Complementary Approaches for a Clustering Problem

Edson L.F. Senne*, Luiz A.N. Lorena**

*FEG/UNESP - Universidade Estadual Paulista
Faculdade de Engenharia - Departamento de Matemática
12500-000 Guaratinguetá, SP - Brazil
e-mail: elfsenne@feg.unesp.br

**LAC/INPE - Instituto Nacional de Pesquisas Espaciais
Av. dos Astronautas, 1758 - Caixa Postal 515
12227-010 São José dos Campos, SP - Brazil
e-mail: lorena@lac.inpe.br

Abstract

The lagrangean/surrogate relaxation has been explored as a faster computational alternative to traditional lagrangean heuristics. This paper discusses two approaches for using lagrangean/surrogate heuristics to a classical clustering problem: the p-median problem, that is, how to locate p facilities (medians) on a network such as the sum of all the distances from each vertex to its nearest facility is minimized.

Key words: Linear Programming, Networks, Optimization Techniques, Heuristics.
1. Introduction

The search for p-median nodes on a network is a classical clustering problem. The objective is to locate p facilities (medians) such as the sum of the distances from each demand point to its nearest facility is minimized. The problem is well known to be NP-hard (Garey and Johnson, 1979) and several heuristics have been developed for p-median problems. The combined used of lagrangean relaxation and subgradient optimization in a primal-dual viewpoint was found to be a good solution approach to the problem (Beasley, 1985). By other hand, column generation is a powerful tool for solving large scale linear programming problems and has been a natural choice in several applications, such as the well-known cutting-stock problem, vehicle routing and crew scheduling (Gilmore and Gomory, 1961, 1963; Desrochers and Soumis, 1989; Desrochers et al., 1992; Vance et al., 1994). However, the use of column generation to solve p-median problems was not sufficiently explored. The initial attempts appear to be the ones of Garfinkel et al. (1974) and Swain (1974), that report convergence problems, even for small instances, when the number of medians is small compared to the number of candidate points in the network. This observation was also confirmed later by Galvão (1981). The solution of large-scale instances using a stabilized approach is reported by du Merle et al. (1999).

The lagrangean/surrogate relaxation has been explored recently as a faster computational alternative to traditional lagrangean heuristics. This new relaxation has been used in a number of applications, such as multidimensional knapsack problems (Freville et al., 1990), set covering problems (Lorena and Lopes, 1994), generalized assignment problems (Lorena and Narciso, 1996), and also p-median problems (Senne and Lorena, 2000).

The lagrangean/surrogate relaxation and the traditional column generation approach can be combined to accelerate and stabilize primal and dual bounds obtained using the reduced cost selection. It is well known the equivalencies of the Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960), column generation and lagrangean relaxation optimization. Solving a linear programming by Dantzig-Wolfe decomposition is the same as solving the lagrangean by Kelley's cutting plane method (Kelley, 1960). However, as observed before, in many cases a straightforward application of column generation may result in slow convergence. The lagrangean/surrogate relaxation can be used to stabilize and accelerate the column generation process, providing the selection of new productive columns.
This paper discusses these two approaches of using lagrangean/surrogate relaxation to solve P-median problems: (i) combined with subgradient optimization in a primal-dual viewpoint, and (ii) combined with the column generation process for linear programming problems. The paper is organized as follows. Section 2 presents the lagrangean/surrogate relaxation combined with subgradient optimization heuristic. Section 3 discusses the combination of lagrangean/surrogate relaxation and column generation process. The computational tests, conducted with problems from the literature, are presented in the Section 4. We conclude that these two approaches of using lagrangean/surrogate relaxation to solve P-median problems are complementary and can be put to work together.

2. The Lagrangean/Surrogate Relaxation and Subgradient Optimization

The P-median problem can be modeled as the following binary integer programming problem:

\[ v(P) = \min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \]

subject to

\[ \sum_{i=1}^{n} x_{ij} = 1 \; ; \; j \in N \]  \hspace{1cm} (1)

\[ \sum_{i=1}^{n} x_{ii} = p \]  \hspace{1cm} (2)

\[ x_{ij} \leq x_{ii} \; ; \; i, j \in N \]  \hspace{1cm} (3)

\[ x_{ij} \in \{0,1\} ; i, j \in N \]  \hspace{1cm} (4)

