Matheuristics for the capacitated p-median problem

Fernando Stefanello\textsuperscript{a}, Olinto C. B. de Araújo\textsuperscript{b,\textsuperscript{*}}, Felipe M. Müller\textsuperscript{c}

\textsuperscript{a}Instituto de Informática - Universidade Federal do Rio Grande do Sul - Porto Alegre - RS - Brasil
\textsuperscript{b}Colégio Técnico Industrial de Santa Maria - Universidade Federal de Santa Maria - Santa Maria - RS - Brasil
\textsuperscript{c}Programa de Pós-Graduação em Informática - Universidade Federal de Santa Maria - Santa Maria - RS - Brasil

Abstract

The recent evolution of computers and mathematical programming techniques has provide the development of a new class of algorithms called matheuristics. Associated with an improvement of MIP solvers, many of these methods have been successful applied to solve combinatorial problems. This work presents an approach that hybridizes metaheuristics based on local search and exact algorithms to solve the classic capacitated p-median problem. The proposal considers a reduced mathematical model performed by a heuristic elimination of variables which are less likely to belong to a good or optimal solution. In addition, an iterative local optimization algorithm based on reduction is proposed. In both cases, the mathematical model is solved by a MIP solver. Computational experiments involving three set of instances are presented to demonstrate the good performance of the developed approaches.

Keywords: capacitated p-median problem, matheuristic algorithm, size-reduction

1. Introduction

Capacitated clustering problems have many practical applications, such as facility location, hospitals, schools and garbage collection, among others. In all these cases, given a set of nodes associated to the reality to be modeled, the goal is to group them to maximize the dissimilarity among the clusters. More specifically, we deal with the Capacitated p-Median Problem (CPMP). The CPMP consists of allocating $p$ facilities (medians) to serve $n$ demand points (nodes). The objective is to minimize the sum of the distances among the nodes and the medians, with the constraint that the total demand of the nodes assigned to each median does not exceed its given capacity.

*Corresponding author

Email addresses: stefanello@inf.ufsm.br (Fernando Stefanello), olinto@ctism.ufsm.br (Olinto C. B. de Araújo), felipe@inf.ufsm.br (Felipe M. Müller)

URL: http://www.inf.ufsm.br/~stefanello (Fernando Stefanello)

July 16, 2012
The CPMP is known to be NP-hard (Garey and Johnson, 1979). In the literature, we find several approaches using heuristic methods to find good quality solutions on acceptable computational time.


The evolution of computers in the last years, characterized by systematic increasing processing power and advances in parallelization techniques, has allowed the resolution of difficult computational problems in a shorter time if compared to computers from a few years ago. Added to this scenario, we had the improvement on the performance of mixed integer programming solvers, whose efficiency cannot be disregarded by the scientific community. Furthermore, the complexity theory provides a overly pessimistic run time bounds for NP-hard problem. But Nievergelt (2000) cites that the fact that many instances of this class of problems, including the practical ones, can be solved efficiently is often ignored. This scenario is propitious to the development of methods that hybridizes exact and heuristic algorithms to incorporate into single framework similarities with the best features of each of these two approaches, guaranteeing to find the good solution and low computational cost, respectively. Talbi (2002), Dumitrescu and Stützle (2003), Puchinger and Raidl (2005), Fernandes and Lourenço (2006) and Jourdan et al. (2009) presented different taxonomies for the hybrid methods as a survey of published studies which use this technique.

This work proposes a size-reduction mathematical models to handle CPMP. Similar methods have been proposed by Mautor and Michelon (1997, 2001) for the quadratic assignment problem and by Fanjul-Peyro and Ruiz (2011) for the unrelated parallel machine scheduling problem. Furthermore, we present an iterative local optimization algorithm (LOPT) similar to that in Taillard and Voss (2001).

The paper is organized as follows: Section 2 provides a formal description of CPMP; Section 3 presents two strategies to reduce variables from the original mathematical model; Section 4 presents the local optimization algorithm to solve large size instances. Section 5 brings extensive computational experiments performed in three different sets of instances; finally conclusions and further work are presented in Section 6.
2. Mathematical formulation

The capacitated p-median problem can be represented as an indirect graph \( G = (V, E) \) where \( V = \{1, \ldots, n\} \) represents the set of nodes or customers and \( E \) represents the edges usually called distance or cost between two pairs of nodes. Let \( M = \{1, \ldots, m\} \) the set of candidate medians or facilities. Each node \( v \in V \) is associated with a demand \( q_i \) and each candidate median \( m \in M \) with a capacity \( Q_i \) which must be respected. Without loss of generality, we can assume that \( M = V \) and \( n = m \). The goal is to find a subset of nodes \( M_p \subseteq M \) and \( |M_p| = p \) to be medians and to assign each node \( v \in V \) at only one median \( p \in M_p \) such that the sum of the distance \( d_{ij} \) between each node \( i \in V \) and its median \( j \in M_p \) is minimized.

