \alpha c_{km} + \delta c_{mj} < \delta c_{kj}, j, k, m \in N \), then
\[
x_{ijhk} = 0 \forall i, h \in N.
\]

PR 6
If \( y_k = 1 \) and \( C_{ijk1k2} > C_{ijkk}, i, j, k, k_1, k_2 \in N \) then
\[
x_{ijk1k2} = 0.
\]

Every variable \( x_{ijkm} \) fixed to zero means a redundant constraint for the dual problem (and thus not included in the subproblem associated to the node).

Dual adjustment

Instead of solving the dual problem from scratch, if the current node is not the root, it is much more efficient to work in the following way:
1) Once preprocessed the node, initialize variables $v_{ijk}$ with the values they have at their parent node.

2) Apply the slackness rectification process (it is now applicable because some constraints may have been removed in the preprocessing of this node).

3) Solve the dual problem.

5. Computational results

The branch-and-bound algorithm was implemented in a 1500 MHz Pentium with 2 GB RAM and 1.5 GB of swap memory. The code was written using C++ under Linux. The heuristic, Method 1, was compared with both the comercial LP-solver Xpress-MP ([5]), Method 2, and the formulation used in [1], Method 3, which was the best exact solving method for the UMAHLP up to date. The CPU times of this third method shown in Table 1 are the ones obtained in that work using a Digital Personal workstation with a 500 MHz alpha chip; despite the different machines employed, the effectiveness of Method 1 is clear. As testing data sets, the AP (see [7]) and the CAB data sets (see [12]) have been used.

The AP data set (Australian Post) corresponds to 200 real-world postcode districts from the Australian Post. Smaller problems are derived from it by aggregating the data. Fixed costs (loose -L- and tight -T) are added to these data according to [8]. The factors $\chi = 3$, $\alpha = 0.75$ and $\delta = 2$ are also given along with these costs. These files and the code to generate the subproblems can be obtained from OR-Library in http://mscmga.ms.ic.ac.uk/info.html. Up to date, instances with up to 50 nodes and using a three-indexed formulation (essentially, 125000 variables) had been solved exactly (see [1]);
with the algorithm here shown problems with up to 120 nodes and using a
data a four-indexed formulation (essentially, more than 207 million variables) were
exactly solved.

The CAB data set (Civil Aeronautic Board) contains the information
about 25 USA cities with high traffic of airline passengers in 1970. Although
CAB data are symmetric, no use of this fact was made in the algorithm.

The headers of the table have the following meanings:

- Data set: $n.x$, where $n$ is the number of nodes and $L$ or $T$ values
  for $x$ mean loose or tight fixed costs, respectively. In Table 6, the
  extra symbols 1, 5 and 9 mean $\alpha = 0.1$, 0.5 and 0.9, respectively, and

\begin{table}[h]
\centering
\begin{tabular}{ccccccccc}
\hline
Data & Optimal & Method 1 & Method 2 & Method 3 \\
set & value & Upper gap & Gap & BB & CPU & & \\
\hline
10L & 221033 & 0.00 & 1.03 & 3 & 0 & 0 & 1 \\
10T & 257558 & 0.00 & 1.25 & 7 & 0 & 0 & 1 \\
20L & 230385 & 1.53 & 2.25 & 3 & 0 & 9 & 20 \\
20T & 266877 & 3.16 & 3.21 & 5 & 0 & 9 & 52 \\
25L & 232407 & 0.00 & 0.01 & 3 & 0 & 30 & 96 \\
25T & 292032 & 3.07 & 3.97 & 7 & 0 & 37 & 324 \\
40L & 237115 & 0.62 & 2.24 & 3 & 5 & * & 5359 \\
40T & 293165 & 0.00 & 1.95 & 5 & 8 & * & 7854 \\
50L & 233905 & 3.89 & 4.87 & 15 & 31 & * & 29624 \\
50T & 296025 & 0.72 & 0.94 & 7 & 34 & * & 48044 \\
\hline
\end{tabular}
\caption{Table 1: AP asymmetric problems, $\chi = 3$, $\alpha = 0.75$, $\delta = 2$}
\end{table}

Remark that, as it has been observed in the text, these computational times are from
the different machine, used in [1]: a Digital Personal workstation with a 500 MHz alpha
chip.

