A column generation method for the capacitated centred clustering problem

Marcos A. Pereira and Edson L. F. Senne

*aDepartment of Mathematics, Faculdade de Engenharia de Guaratinguetá, Av. Dr. Ariberto Pereira da Cunha 333, 12516-410 - Guaratinguetá - SP, Brazil, {mapereira, elfsenne}@unesp.feg.br

Keywords: location problem, clustering problem, capacitated centred clustering problem, p-median problem, lagrangean/surrogate relaxation, column generation.

Abstract. The capacitated centred clustering problem (CCCP) consists of determining a set of p clusters with maximal similarity on a network with n vertices. Demand values are associated with each vertex and each cluster has a demand capacity. The problem is well known to be \(\mathcal{NP}\)-hard and has many practical applications. In this paper, a column generation method, originally developed for capacitated p-median problems, is adapted to solve CCCP instances. The proposed algorithm uses information of the corresponding la grangean/surrogate relaxation for the problem, attempting to stabilize the column generation process. Computational results considering instances available in the literature are presented to demonstrate the effectiveness of the developed approach.

1 Introduction

Facility location problems deals with the spatial localization of emergency services, commerce depots, telecommunication networks, etc. In some applications the facilities have limited capacity. In these cases, the most commonly mathematical formulations used are the capacitated facility location problem (CFLP), capacitated p-median problem (CPMP), capacitated clustering problem (CCP) and the capacitated centred clustering problem (CCCP). The CPMP, CCP and CCCP are very similar, in the sense that: (1) each demand vertex must be allocated to exactly one facility; (2) allocation costs are based on some dissimilarity metric (e.g., distance between each pair of vertices in the CPMP); and (3) there are no fixed costs for the installation of facilities (as occurs in the CFLP).

Considering a network of \(n\) vertices with associated demand values, the CCCP (Negreiros and Palhano, 2006) consists of determining a set of \(p\) clusters of vertices with maximal similarity, with demand capacities for each cluster. If the similarity is related with some distance metric, the objective of the CCCP is to minimize the total distance between each demand vertex and the centroid of the cluster to which it is allocated.

The CCCP is \(\mathcal{NP}\)-hard, and solution techniques to solve it are based on heuristics developed for the CCP. This paper outlines the similarities of the CCCP and the CPMP, and presents a column generation algorithm for the CPMP that can be used to obtain feasible solutions for the CCCP.

2 Location and clustering problems

The objective of the CCP and the CPMP are the same: they focus on the determination of clusters that minimize the sum of distances between each vertex and the nearest facility. The demand values assigned to the vertices must be satisfied, but the facilities have limited capacity. The difference between these two problems is that in the CPMP the facilities will be located on vertices (medians) of the network, while in the CCP a facility will be installed at the geometric center (centroid) of a cluster, which not necessarily corresponds to an existing
vertex of the considered network. A location-allocation solution for each problem is presented in Figure 1.

The CPMP is a well known location problem and many solution approaches are proposed in the literature, from exact to heuristic methods. The CPMP is a clustering problem as well, since to each median is associated a subset of demand vertices, forming a cluster. However, if the median vertex is considered as a demand vertex only, the cost of any cluster can be easily calculated as the total distance between the demand vertices and the centroid of the cluster. So, some solution approaches for the CPMP can be adapted to solve the CCCP.

In this paper the column generation approach of Lorena and Senne (2004), originally developed for the CPMP, is considered to calculate solutions for CCCP instances. It is well known that the direct application of column generation methods produces a large number of columns that are not considered in the solution of the problem, stalling the process (Briant et al., 2005; Desrosiers and Lübbecke, 2005). Senne et al. (2007) use information of the lagrangean/surrogate relaxation of the corresponding problem to stabilize the column generation process. When applied to the CPMP, this technique resulted in more productive columns, accelerating the column generation method.