where: \([d_{ij}]_{n \times n}\) is a symmetric cost (distance) matrix, with \(d_{ii} = 0, \forall i\); \([x_{ij}]_{n \times n}\) is the allocation matrix, with \(x_{ij} = 1\) if node \(i\) is allocated to node \(j\), and \(x_{ij} = 0\), otherwise; \(x_{ii} = 1\) if node \(i\) is a median and \(x_{ii} = 0\), otherwise; \(p\) is the number of facilities (medians) to be located; \(n\) is the number of nodes in the network, and \(N = \{1, ..., n\}\). Constraints (1) and (3) ensure that each node \(j\) is allocated to only one node \(i\), which must be a median. Constraint (2) determines the exact number of medians to be located (p), and (4) gives the integer conditions.

For a given \(\lambda \in \mathbb{R}^n_+\) and \(t \geq 0\), the lagrangean/surrogate relaxation of \((P)\) can be defined by:
\[ v(L_{\text{tSP}}^\lambda) = \min \sum_{j=1}^{n} \sum_{i=1}^{n} (d_{ij} - t\lambda_{j})x_{ij} + t\sum_{j=1}^{n} \lambda_{j} \]
subject to (2), (3) and (4).

The problem \((L_{\text{tSP}}^\lambda)\) is solved considering implicitly constraint (2) and decomposing for index \(i\), obtaining the following \(n\) problems:
\[ \min \sum_{j=1}^{n} (d_{ij} - t\lambda_{j})x_{ij} \]
subject to (3) and (4).

Each problem is easily solved by letting: \(\beta_{i} = \sum_{j=1}^{n} \min \{ 0, d_{ij} - t\lambda_{j} \}\), and choosing \(I\) as the index set of the \(p\) smallest \(\beta_{i}\) (here constraint (2) is implicitly considered). Then, a solution \(x_{ij}^\lambda\) to the problem \((L_{\text{tSP}}^\lambda)\) is:
\[ x_{ij}^\lambda = \begin{cases} 1, & \text{if } i \in I \\ 0, & \text{otherwise} \end{cases} \]
and for all \(i \neq j\),
\[ x_{ij}^\lambda = \begin{cases} 1, & \text{if } i \in I \text{ and } d_{ij} - t\lambda_{j} < 0 \\ 0, & \text{otherwise} \end{cases} \]

The lagrangean/surrogate solution is given by: \(v(L_{\text{tSP}}^\lambda) = \sum_{i=1}^{n} \beta_{i}x_{ii} + t\sum_{j=1}^{n} \lambda_{j}\). The interesting characteristic of relaxation \((L_{\text{tSP}}^\lambda)\), is that for \(t = 1\) we have the usual lagrangean relaxation using the multiplier \(\lambda\). For a fixed multiplier \(\lambda\), the best value for \(t\) can result of a lagrangean dual:
\[ v(D_{\lambda}^t) = \max_{t \geq 0} v(L_{\text{tSP}}^\lambda). \]

A search procedure for finding an approximate value for \(t\) is shown in (Senne and Lorena, 2000).

The combined use of lagrangean/surrogate and subgradient optimization is given by the following algorithm:

**Algorithm LSSH** (lagrangean/surrogate subgradient heuristic)

Given \(\lambda \geq 0, \lambda \neq 0\);
Set \( lb = -\infty, \; ub = +\infty \);

Repeat

- Solve relaxation \((L_tSP^{\lambda})\) obtaining \( x^{\lambda} \) and \( v(L_tSP^{\lambda}) \);
- Obtain a feasible solution \( x_f \) and update \( v_f \) accordingly;
- Update \( lb = \max \{ lb, v(L_tSP^{\lambda}) \} \);
- Update \( ub = \min \{ ub, v_f \} \);
- Set \( g_i^{\lambda} = 1 - \sum x_{ij}^{\lambda}, \; i \in N \);
- Update the step size \( \theta \);
- Set \( \lambda_i = \max \{ 0, \lambda_i + \theta \cdot g_i^{\lambda} \}, \; i \in N \);

Until (stopping tests).

The initial \( \lambda \) used is \( \lambda_{i} = \min \{ d_{ij} \}, \; i \in N \). The step sizes used are: \( \theta = \pi (ub - lb) / \| g^{\lambda} \|^2 \).