Asterisks mean that the machine has not memory enough to generate the matrix used
by Xpress-MP.
<table>
<thead>
<tr>
<th>Data set</th>
<th>Optimal value</th>
<th>Upper gap</th>
<th>Gap</th>
<th>BB</th>
<th>CPU (method 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>60L</td>
<td>225042</td>
<td>0.00</td>
<td>0.66</td>
<td>11</td>
<td>57</td>
</tr>
<tr>
<td>60T</td>
<td>243416</td>
<td>0.00</td>
<td>0.09</td>
<td>5</td>
<td>71</td>
</tr>
<tr>
<td>70L</td>
<td>229874</td>
<td>3.84</td>
<td>5.82</td>
<td>7</td>
<td>144</td>
</tr>
<tr>
<td>70T</td>
<td>249603</td>
<td>2.13</td>
<td>2.23</td>
<td>5</td>
<td>190</td>
</tr>
<tr>
<td>80L</td>
<td>225167</td>
<td>2.88</td>
<td>2.95</td>
<td>7</td>
<td>291</td>
</tr>
<tr>
<td>80T</td>
<td>268209</td>
<td>0.00</td>
<td>0.24</td>
<td>7</td>
<td>440</td>
</tr>
<tr>
<td>90L</td>
<td>226857</td>
<td>1.42</td>
<td>1.97</td>
<td>7</td>
<td>597</td>
</tr>
<tr>
<td>90T</td>
<td>277418</td>
<td>1.08</td>
<td>2.71</td>
<td>5</td>
<td>915</td>
</tr>
<tr>
<td>100L</td>
<td>235097</td>
<td>0.00</td>
<td>0.93</td>
<td>5</td>
<td>883</td>
</tr>
<tr>
<td>100T</td>
<td>305098</td>
<td>0.00</td>
<td>0.30</td>
<td>3</td>
<td>1534</td>
</tr>
<tr>
<td>110L</td>
<td>218662</td>
<td>0.00</td>
<td>0.61</td>
<td>3</td>
<td>1568</td>
</tr>
<tr>
<td>110T</td>
<td>223892</td>
<td>1.06</td>
<td>1.46</td>
<td>7</td>
<td>2057</td>
</tr>
<tr>
<td>120L</td>
<td>222239</td>
<td>0.31</td>
<td>1.27</td>
<td>5</td>
<td>4706</td>
</tr>
<tr>
<td>120T</td>
<td>229582</td>
<td>1.18</td>
<td>1.24</td>
<td>5</td>
<td>4643</td>
</tr>
</tbody>
</table>

Table 2: Large AP asymmetric problems, $\chi = 3$, $\alpha = 0.75$, $\delta = 2$

$\chi = \delta = 1$; letter $a$ means the asymmetric case of Table 1, that is,
$\chi = 3$, $\alpha = 0.75$, $\delta = 2$;

- Optimal value: the optimal value of the problem;
- Upper gap:
  $$100 \left( \frac{z_U^0}{z^*} - 1 \right),$$
  where $z_U^0$ and $z^*$ are respectively the objective values of the heuristic
  primal solution in the root node and the optimal solution;
- Gap:
  $$100 \left( \frac{z_L^0}{z_L^*} - 1 \right),$$
  where $z_L^0$ is the objective value of the heuristic dual solution in the root
  node;
- BB: the number of branching nodes generated;

- CPU: the computational time in seconds;

The comparison of the branch-and-bound algorithm (Method 1) with Xpress-MP (Method 2) and the method described in [1] is shown in Table 1. Despite the different computers used in Methods 1 and 3, it is clear that the new algorithm outperforms the other one. Besides, the obtained gaps were small and few branching nodes were needed to solve the problem.

Larger problems were solved in Table 2. The trend of small gaps and few branching nodes kept. It is interesting to remark that from one problem size to the next one, that is, with 10 more nodes, computational times were approximately doubled. Hence, although the difficulty of the problem grows quickly (remember the $O(n^4)$ variables), computational times do not increase exponentially but at a reasonable rate.

In Tables 3 to 5, symmetric instances with values 0.1, 0.5 and 0.9 for $\alpha$, respectively, are shown. There is again a great performance of the algorithm, being of course computational times larger for lower values of $\alpha$ due to the fact that the lower $\alpha$ is, the less effective the preprocessing is. However, observe that, as a rule, gaps are much smaller than in the asymmetric case, being respectively 6, 11 and 15 out of the 24 instances for each value of $\alpha$ exactly solved at the root node, even for the largest instances, which proves the efficiency of the algorithm.