3 The proposed column generation method

The algorithm proposed for solving the CCCP alternates between a master problem and a column generation subproblem. Let $N$ be the set of indexes of the network vertices and $q_i$ be the demand of vertex $i \in N$. Define $S = \{S_1, ..., S_m\}$ as the set of all clusters of vertices in $N$ and let $Q$ be the demand capacity of each cluster. The cardinality of the set $S$ can be very large, so only a subset $K \subset M = \{1, ..., m\}$ is considered for practical reasons, defining a restricted master problem (RMP). The RMP corresponds to the linear relaxation of the following set covering problem:

$$v(SCP) = \text{Min} \sum_{k \in K} c_k v_k$$

subject to

$$\sum_{k \in K} A_{ik} v_k \geq 1$$

$$\sum_{k \in K} v_k = p$$

$$v_k \in [0, 1], \quad \forall k \in K$$
where:

- \( c_k = \sum_{i \in S_k} d_{ik} \), with \( d_{ik} \) being the distance between each vertex \( i \) and the centroid of the cluster \( S_k \);
- \( A = [a_{ik}] \) is a matrix with \( a_{ik} = 1 \), if vertex \( i \in S_k \), satisfying \( \sum_{i \in N} q_i a_{ik} \leq Q \), and \( a_{ik} = 0 \), otherwise;
- \( [v_k] \) is the vector of decision variables, with \( v_k = 1 \) if cluster \( S_k \) is considered in the solution and \( v_k = 0 \), otherwise.

The classical formulation for the CPMP is used as the column generation subproblem. In Lorena and Senne (2004), the lagrangean/surrogate relaxation \( L_t P_{\pi} \) is considered in a heuristic method for solving the CPMP:

\[
v(L_t P_{\pi}) = \min \sum_{i \in N} \sum_{j \in N} (d_{ij} - t \pi_i) x_{ij} + t \sum_{i \in N} \pi_i \tag{5}
\]

subject to

\[
\sum_{j \in N} x_{jj} = p \tag{6}
\]

\[
\sum_{i \in N} q_i x_{ij} \leq Q x_{jj}, \quad \forall j \in N \tag{7}
\]

\[
x_{i,j} \in \{0, 1\}, \quad \forall i \in N, \forall j \in N \tag{8}
\]

where \( d_{ij} \) is the distance between vertex \( i \) and the centroid of cluster \( j \), and \( [x_{ij}] \) is the vector of decision variables in which \( x_{ij} = 1 \) if vertex \( i \) is allocated to cluster \( j \) and \( x_{ij} = 0 \), otherwise. Note that \( x_{jj} = 1 \), if \( j \) is the centroid of a cluster and \( x_{jj} = 0 \), otherwise.

Observe that, for \( t = 1 \), \( L_t P_{\pi} \) is the traditional lagrangean relaxation with multipliers \( \pi_i \) \((i \in N)\). The lagrangean/surrogate relaxation is integrated into the column generation process by using the multipliers \( \pi_i \) \((i \in N)\) of SCP to calculate \( t \) as the optimal solution of the dual of problem \( L_t P_{\pi} \):

\[
v(D_{\pi}) = \max_{t \geq 0} v(L_t P_{\pi})
\]

In this paper, the best value for \( t \) was calculated using the search algorithm presented in Senne and Lorena (2000).

After defining a starting set of columns, SCP is solved to optimality and the corresponding dual costs \( \pi_i \) \((i \in N)\) and \( \mu \) are used to generate new columns. The column \( \left[ \frac{x_{ij}}{1} \right] \) will be added to SCP if its reduced cost is negative. To speed up the column generation process, any column \( j \in N \) satisfying:

\[
\sum_{i \in N} (d_{ij} - \pi_i) x_{ij} < |\mu|
\]

will be added to SCP (multi-pricing).
4 The column generation algorithm

Depending on the values for parameter $t$, it is possible to calculate traditional lagrangean bounds (fixing $t = 1$), or improved bounds, if $t$ is determined as solution of problem $D_\pi$. Figure 2 presents the proposed column generation algorithm $CG(t)$.

