AN EXACT ALGORITHM FOR A LOCATION-ROUTING PROBLEM

Martine Labbé¹, Inmaculada Rodríguez Martín², Juan José Salazar González²

¹ISRO and SMG, Université Libre de Bruxelles, C.P. 210/01, B-1050, Bruxelles, Belgium
E-mail: mlabbe@smg.ulb.ac.be
²Departamento de Estadística, Investigación Operativa y Computación
Universidad de La Laguna, 38271 Tenerife, España
E-mail: irguez@ull.es, jjsalaza@ull.es

ABSTRACT

In this paper we address a combinatorial problem, the Plant-Cycle Location Problem (PCLP), with applications in routing and telecommunications. For this NP-hard problem we propose an integer linear programming formulation, and we develop an exact branch-and-cut algorithm. Preliminary computational results on randomly generated instances are presented.

Key words: routing, location, branch-and-cut.
AMS code: 90B06, 90B80

1 Introduction

We present the Plant-Cycle Location Problem (PCLP), related with the well-known Capacitated Facility Location Problem (CFLP) (see Cornuejols et al. (1990)), and defined as follows. Let us consider two sets of locations, one with customers and another with potential plants (or depots). The travel distance between two locations is assumed to be known and symmetric. The opening of each potential plant has a given known cost. Moreover, each potential plant has a capacity that limits the number of customers it can serve. As in the CFLP, PCLP consists of choosing the plants to be opened and the assignment of customers to opened plants minimizing the cost. The novelty respect to the CFLP is that in PCLP the customers assigned
to a plant must be served with a cycle, thus the cost of a solution also includes the Routing cost.
Therefore, the PCLP looks for a set of disjoint cycles covering all the customers, each one containing exactly one plant, and minimizing the total sum of

- cost of opening the selected plants,
- cost of customer-plant assignments,
- routing cost.

The problem considers capacity constraints limiting the number of customers that can be served by an open plant. In a more general case, a different demand could be associated to each customer and the constraint would limit the total demand served by each plant. Nevertheless, this extension was not present in the application motivating the PCLP.
When there is only one plant and there are not assignment costs, the uncapacitated PCLP reduces to the Traveling Salesman Problem (TSP). Moreover, the PCLP reduces to the Vehicle Routing Problem with unit demand (VRP) when all the plants have the same location. Hence, PCLP is $\mathcal{NP}$-hard in the strong sense and has several applications in the routing context. See, e.g., Toth and Vigo (2001) for a survey on the CVRP.
We present a 0-1 integer linear programming model, and describe a branch-and-cut algorithm for the exact solution based on several families of inequalities strengthening the LP-relaxation of the model. We also provide computational results showing the good performance of our approach on random instances.

## 2 Mathematical Model

Let $V = I \cup J$ be the set of locations, where $I$ represents the locations of the customers and $J$ the potential plant locations. Let $E$ be the set of undirected edges linking all possible pairs of locations in $V$ and $G = (V, E)$ the graph on which the PCLP is defined. Each potential plant location $j \in J$ has an associated cost $f_j$, and can serve at most $q_j$ customers. Associated to each customer $i \in I$ and each potential depot $j \in J$ there is an assignment cost $d_{ij}$, and associated to each edge $e \in E$ there is a routing cost $c_e$.

The PCLP can be mathematically formulated by defining the following variables. For each plant $j \in J$, let $y_j$ be a binary variable that takes value 1 if the plant $j$ is open in the solution and 0 otherwise. Every edge $e \in E$ has an associated integer variable $x_e$ taking value 2 if one of its extreme vertices is a plant and the other is a customer, and they are the only points in a cycle; value 1 if the edge $e$ is part of a cycle visiting other customers besides its extremes; and value 0 otherwise. For each $i \in I$ and each $j \in J$ a binary variable $z_{ij}$ takes value 1 if the customer $i \in I$ is assigned to the plant $j \in J$, and value 0 otherwise.
In addition, for each set of vertices $S \subset V$ we define

$$\delta(S) := \{ [u, v] \in E : u \in S, v \notin S \}$$

$$E(S) := \{ [u, v] \in E : u \in S, v \in S \}$$

and for any $v \in V$ we write $\delta(v)$ instead of $\delta(\{v\})$. Moreover, for each $F \subset E$ we write $x(F)$ instead of $\sum_{e \in F} x_e$.