The control of parameter \( \pi \) is the Held and Karp (1971) classical control. It makes \( 0 \leq \pi \leq 2 \), beginning with \( \pi = 2 \) and halving \( \pi \) whenever \( lb \) not increases for 30 successive iterations. The stopping tests used are: number of iterations greater than 1000, \( \pi \leq 0.005 \), and \( (ub - lb) < 1 \).

Solution \( x^{\lambda} \) is not necessarily feasible to \((P)\), but the set \( I \) identify median nodes that can be used to produce feasible solutions to \((P)\). Two heuristics are used to make \( x^{\lambda} \) primal feasible. The first calculates the upper bound at each iteration of LSSH while \( \pi \) is not halved.

This heuristic simply makes: \( v_f = \sum_{i=1}^{n} (\min d_{ij}) \). The second, as suggested by Beasley (1993), is an interchange heuristic which is used when \( \pi \) is updated to \( \pi/2 \).

3. The Lagrangean/Surrogate Relaxation and Column Generation

The p-median problem can be also modeled as the following set partition problem:

\[
\begin{align*}
\text{v(SPP)} = \text{Min} & \sum_{j=1}^{m} c_j x_j \\
\text{subject to} & \sum_{j=1}^{m} A_j x_j = 1
\end{align*}
\] (5)
\[ \sum_{j=1}^{m} x_j = p \quad (6) \]

\[ x_j \in \{0,1\} \]

where: \( S = \{S_1, S_2, \ldots, S_m\} \) is a set of subsets of \( N \); \( A = [a_{ij}]_{n \times m} \) is a matrix with \( a_{ij} = 1 \) if \( i \in S_j \), and \( a_{ij} = 0 \), otherwise; and \( c_j = \min_{i \in S_j} \left( \sum_{k \in S_j} d_{ik} \right) \). This formulation is found in Minoux (1987). The same formulation can be obtained from the problem (P) applying the Dantzig-Wolfe decomposition considered by Garfinkel et al. (1974) and Swain (1974). If \( S \) is the set of all subsets of \( N \), the formulation can give an optimal solution to the \( p \)-median problem. But the number of subsets can be huge, and a partial set of columns should be considered. Problem (SPP) is also known as the restricted master problem in the column generation context (Barnhart et al., 1998).

In this paper we consider the following linear programming set covering relaxation of (SPP):

\[ v(SCP) = \min \sum_{j=1}^{m} c_j x_j \]

subject to

\[ \sum_{j=1}^{m} A_j x_j \geq 1 \quad (7) \]

\[ \sum_{j=1}^{m} x_j = p \]

\[ x_j \in [0,1]. \]

Observe that \( d_{ij} \geq 0, \forall i,j \) and (5) can be replaced with (7) in the linear model.

The lagrangean/surrogate relaxation is integrated to the column generation process transferring the multipliers \( \pi_j \) (\( j = 1,\ldots,n \)) of problem (SCP) to the problem \( \max_{t \geq 0} v(\text{LS}_t, P^\pi) \). The median with smallest contribution on \( v(\max_{t \geq 0} v(\text{LS}_t, P^\pi)) \) (and allocated non-mediants) results to be the one selected to produce the incoming column on the sub-problem:

\[ v(\text{Sub}_1 P) = \min_{j \in N} \left[ \min_{y_j \in (0,1)} \sum_{i=1}^{n} (d_{ij} - t \pi_j) y_{ij} \right]. \]
The reduced cost (for $t = 1$) is $rc = v(Sub_1 - P) - \alpha$ and $rc < 0$ is the condition for incoming columns, but it well known (Barnhart et al., 1998) that, for $j = 1, \ldots, n$, all the corresponding columns $\begin{bmatrix} y_{ij} \\ 1 \end{bmatrix}$ satisfying: $\begin{bmatrix} \min_{y_{ij} \in \{0,1\}} \sum_{i=1}^{n} (d_{ij} - \pi_j) y_{ij} \end{bmatrix} < |\alpha|$, can be added to the pool of columns, accelerating the column generation process.