Let \( x_{ij} = 1 \) if customer \( i \in V \) is assigned to a median \( j \in M \), 0 otherwise. Thus, the CPMP can be formulated mathematically as follows:

\[
\begin{align*}
\text{z} &= \min \sum_{i \in V} \sum_{j \in M} d_{ij} x_{ij} \\
\text{Subject to:} & \\
\sum_{j \in M} x_{jj} &= p, \\
\sum_{j \in M} x_{ij} &= 1 \quad \forall i \in V, \\
x_{ij} &\leq x_{jj} \quad \forall i \in V, \forall j \in M, \\
\sum_{i \in V} q_i x_{ij} &\leq Q_j x_{jj} \quad \forall j \in M, \\
x_{ij} &\in \{0, 1\} \quad \forall i \in V, \forall j \in M.
\end{align*}
\]

Constraints (2) ensures the existence of exactly \( p \) clusters and constraints (3) and (4) requires each node to be assigned to only one median. Constraints (5) limit the capacity of medians and constraints (6) define the domains of the variables.

Constraints (4) are redundant, but it is known that these constraints improve the performance of methods based on the relaxation of integer constraints.

3. Size-reduction heuristics

Several exact and heuristic methods are proposed in literature for solving CPMP. Although heuristic methods provide the possibility of solving large instances, there is no guarantee that the solution is optimal. On the other hand, exact methods have this property, but their application is limited only to small and medium-sized instances due to the inherent complexity of combinatorial optimization problems.

Even considering that nowadays the resolution of combinatorial optimization problems by MIP solvers is an increasingly viable option, for large instances it becomes impracticable due to the large number of variables involved. However, evaluating some characteristics of the problem, it is possible to observe that certain variables in the mathematical model have
a very low probability of belonging to a good solution. Such variables can be strategically eliminated by a process referred in this work as Size-Reduction Variables. The mathematical model resulting from this strategy is called Reduced Model, while the mathematical model containing all variables is called Full Model.

In the following subsections we present two size-reduction heuristics to eliminate variables of the mathematical model.

3.1. Demand size-reduction (R1)

The first strategy considers a reduction of variables $x_{ij}$ from a candidate median $j \in M$ such that the nodes $j$ do not belong to the set of closest nodes to $i$. Formally, we define the subset $K_i \subseteq V$ of closest $k \in V$ nodes of candidate median $i$ where $K_i = \{k \in V/ \sum q_k \leq \alpha Q_i - q_i\}$, \forall $i \in M$. We preserve the variable $x_{ij}$ such that $j \in M$ and $j \in K_i$, i.e, if a candidate median $j \notin K_i$, then $x_{ij}$ is disposed. Also, if there is a node $i$ such that $q_i > Q_j - q_j$, then the variable $x_{ij}$ is disposed. The parameter $\alpha$ is an expand capacity factor used to control the number of variables considered in the model.

This methodology provides a significant reduction in the number of variables and, although not guaranteeing optimality, allows handling larger problems.

Figure 1 illustrates the proposed reduction, indicating which items are likely to be associated with two candidate medians for the instance sjc1 proposed by Lorena and Senne (2004). In this figure, (a) and (b) shows the representation of the remaining variables for two nodes and the optimal solution for this instance, respectively. We see that the reduction does not eliminate the variables to the optimal solution.

Figure 1: Comparison between remaining variables and optimal solution. (a) Graphical representation of the size-reduction R1 for two nodes; (b) Graphical representation of the optimal solution.

3.2. Neighborhood median size-reduction (R2)

This reduction starts from an initial solution and allows variables that define a node as median candidate only in adjacent regions to the current median. More specifically, only the
\( \beta \) nodes nearest to a median in current solution are considered as candidate median, where \( \beta > 0 \) is the integer parameter which defines the scope of the reduction method.

The problem is restricted to the assignment problem for \( \beta = 1 \). When the number of nodes in a specific cluster is smaller than \( \beta \), then all nodes in the cluster are considered candidate medians.

Figure 2 shows a representation of an initial solution for the instance \( u724_010 \) described in Section 5. The filled circles corresponding to candidate medians (\( \beta = 10 \)) and the nodes depicted as a white square cannot be medians.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{graph.png}
\caption{Graphical representation of the size-reduction \( R2 \).}
\end{figure}

### 4. Local optimization algorithm based on reductions

In this section we proposed an iterative local optimization algorithm based on reductions \( R1 \) and \( R2 \) to solve the capacitated p-median problem by a MIP solver. This approach is very similar to metaheuristic POPMUSIC propose in Taillard and Voss (2001).