Then the model is the following:

$$\begin{align*}
\text{minimize} \quad & \sum_{j \in J} f_j y_j + \sum_{e \in E} c_e x_e + \sum_{i \in I} \sum_{j \in J} d_{ij} z_{ij} \\
\text{subject to:} \quad & \sum_{j \in J} z_{ij} = 1 \quad \text{for all } i \in I, \\
& \sum_{i \in I} z_{ij} \leq q_j y_j \quad \text{for all } j \in J, \\
& x(\delta(i)) = 2 \quad \text{for all } i \in I, \\
& x(\delta(j)) = 2 y_j \quad \text{for all } j \in J, \\
& x(\delta(S)) \geq 2 \sum_{j \in J \setminus S} z_{ij} \quad \text{for all } S \subset V, i \in S \cap I, \\
& x_{ij} \leq 2z_{ij} \quad \text{for all } i \in I, j \in J, \\
& x_{ii'} + z_{ij} + z_{i'j'} \leq 2 \quad \text{for all } i, i' \in I, j, j' \in J, \\
& y_j \in \{0, 1\} \quad \text{for all } j \in J, \\
& z_{ij} \in \{0, 1\} \quad \text{for all } i \in I, j \in J, \\
& x_{ij} \in \{0, 1, 2\} \quad \text{for all } [i, j] \in E.
\end{align*}$$

Constraints (2) enforce each customer $i$ to be assigned to exactly one plant $j$. Capacity constraints (3) limit to $q_j$ the number of customers a plant $j \in J$ can serve, and prevent customers to be assigned to a plant not open. Constraints (4) and (5) are degree constraints, and they ensure that the degree of every customer is 2, and that the degree of every potential plant is 2 if and only if it belongs to a cycle. Constraints (6) are connectivity constraints. They state that each set of vertices $S \subset V$ must be connected to its complement by at least 2 edges whenever there is pair of vertices $i$ and $j$ such that $i$ is a customer in $S$, $j$ is a plant not in $S$, and $i$ is assigned to $j$. Constraints (7) state that if customer $i$ is not assigned to plant $j$ then the edge $[i, j]$ can not be routed. Constraints (8) state that if customers $i$ and $i'$ are assigned to different depots $j$ and $j'$, then they cannot be in the same cycle. Finally, Constraints (9)–(11) are the integrality constraints for the different kind of variables.
3 Branch-and-cut algorithm and computational results

A branch-and-cut algorithm is a branch-and-bound algorithm in which cutting planes are generated throughout the search tree (see Padberg and Rinaldi (1991)). The idea is to try to close as much as possible the gap between the LP-relaxation optimal value and the integer optimal value of each subproblem in order to significantly reduce the number of nodes in the search tree. An important point within this scheme is to be able to develop efficient separation routines to generate cuts in the separation phase. In fact, the different families of valid inequalities that strengthen the LP-relaxation are not of practical use unless their associated separations problem, i.e., the problem of identifying some of them that are violated by a given fractional solution, can be easily solved.

We have found the following families of valid inequalities for the PCLP, and designed the corresponding separation procedures.

**Logical constraints:**

\[ z_{ij} \leq y_j \quad \text{for all } i \in I, j \in J, \quad (12) \]

**Path inequalities:**

\[ x(P) + \sum_{j \in Q} z_{ij} + \sum_{j \notin Q} z_{i'j} \leq |P| + 1, \quad (13) \]

for each \( Q \subseteq J, \{i, i'\} \subseteq I \) and \( P := \{[i, i_1], [i_1, i_2], ..., [i_k, i']\} \) a path from \( i \) to \( i' \).

**Generalized connectivity constraints:**

\[ x(\delta(S)) \geq 2(\sum_{j \in I \setminus S} z_{ij} + \sum_{j \in J \setminus S} z_{i'j}), \quad (14) \]

for each \( S \subseteq V, i \in S \cap I, i' \in I \setminus S \).