The combined use of lagrangean/surrogate and column generation process is given by the following algorithm:

**Algorithm LSCG** (lagrangean/surrogate column generation heuristic)

1. Set an initial pool of columns to (SCP);
2. Solve (SCP) obtaining the duals prices $\pi_j, j = 1, \ldots, n$, and $\alpha$;
3. Solve approximately a local lagrangean/surrogate dual $\max v(LS_t P^\pi)$, returning the corresponding columns of (Sub$_t$P);
4. Append to (SCP) the columns $\begin{bmatrix} y_{ij} \\ 1 \end{bmatrix}$ satisfying $\begin{bmatrix} \min_{y_{ij} \in \{0,1\}} \sum_{i=1}^{n} (d_{ij} - \pi_j) y_{ij} \end{bmatrix} < |\alpha|$;
5. If no columns are found in step 4 then stop;
6. Perform tests to remove columns and return to step 2.

Note that, by setting $t = 1$, the algorithm LSCG gives the traditional column generation process.

**4. Computational Tests**

The approaches discussed above were programmed in C and run on a Sun Ultra 30 workstation. First, we have considered instances drawn from OR-Library (Beasley, 1990) for which optimal solutions are known. The results are reported in the table below. Table 1 reports the results for LSSH and LSCG algorithms and contains:

- the number of nodes in the network and the number of medians to be located;
- the optimal integer solution for the instance;
- \( p_{\text{gap}} = 100 \times \frac{|v(\text{SCP}) - \text{optimal}|}{\text{optimal}} \), that is, the percentage deviation from optimal to the best primal solution value \( v(\text{SCP}) \) found;
- \( d_{\text{gap}} = 100 \times \frac{(\text{optimal} - v(L_S P^\pi))}{\text{optimal}} \), that is, the percentage deviation from optimal to the best relaxation value \( v(L_S P^\pi) \) found;
- the total computational time (in seconds).

Table 1: Computational results for OR-Library instances

<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>n</th>
<th>( p_{\text{gap}} )</th>
<th>( d_{\text{gap}} )</th>
<th>total time</th>
<th>( p_{\text{gap}} )</th>
<th>( d_{\text{gap}} )</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>5</td>
<td>5819</td>
<td>-</td>
<td>-</td>
<td>0.76</td>
<td>-</td>
<td>-</td>
<td>36.35</td>
</tr>
<tr>
<td>100</td>
<td>33</td>
<td>1355</td>
<td>-</td>
<td>-</td>
<td>1.14</td>
<td>-</td>
<td>-</td>
<td>0.37</td>
</tr>
<tr>
<td>200</td>
<td>5</td>
<td>7824</td>
<td>-</td>
<td>0.523</td>
<td>5.61</td>
<td>-</td>
<td>-</td>
<td>902.77</td>
</tr>
<tr>
<td>200</td>
<td>10</td>
<td>5631</td>
<td>-</td>
<td>-</td>
<td>5.87</td>
<td>-</td>
<td>-</td>
<td>996.00</td>
</tr>
<tr>
<td>200</td>
<td>67</td>
<td>1255</td>
<td>-</td>
<td>-</td>
<td>12.33</td>
<td>-</td>
<td>-</td>
<td>1.29</td>
</tr>
<tr>
<td>300</td>
<td>5</td>
<td>7696</td>
<td>-</td>
<td>0.046</td>
<td>12.94</td>
<td>0.246</td>
<td>1.796</td>
<td>17889.12</td>
</tr>
<tr>
<td>300</td>
<td>10</td>
<td>6634</td>
<td>-</td>
<td>0.131</td>
<td>14.43</td>
<td>-</td>
<td>-</td>
<td>10749.91</td>
</tr>
<tr>
<td>300</td>
<td>30</td>
<td>4374</td>
<td>-</td>
<td>-</td>
<td>15.74</td>
<td>-</td>
<td>-</td>
<td>831.22</td>
</tr>
<tr>
<td>300</td>
<td>100</td>
<td>1729</td>
<td>-</td>
<td>-</td>
<td>57.73</td>
<td>0.116</td>
<td>0.058</td>
<td>4.55</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>8162</td>
<td>-</td>
<td>0.866</td>
<td>18.48</td>
<td>0.686</td>
<td>1.662</td>
<td>52807.93</td>
</tr>
<tr>
<td>400</td>
<td>10</td>
<td>6999</td>
<td>-</td>
<td>0.440</td>
<td>24.23</td>
<td>-</td>
<td>-</td>
<td>36829.25</td>
</tr>
<tr>
<td>400</td>
<td>40</td>
<td>4809</td>
<td>-</td>
<td>-</td>
<td>46.62</td>
<td>-</td>
<td>-</td>
<td>1055.20</td>
</tr>
<tr>
<td>400</td>
<td>133</td>
<td>1789</td>
<td>-</td>
<td>-</td>
<td>231.51</td>
<td>0.112</td>
<td>0.950</td>
<td>6.21</td>
</tr>
<tr>
<td>500</td>
<td>167</td>
<td>1828</td>
<td>-</td>
<td>-</td>
<td>377.14</td>
<td>0.055</td>
<td>0.310</td>
<td>11.00</td>
</tr>
<tr>
<td>600</td>
<td>200</td>
<td>1989</td>
<td>-</td>
<td>-</td>
<td>879.95</td>
<td>0.302</td>
<td>0.285</td>
<td>15.81</td>
</tr>
<tr>
<td>700</td>
<td>233</td>
<td>1847</td>
<td>-</td>
<td>-</td>
<td>494.52</td>
<td>0.081</td>
<td>0.379</td>
<td>21.50</td>
</tr>
<tr>
<td>800</td>
<td>267</td>
<td>2026</td>
<td>-</td>
<td>-</td>
<td>1360.49</td>
<td>0.518</td>
<td>0.346</td>
<td>26.14</td>
</tr>
<tr>
<td>900</td>
<td>300</td>
<td>2106</td>
<td>0.047</td>
<td>0.004</td>
<td>2994.11</td>
<td>0.518</td>
<td>0.827</td>
<td>33.37</td>
</tr>
</tbody>
</table>