In Table 6, several instances of the CAB data were solved, again with very good results. Demands $w_{ij}$ were been divided by 50000 for the problems shown in this table (unlike the AP data problems, where the demands were divided by 1000, see the corresponding files in the OR Library).
Finally, by beholding the whole set of problems, it can be observed that many of the initial heuristic primal solutions (the ones given at the root node) proved to be optimal (almost 70%).

6. Conclusions

This paper has shown how the dual formulation of the UMAHLP can be preprocessed. After, a very efficient heuristic method for solving this problem has been presented. The fastness of the algorithm is not only achieved because of a strong formulation, but also due to the thorough analysis of the problem.

The dual heuristic has proved to be very useful thanks to the multiple considerations done (horizons use and slackness rectification) in order to solve some troubles inherent to the use of dual-ascent techniques like the exposed in the examples (e.g., doing identical iterations in a row when the same result can be achieved in just one single iteration).

However, one difficulty of this problem is the high number of variables and thus the high memory required to solve it. Although the management of the problem is much better in the algorithm (up to 120 nodes) than in the commercial solver Xpress-MP (not even 40 nodes), the 200 nodes of the whole AP problem prove still to be a too large problem.

7. Acknowledgements

The authors’ research has been funded by Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica (I+D+I), projects TIC2000-1750-C06-06 and TIC2003-05982-C05-03.
References