After some analysis between several good solutions, we observed that most clusters remain unchanged when there is an improvement in the objective function value. This fact suggests the possibility of reducing the size of the problem from an iterative exploration of sub-regions most likely to improve the objective function value. Thus, in the approach described in this section we iteratively select sub-regions and optimize it, until explore all sub-regions of the problem.

In order to define the sub-region to be explored it is necessary to define a base cluster with median \( m' \). After, we select a subset of \( nc \) clusters closest \( m' \) to be optimized (free clusters). The number of free clusters should be large enough to explore a good area but, at the same time, small enough to allow the solver to optimize them. We defined this number by \( nc = max(\lfloor \frac{p}{n} \omega \rfloor, C_{lm}) \), where \( \omega \) is the number of nodes that the solver can handle and \( C_{lm} \) is the minimum number of clusters to be analyzed (\( C_{lm} = 5 \) as default). When \( nc \) is equal to \( C_{lm} \), the number of nodes to be analyzed can be large, but the combined use with reduction methods \( R1 \) and \( R2 \) tends to minimize this matter.
We see this combination in Figure 3 (b) in which the number of variables, represented by lines, is significantly small in comparison to Figure 3 (a), which has all the variables that represent possible assignments.

![Figure 3: First iteration of local optimization algorithm. (a) Representation of sub-region without size-reduction variables; (b) Representation of sub-region with size-reduction R1 and R2 combined.](image)

Also, the Figure 3 shows the sub-region selected in the first iteration of local optimization algorithm, where the dots on the filled square in this sub-region represent the nodes that belong to free clusters. The remaining nodes belong to fixed nodes.

The algorithm described the local optimization algorithm used to solve the capacitated p-median problem.

**Algorithm 1**: Local optimization algorithm

In the Step 1 we choose between the node with smaller and larger sum of coordinates \(x\) and \(y\) of all nodes \(v \in V\). The choice by fixed nodes ensures a deterministic search procedure. In Step 4 we apply the size-reduction \(R1\) and \(R2\) before solve the mathematical model. Step 7 ensures that the solution found by the algorithm is a local optimum in the considered neighborhood. Due to the characteristics of instances, the time consumed in
Step 4 can vary significantly. To overcome this situation the parameter \( \tau \), that delimits the maximum time spent in the Step 4, is defined.

The solver presents different behaviors for different instances, thus we define dynamically and self-adjusting the parameter \( \omega \). For the first iteration we set the parameter \( \omega_0 = \omega \). To the following iteration defined as \( \omega_{i+1} = \omega - (2^{i+1} - 1)\omega_0 0.1e^{-5(t_i/\omega)^2} \) where \( t_i \) is the time to solve the iteration \( i \).

5. Computational results

The computational results presented were obtained using CPLEX 12.3\(^1\) with default configuration on an Intel i5-2300 2.80GHz 4GB RAM with Ubuntu 10.10.

Three sets of instances were used.

Set 1 was proposed by Lorena and Senne (2004)\(^2\) and comprises six real instances named \( sjc1 \) to \( sjc6 \) and obtained from a geographic database from São José dos Campos city, SP-Brazil.

We also used four instances described by Díaz and Fernández (2006), containing 737 nodes to 74 and 148 medians and two different vectors of demand, representing cities in Spain. This is considered set 2 and the number of nodes and medians are coded in the instances names. For example, we have “spain737.74.1”, where 737 indicates the number of nodes representing cities in Spain; the number 74 indicates the number of medians and 1 indicates the vector of demands used. This set of instances uses a non-Euclidean distance matrix.

The third set of instances includes five large instances proposed by Lorena et al. (2003), named \( p3038.600 \) to \( p3038.1000 \), originated from one instance with 3038 nodes from the TSP-LIB\(^3\), which was adapted to include 600, 700, 800, 900 and 1000 clusters.

All sets of instances and their best known solutions are referenced or available at \( \text{http://www.inf.ufsm.br/~stefanello/instances/} \).

The initial solution was built based on primal heuristic proposed in Mulvey and Beck (1984). The difference between these heuristics is that the order to select a node to assign to a median is completely random. Also, we iterate this heuristic for a fixed number of iterations.