While stop conditions are not satisfied do:

1. Solve $SCP$, obtaining the dual variables $\pi_i (i \in N)$ and $\mu$;
2. Solve the dual problem $D_\pi$ to determine the best value for $t$;
3. Solve the column generation subproblem and determine the incoming columns in $SCP$, i.e., columns satisfying condition (9);
4. If no incoming columns are found in step 3, stop.
5. Add the columns $\left[ \frac{x_{ij}}{1} \right]$ found in step 3 to $SCP$.

Figure 2: The proposed $CG(t)$ algorithm.

The application of algorithm $CG(t)$ requires a starting feasible set of columns for the SCP. Such columns can be generated by randomly selecting $p$ vertices of the network and considering them as medians for the corresponding CPMP, which is solved until the first integer solution is obtained. Any resulting cluster satisfying constraint (7) is added to the RMP, and the process is repeated until the starting RMP contains at least 1,000 columns. The best feasible solution found during the generation of the initial set of columns is assumed as an upper bound to CCCP. This upper bound is kept unaltered until the end of the algorithm. A final step to improve the best feasible solution found is executed at the end of the column generation process, solving the final RMP as an integer programming problem, and considering the first integer solution found.

5 Computational results

The algorithms described in Section 4 were coded in C and executed on a microcomputer with Intel Core 2 Duo 2 GHz processor and 2 GB RAM. Solutions for the RMPs were obtained using ILOG CPLEX 10.1. The algorithm of Horowitz and Sahni (Martello and Toth, 1990) was used for solving the knapsack problems in the column generation subproblem.

The computational results presented in tables 1 and 2 refer to 13 instances considered in Lorena and Senne (2004) and Negreiros and Palhano (2006). These tables present:

- Prob: name of the test-problem;
- $n$: number of vertices in the network;
- $p$: number of facilities to be installed;
- $v$(CG): best solution value found by the algorithm $CG(t)$ or $CG(1)$;
- Iter: number of executed iterations for the algorithm $CG(t)$ or $CG(1)$;
- Cols: number of generated columns for the algorithm $CG(t)$ or $CG(1)$;
- Time: processing time (in seconds);
• \( v(\text{pCCCP}) \): best solution value found by the algorithm pCCCP;
• \( v(\text{VNS}) \): best solution value found by the algorithm VNS;
• \( \Delta P \): the relative difference \( \frac{100 \times (v(\text{CG}) - v(\text{pCCCP}))}{v(\text{pCCCP})} \);
• \( \Delta V \): the relative difference \( \frac{100 \times (v(\text{CG}) - v(\text{pCCCP}))}{v(\text{pCCCP})} \).

Table 1 compares the results obtained for algorithms CG(1) and CG(\( t \)). Considering only the usual computational aspects (number of iterations, generated columns and processing times), both algorithms performed similarly, but the columns obtained by algorithm CG(\( t \)) produced better solution values than those calculated by CG(1) for most of the considered instances.