<table>
<thead>
<tr>
<th>Data set</th>
<th>Optimal value</th>
<th>Upper gap</th>
<th>Gap</th>
<th>BB</th>
<th>CPU (method 1)</th>
<th>CPU (method 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10L</td>
<td>122039</td>
<td>0.00</td>
<td>0.25</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10T</td>
<td>127426</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20L</td>
<td>125310</td>
<td>3.32</td>
<td>5.30</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20T</td>
<td>129080</td>
<td>1.53</td>
<td>2.15</td>
<td>5</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>25L</td>
<td>126822</td>
<td>0.06</td>
<td>2.31</td>
<td>15</td>
<td>1</td>
<td>68</td>
</tr>
<tr>
<td>25T</td>
<td>143422</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td>41</td>
</tr>
<tr>
<td>40L</td>
<td>124994</td>
<td>0.16</td>
<td>1.96</td>
<td>7</td>
<td>18</td>
<td>*</td>
</tr>
<tr>
<td>40T</td>
<td>140963</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>15</td>
<td>*</td>
</tr>
<tr>
<td>50L</td>
<td>120872</td>
<td>0.00</td>
<td>0.08</td>
<td>5</td>
<td>59</td>
<td>*</td>
</tr>
<tr>
<td>50T</td>
<td>152294</td>
<td>4.87</td>
<td>4.93</td>
<td>3</td>
<td>81</td>
<td>*</td>
</tr>
<tr>
<td>60L</td>
<td>112992</td>
<td>0.00</td>
<td>0.17</td>
<td>5</td>
<td>147</td>
<td>*</td>
</tr>
<tr>
<td>60T</td>
<td>124961</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>184</td>
<td>*</td>
</tr>
<tr>
<td>70L</td>
<td>114596</td>
<td>0.00</td>
<td>0.11</td>
<td>5</td>
<td>402</td>
<td>*</td>
</tr>
<tr>
<td>70T</td>
<td>134324</td>
<td>0.00</td>
<td>0.60</td>
<td>5</td>
<td>625</td>
<td>*</td>
</tr>
<tr>
<td>80L</td>
<td>116506</td>
<td>0.00</td>
<td>0.24</td>
<td>5</td>
<td>991</td>
<td>*</td>
</tr>
<tr>
<td>80T</td>
<td>138971</td>
<td>0.00</td>
<td>2.08</td>
<td>3</td>
<td>1460</td>
<td>*</td>
</tr>
<tr>
<td>90L</td>
<td>115226</td>
<td>0.00</td>
<td>0.34</td>
<td>3</td>
<td>1989</td>
<td>*</td>
</tr>
<tr>
<td>90T</td>
<td>130559</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>2030</td>
<td>*</td>
</tr>
<tr>
<td>100L</td>
<td>123823</td>
<td>1.44</td>
<td>2.04</td>
<td>15</td>
<td>4750</td>
<td>*</td>
</tr>
<tr>
<td>100T</td>
<td>143120</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>4400</td>
<td>*</td>
</tr>
<tr>
<td>110L</td>
<td>110193</td>
<td>0.00</td>
<td>0.04</td>
<td>5</td>
<td>5794</td>
<td>*</td>
</tr>
<tr>
<td>110T</td>
<td>114895</td>
<td>0.00</td>
<td>0.02</td>
<td>3</td>
<td>5976</td>
<td>*</td>
</tr>
<tr>
<td>120L</td>
<td>111758</td>
<td>0.00</td>
<td>0.08</td>
<td>5</td>
<td>16122</td>
<td>*</td>
</tr>
<tr>
<td>120T</td>
<td>118377</td>
<td>0.00</td>
<td>0.15</td>
<td>7</td>
<td>20578</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 3: AP symmetric problems, $\chi = 1$, $\alpha = 0.1$, $\delta = 1$
<table>
<thead>
<tr>
<th>Data set</th>
<th>Optimal value</th>
<th>Upper gap</th>
<th>Gap</th>
<th>BB</th>
<th>CPU (method 1)</th>
<th>CPU (method 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10L</td>
<td>125592</td>
<td>0.93</td>
<td>1.08</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10T</td>
<td>127426</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20L</td>
<td>126058</td>
<td>0.00</td>
<td>0.23</td>
<td>3</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>20T</td>
<td>129080</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>25L</td>
<td>126901</td>
<td>0.00</td>
<td>0.45</td>
<td>3</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>25T</td>
<td>143422</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>40L</td>
<td>125200</td>
<td>0.00</td>
<td>0.51</td>
<td>3</td>
<td>5</td>
<td>*</td>
</tr>
<tr>
<td>40T</td>
<td>140963</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>5</td>
<td>*</td>
</tr>
<tr>
<td>50L</td>
<td>124917</td>
<td>4.71</td>
<td>5.13</td>
<td>5</td>
<td>20</td>
<td>*</td>
</tr>
<tr>
<td>50T</td>
<td>152294</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>21</td>
<td>*</td>
</tr>
<tr>
<td>60L</td>
<td>116799</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>39</td>
<td>*</td>
</tr>
<tr>
<td>60T</td>
<td>124961</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>44</td>
<td>*</td>
</tr>
<tr>
<td>70L</td>
<td>120503</td>
<td>0.00</td>
<td>0.27</td>
<td>5</td>
<td>113</td>
<td>*</td>
</tr>
<tr>
<td>70T</td>
<td>135017</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>112</td>
<td>*</td>
</tr>
<tr>
<td>80L</td>
<td>119406</td>
<td>0.00</td>
<td>0.20</td>
<td>3</td>
<td>227</td>
<td>*</td>
</tr>
<tr>
<td>80T</td>
<td>138971</td>
<td>0.00</td>
<td>0.46</td>
<td>3</td>
<td>295</td>
<td>*</td>
</tr>
<tr>
<td>90L</td>
<td>118612</td>
<td>4.55</td>
<td>4.85</td>
<td>3</td>
<td>475</td>
<td>*</td>
</tr>
<tr>