**2-matching constraints:**

\[ x(E(H)) + x(T) \leq |H \cap I| + \sum_{v_j \in H} y_j + \sum_{v_j \in T} y_j + (|T| - 1)/2 \quad (15) \]

for each \( H \subset V \) and \( T \subset \delta(H) \), where \( T = T_I \cup T_J \), being \( T_I \) the set of edges in \( T \) linking customers and \( T_J \) the set of edges of \( T \) that have a plant in one extreme, and \( H \) and \( T \) satisfying:

(i) \( \{i, j\} \cap \{k, l\} = \emptyset \) for \( [i, j], [k, l] \in T \) and \( [i, j] \neq [k, l] \),

(ii) \( |T| \geq 3 \) and odd.
The branch-and-cut algorithm was implemented in the C++ programming language. ABACUS 2.3 linked with CPLEX 6.0 was used as a branch-and-cut framework. See Jünger and Thienel (1997) for details on this software. The implementation was run on a PC Pentium IV 1.5 Ghz under Linux.

In Table (1) we present preliminary computational results on a set of randomly generated instances, with vertices coordinates in $[0,1000] \times [0,1000]$. For each value of $|I|$, five different instances were generated, and the results are shown on average over the successful instances, i.e., those solved in less than three hours.

4 Acknowledgements

This work was partially supported by “Ministerio de Ciencia y Tecnología” (TIC2000-1750-C06-02) and by “Gobierno de Canarias” (PI2000/116), Spain.

References

<table>
<thead>
<tr>
<th>n</th>
<th></th>
<th></th>
<th>succ</th>
<th>nodes</th>
<th>time</th>
<th>open</th>
<th>%-LB</th>
<th>%-UB</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>50</td>
<td>3</td>
<td>1</td>
<td>453.00</td>
<td>0:48:05</td>
<td>19.00</td>
<td>98.79</td>
<td>103.12</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>3</td>
<td>2</td>
<td>453.00</td>
<td>0:48:05</td>
<td>19.00</td>
<td>98.79</td>
<td>103.12</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>5</td>
<td>1</td>
<td>4.20</td>
<td>0:00:11</td>
<td>24.00</td>
<td>99.81</td>
<td>100.07</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>5</td>
<td>2</td>
<td>4.20</td>
<td>0:00:11</td>
<td>24.00</td>
<td>99.81</td>
<td>100.07</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>5</td>
<td>5</td>
<td>2.00</td>
<td>0:00:56</td>
<td>13.80</td>
<td>99.97</td>
<td>100.24</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>5</td>
<td>10</td>
<td>1.40</td>
<td>0:00:23</td>
<td>11.40</td>
<td>99.99</td>
<td>100.66</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>5</td>
<td>25</td>
<td>1.00</td>
<td>0:00:02</td>
<td>23.80</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>10</td>
<td>1</td>
<td>1.00</td>
<td>0:00:02</td>
<td>23.80</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>10</td>
<td>2</td>
<td>1.00</td>
<td>0:00:02</td>
<td>23.80</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1.40</td>
<td>0:00:24</td>
<td>11.40</td>
<td>99.98</td>
<td>100.62</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>10</td>
<td>10</td>
<td>1.40</td>
<td>0:00:24</td>
<td>11.40</td>
<td>99.98</td>
<td>100.62</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>10</td>
<td>25</td>
<td>1.40</td>
<td>0:00:24</td>
<td>11.40</td>
<td>99.98</td>
<td>100.62</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>15</td>
<td>1</td>
<td>1.00</td>
<td>0:00:02</td>
<td>23.80</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>15</td>
<td>5</td>
<td>2.20</td>
<td>0:00:54</td>
<td>13.80</td>
<td>99.97</td>
<td>100.24</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>15</td>
<td>25</td>
<td>1.40</td>
<td>0:00:24</td>
<td>11.40</td>
<td>99.98</td>
<td>100.62</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>15</td>
<td>50</td>
<td>8.20</td>
<td>0:01:59</td>
<td>8.80</td>
<td>99.85</td>
<td>100.36</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>15</td>
<td>500</td>
<td>43.00</td>
<td>0:04:34</td>
<td>5.00</td>
<td>99.50</td>
<td>101.36</td>
</tr>
</tbody>
</table>

**Tabla 1:** Computational results for random instances