<table>
<thead>
<tr>
<th>Prob</th>
<th>( n )</th>
<th>( p )</th>
<th>( v(\text{CG}) )</th>
<th>Iter</th>
<th>Cols</th>
<th>Time</th>
<th>( v(\text{CG}) )</th>
<th>Iter</th>
<th>Cols</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ta1</td>
<td>25</td>
<td>5</td>
<td>1470.50</td>
<td>34</td>
<td>1481</td>
<td>0.05</td>
<td>1280.49</td>
<td>36</td>
<td>1472</td>
<td>0.05</td>
</tr>
<tr>
<td>ta2</td>
<td>50</td>
<td>5</td>
<td>4474.52</td>
<td>42</td>
<td>1961</td>
<td>0.14</td>
<td>4478.15</td>
<td>42</td>
<td>1995</td>
<td>0.09</td>
</tr>
<tr>
<td>ta3</td>
<td>60</td>
<td>5</td>
<td>5357.34</td>
<td>62</td>
<td>2943</td>
<td>0.39</td>
<td>5357.34</td>
<td>54</td>
<td>2837</td>
<td>0.37</td>
</tr>
<tr>
<td>ta4</td>
<td>70</td>
<td>5</td>
<td>6240.67</td>
<td>47</td>
<td>2573</td>
<td>0.30</td>
<td>6240.67</td>
<td>43</td>
<td>2453</td>
<td>0.28</td>
</tr>
<tr>
<td>ta5</td>
<td>80</td>
<td>7</td>
<td>6029.69</td>
<td>48</td>
<td>3126</td>
<td>0.33</td>
<td>5515.46</td>
<td>51</td>
<td>3242</td>
<td>0.45</td>
</tr>
<tr>
<td>ta6</td>
<td>90</td>
<td>4</td>
<td>9040.69</td>
<td>110</td>
<td>5477</td>
<td>3.41</td>
<td>8899.05</td>
<td>105</td>
<td>5094</td>
<td>2.66</td>
</tr>
<tr>
<td>ta7</td>
<td>100</td>
<td>6</td>
<td>8195.60</td>
<td>75</td>
<td>4090</td>
<td>1.33</td>
<td>8168.36</td>
<td>70</td>
<td>3760</td>
<td>1.45</td>
</tr>
<tr>
<td>sjc1</td>
<td>100</td>
<td>10</td>
<td>17375.36</td>
<td>49</td>
<td>3605</td>
<td>0.50</td>
<td>17975.92</td>
<td>45</td>
<td>3290</td>
<td>0.56</td>
</tr>
<tr>
<td>sjc2</td>
<td>200</td>
<td>15</td>
<td>34060.73</td>
<td>75</td>
<td>8581</td>
<td>7.12</td>
<td>33357.75</td>
<td>87</td>
<td>8734</td>
<td>6.25</td>
</tr>
<tr>
<td>sjc3a</td>
<td>300</td>
<td>25</td>
<td>49032.97</td>
<td>69</td>
<td>12659</td>
<td>13.80</td>
<td>45379.69</td>
<td>78</td>
<td>13542</td>
<td>16.03</td>
</tr>
<tr>
<td>sjc3b</td>
<td>300</td>
<td>30</td>
<td>42402.73</td>
<td>66</td>
<td>12637</td>
<td>8.41</td>
<td>41185.18</td>
<td>79</td>
<td>13364</td>
<td>10.08</td>
</tr>
<tr>
<td>sjc4a</td>
<td>402</td>
<td>30</td>
<td>62729.15</td>
<td>77</td>
<td>17359</td>
<td>46.83</td>
<td>61969.06</td>
<td>85</td>
<td>17338</td>
<td>44.48</td>
</tr>
<tr>
<td>sjc4b</td>
<td>402</td>
<td>40</td>
<td>53554.43</td>
<td>64</td>
<td>14759</td>
<td>18.66</td>
<td>52989.44</td>
<td>65</td>
<td>14563</td>
<td>16.41</td>
</tr>
</tbody>
</table>

Table 1: Comparison of algorithms CG(1) and CG(\( t \)).

Table 2 compares the results obtained by algorithm CG(\( t \)) and the results of the algorithms pCCCP and VNS, presented in Negreiros and Palhano (2006), specifically developed for CCCP instances.

The negative values for \( \Delta P \) or \( \Delta V \) indicate that the algorithm CG(\( t \)) was able to find better solution values than those obtained by algorithms pCCCP or VNS, respectively. Note that algorithm CG(\( t \)) was able to find solutions for two instances that could not be solved by algorithms pCCCP or VNS. For the other instances, algorithm CG(\( t \)) improved all the solution values reported for algorithm pCCCP (a 6.10% reduction, in average) and, even marginally, it was also able to improve most of the solution values reported for algorithm VNS (a 1.43% reduction, in average).