<td>90T</td>
<td>130559</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>448</td>
<td>*</td>
</tr>
<tr>
<td>100L</td>
<td>125484</td>
<td>0.10</td>
<td>0.63</td>
<td>5</td>
<td>836</td>
<td>*</td>
</tr>
<tr>
<td>100T</td>
<td>143120</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>854</td>
<td>*</td>
</tr>
<tr>
<td>110L</td>
<td>116255</td>
<td>0.00</td>
<td>0.09</td>
<td>5</td>
<td>1282</td>
<td>*</td>
</tr>
<tr>
<td>110T</td>
<td>121485</td>
<td>0.00</td>
<td>0.14</td>
<td>5</td>
<td>1466</td>
<td>*</td>
</tr>
<tr>
<td>120L</td>
<td>118048</td>
<td>0.00</td>
<td>0.25</td>
<td>5</td>
<td>3167</td>
<td>*</td>
</tr>
<tr>
<td>120T</td>
<td>122850</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>2971</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 4: AP symmetric problems, $\chi = 1$, $\alpha = 0.5$, $\delta = 1$
<table>
<thead>
<tr>
<th>Data set</th>
<th>Optimal value</th>
<th>Upper gap</th>
<th>Gap</th>
<th>BB</th>
<th>CPU (method 1)</th>
<th>CPU (method 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10L</td>
<td>125592</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10T</td>
<td>127426</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20L</td>
<td>126058</td>
<td>0.00</td>
<td>0.01</td>
<td>3</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>20T</td>
<td>129080</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>25L</td>
<td>126901</td>
<td>0.00</td>
<td>0.33</td>
<td>3</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>25T</td>
<td>143422</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>40L</td>
<td>125200</td>
<td>0.00</td>
<td>0.29</td>
<td>3</td>
<td>3</td>
<td>*</td>
</tr>
<tr>
<td>40T</td>
<td>140963</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>4</td>
<td>*</td>
</tr>
<tr>
<td>50L</td>
<td>124917</td>
<td>4.71</td>
<td>4.84</td>
<td>5</td>
<td>12</td>
<td>*</td>
</tr>
<tr>
<td>50T</td>
<td>152294</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>14</td>
<td>*</td>
</tr>
<tr>
<td>60L</td>
<td>116799</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>20</td>
<td>*</td>
</tr>
<tr>
<td>60T</td>
<td>124961</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>24</td>
<td>*</td>
</tr>
<tr>
<td>70L</td>
<td>121859</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>47</td>
<td>*</td>
</tr>
<tr>
<td>70T</td>
<td>135017</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>59</td>
<td>*</td>
</tr>
<tr>
<td>80L</td>
<td>119406</td>
<td>0.00</td>
<td>0.07</td>
<td>3</td>
<td>105</td>
<td>*</td>
</tr>
<tr>
<td>80T</td>
<td>138971</td>
<td>0.00</td>
<td>0.22</td>
<td>3</td>
<td>140</td>
<td>*</td>
</tr>
<tr>
<td>90L</td>
<td>118612</td>
<td>4.55</td>
<td>4.65</td>
<td>3</td>
<td>205</td>
<td>*</td>
</tr>
<tr>
<td>90T</td>
<td>130559</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>205</td>
<td>*</td>
</tr>
<tr>
<td>100L</td>
<td>125484</td>
<td>0.39</td>
<td>0.58</td>
<td>5</td>
<td>335</td>
<td>*</td>
</tr>
<tr>
<td>100T</td>
<td>143120</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>395</td>
<td>*</td>
</tr>
<tr>
<td>110L</td>
<td>119008</td>
<td>0.00</td>
<td>0.53</td>
<td>5</td>
<td>530</td>
<td>*</td>
</tr>
<tr>
<td>110T</td>
<td>122257</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>470</td>
<td>*</td>
</tr>
<tr>
<td>120L</td>
<td>119561</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>747</td>
<td>*</td>
</tr>
<tr>
<td>120T</td>
<td>122850</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
<td>807</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 5: AP symmetric problems, $\chi = 1, \alpha = 0.9, \delta = 1$
<table>
<thead>
<tr>
<th>Data set</th>
<th>Optimal value</th>
<th>Upper gap</th>
<th>Gap</th>
<th>BB</th>
<th>CPU (method 1)</th>
<th>CPU (method 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25La</td>
<td>390369</td>
<td>2.02</td>
<td>2.03</td>
<td>3</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>25L1</td>
<td>196903</td>
<td>0.00</td>
<td>0.10</td>
<td>5</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>25L5</td>
<td>234523</td>
<td>2.60</td>
<td>3.13</td>
<td>7</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>25L9</td>
<td>256691</td>
<td>0.38</td>
<td>0.78</td>
<td>9</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>25Ta</td>
<td>484591</td>
<td>1.95</td>
<td>2.15</td>
<td>5</td>
<td>0</td>
<td>26</td>
</tr>
<tr>
<td>25T1</td>
<td>258577</td>
<td>0.42</td>
<td>3.45</td>
<td>9</td>
<td>1</td>
<td>48</td>
</tr>
<tr>
<td>25T5</td>
<td>279144</td>
<td>0.00</td>
<td>2.20</td>
<td>7</td>
<td>0</td>
<td>41</td>
</tr>
<tr>
<td>25T9</td>
<td>284852</td>
<td>0.00</td>
<td>2.43</td>
<td>7</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 6: CAB problems