Table 1 shows that the combined use of lagrangean/surrogate and column generation can be very interesting, especially for large-scale problems. The results of Table 1 also show that for a given number of nodes, smaller the number of medians in the instance, harder is the problem to be solved using the column generation approach. The opposite occurs for lagrangean/surrogate approach combined with subgradient search methods, i.e., the instances for which the number of medians is about a third of the number the nodes seem to be easy to LSCG and hard to LSSH.

The computational tests for a large-scale instance drawn from the TSPLIB, compiled by Reinelt (1998), confirm these conjectures. Table 2 shows the results for the Pcb3038 instance.
We can note that as the number of median increases, the performance of LSCG improves in such a way that, for \( p = 500 \), it is better than LSSH. In this table, \( p_{\text{gap}} \) and \( d_{\text{gap}} \) are calculated as follows:

\[
P_{\text{gap}} = 100 \times \left( \frac{|v(\text{SCP}) - \text{best known solution}|}{\text{best known solution}} \right)
\]

\[
d_{\text{gap}} = 100 \times \left( \frac{\text{best known solution} - v(LS_{\Pi}^\pi)}{\text{best known solution}} \right)
\]

Table 2: Computational results for OR-Library instances

<table>
<thead>
<tr>
<th>p</th>
<th>best known solution</th>
<th>LSSH</th>
<th>LSCG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p_{\text{gap}}</td>
<td>d_{\text{gap}}</td>
<td>total time</td>
</tr>
<tr>
<td>300</td>
<td>187723.46</td>
<td>1.305</td>
<td>0.056</td>
</tr>
<tr>
<td>350</td>
<td>170973.34</td>
<td>2.067</td>
<td>0.050</td>
</tr>
<tr>
<td>400</td>
<td>157030.46</td>
<td>1.630</td>
<td>0.012</td>
</tr>
<tr>
<td>450</td>
<td>145422.94</td>
<td>1.612</td>
<td>0.056</td>
</tr>
<tr>
<td>500</td>
<td>135467.85</td>
<td>2.344</td>
<td>0.040</td>
</tr>
</tbody>
</table>

5. Conclusion

The combined use of lagrangean/surrogate relaxation with subgradient optimization (LSSH) and with column generation (LSCG) seen to be a good approach to solve \( p \)-median clustering problems. In terms of computational performance we have noted that, for a given number of nodes (n), the most time-consuming problems for LSSH correspond to \( p = n/3 \). By other hand, for a given number of nodes, the computational performance of LSCG improves as the number of median increases. So, these two approaches have complementary computational behavior and can be put to work together. That will be important to develop decision support systems for large-scale data obtained from geographical information systems.

Acknowledgments: The authors acknowledge Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (process 99/06954-7) and Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (processes 302408/88-6 and 300837/89-5) for partial financial research support.

References