6 Conclusions

This paper discussed a column generation method for solving CCCP instances. The proposed method uses a capacitated \( p \)-median problem as the column generation subproblem and embeds the lagrangean/surrogate relaxation to stabilize the column generation process. The computational results show that the information of the lagrangean/surrogate relaxation causes
the column generation process to generate improved columns, when compared to those generated with the traditional lagrangean relaxation. The proposed method is faster and produces better solution values than the traditional approach.

Computational results shows that the algorithm CG(t) can also improve the solution values reported by Negreiros and Palhano (2006), for most of the considered instances. The algorithm CG(t) presents better solution values for all instances solved by algorithm pCCCP and for 8 out of 11 instances solved by algorithm VNS.

Also, the proposed method can be useful in developing branch-and-price algorithms to obtain exact solutions for the CCCP.

References


<table>
<thead>
<tr>
<th>Prob</th>
<th>n</th>
<th>p</th>
<th>$v$(CG)</th>
<th>$v$(pCCCP)</th>
<th>$\Delta P$</th>
<th>$v$(VNS)</th>
<th>$\Delta V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ta1</td>
<td>25</td>
<td>5</td>
<td>1280.49</td>
<td>1528.64</td>
<td>-16.23</td>
<td>1257.51</td>
<td>1.83</td>
</tr>
<tr>
<td>ta2</td>
<td>50</td>
<td>5</td>
<td>4478.15</td>
<td>4605.03</td>
<td>-2.76</td>
<td>4485.44</td>
<td>-0.16</td>
</tr>
<tr>
<td>ta3</td>
<td>60</td>
<td>5</td>
<td>5357.34</td>
<td>5475.96</td>
<td>-2.17</td>
<td>5391.47</td>
<td>-0.63</td>
</tr>
<tr>
<td>ta4</td>
<td>70</td>
<td>5</td>
<td>6240.67</td>
<td>6292.06</td>
<td>-0.82</td>
<td>6275.99</td>
<td>-0.56</td>
</tr>
<tr>
<td>ta5</td>
<td>80</td>
<td>7</td>
<td>5515.46</td>
<td>5918.44</td>
<td>-6.81</td>
<td>5846.94</td>
<td>-5.67</td>
</tr>
<tr>
<td>ta6</td>
<td>90</td>
<td>4</td>
<td>8899.05</td>
<td>9646.38</td>
<td>-7.75</td>
<td>9134.69</td>
<td>-2.58</td>
</tr>
<tr>
<td>ta7</td>
<td>100</td>
<td>6</td>
<td>8168.36</td>
<td>8270.26</td>
<td>-1.23</td>
<td>8141.70</td>
<td>0.33</td>
</tr>
<tr>
<td>sjc1</td>
<td>100</td>
<td>10</td>
<td>17975.92</td>
<td>19429.46</td>
<td>-7.48</td>
<td>17696.53</td>
<td>1.58</td>
</tr>
<tr>
<td>sjc2</td>
<td>200</td>
<td>15</td>
<td>33357.75</td>
<td>35322.48</td>
<td>-5.56</td>
<td>33423.84</td>
<td>-0.20</td>
</tr>
<tr>
<td>sjc3a</td>
<td>300</td>
<td>25</td>
<td>45379.69</td>
<td>50254.51</td>
<td>-9.70</td>
<td>47985.29</td>
<td>-5.43</td>
</tr>
<tr>
<td>sjc3b</td>
<td>300</td>
<td>30</td>
<td>41185.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sjc4a</td>
<td>402</td>
<td>30</td>
<td>61969.06</td>
<td>76352.45</td>
<td>-18.84</td>
<td>66689.96</td>
<td>-7.08</td>
</tr>
<tr>
<td>sjc4b</td>
<td>402</td>
<td>40</td>
<td>52989.44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Comparison of algorithms CG(t), pCCCP and VNS.