The resolution procedure was divided into three phases. In the phase 1, we built an initial solution. Phase 2 starts from the solution found in phase 1 and solves the mathematical model with size-reductions \( R1 \) and \( R2 \). Phase 3 starts from the solution at the previous stage and uses the local optimization algorithm based on reductions \( R1 \) and \( R2 \). The parameters and their values in each phase are presented in Table 1. These parameters were defined after several computational tests with different configurations and were chosen because they provide a better tradeoff between solution quality and computational time.

---

\(^1\) \url{http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/}

\(^2\) Available at \url{http://www.lac.inpe.br/lorena/instancias.html}

\(^3\) \url{http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/}
Table 1: Summary of the parameters used in the three phases of resolution.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Phase1</th>
<th>Phase2</th>
<th>Phase3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>1000</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>α</td>
<td>-</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>β</td>
<td>-</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>τ</td>
<td>-</td>
<td>0.7n</td>
<td>30</td>
</tr>
<tr>
<td>ω</td>
<td>-</td>
<td>-</td>
<td>300</td>
</tr>
</tbody>
</table>

We used the phase 1 and 2 without the reduction $R2$ when $n < 500$, and the phase 3 is applied only if the solver is not able to prove optimality. Also, by default we used reduction $R2$ when $n/p > 10$.

In phase 2 the models to solve can be significantly large. This phase could solve or improve the solution in all sets of instances cited in this work. However, in some cases the solver needed large amount of memory. To avoid early closure of the solver or using a large amount of memory we used tree memory solver parameter equal to 2000.

Due to random choices in phase 1 we ran the method 10 times for each instance to better evaluate the performance in relation to objective function and runtime.

5.1. Results for instance set 1

In the first experiment we compare the results for reduced and full model. In the Table 2 we present the instances, number of nodes and medians. For the full model we show the objective function ($z$) and run time in seconds to solve each instance in our computer. Also, we show the objective function ($z^*$) to the reduced model with default configuration ($\alpha = 2$) and a slightly higher value of alfa ($\alpha = 2.2$). In these instances, the procedure is deterministic but the runtime change with the solution found in phase 1, and then we show the average runtime of ten runs.

<table>
<thead>
<tr>
<th>Instance</th>
<th>n</th>
<th>p</th>
<th>Full Model</th>
<th>Reduced Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>sjc1</td>
<td>100</td>
<td>10</td>
<td>17,288.99</td>
<td>17,288.99</td>
</tr>
<tr>
<td>sjc2</td>
<td>200</td>
<td>15</td>
<td>33,270.94</td>
<td>33,270.94</td>
</tr>
<tr>
<td>sjc3a</td>
<td>300</td>
<td>25</td>
<td>45,335.16</td>
<td>45,335.16</td>
</tr>
<tr>
<td>sjc3b</td>
<td>300</td>
<td>30</td>
<td>40,635.90</td>
<td>40,635.90</td>
</tr>
<tr>
<td>sjc4a</td>
<td>402</td>
<td>30</td>
<td>61,925.51</td>
<td>61,925.51</td>
</tr>
<tr>
<td>sjc4b</td>
<td>402</td>
<td>40</td>
<td>52,458.02</td>
<td>52,458.02</td>
</tr>
</tbody>
</table>

Compared with the full model, we were able to decrease significantly the computational time setting $\alpha = 2$ for the reduced model. Using $\alpha = 2.2$ we find all optimal solutions with small changes in average runtime.

Table 3 provides the best results in the literature in recent years for this set of instances. The first columns give the identification of the instance. For each approach, we show the objective function and the computational time, respectively. The first approach SS is a hybrid metaheuristic that combines Scatter Search with Path Relinking proposed by Scheuerer and Wendolsky (2006). The approach VNS described in Fleszar and Hindi (2008) uses Variable Neighborhood Search combined with CPLEX version 10.1. The third approach is described
by Chaves et al. (2007) and named CS. Fen-Cplex is proposed by Boccia et al. (2008) based on Fenchel cuts. Finally, we show the results obtained using the phase 1 and 2 with $\alpha = 2.2$.

### Table 3: Performance comparison with the approaches of the literature.

<table>
<thead>
<tr>
<th>ID</th>
<th>n</th>
<th>$p$</th>
<th>$Z$ (s)</th>
<th>Time (s)</th>
<th>$Z$ (s)</th>
<th>Time (s)</th>
<th>$Z$ (s)</th>
<th>Time (s)</th>
<th>$Z$ (s)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sjc1</td>
<td>100</td>
<td>10</td>
<td>17,288.99</td>
<td>60.00</td>
<td>17,288.99</td>
<td>50.50</td>
<td>17,288.99</td>
<td>22.72</td>
<td>17,288.99</td>
<td>1.27</td>
</tr>
<tr>
<td>sjc2</td>
<td>200</td>
<td>15</td>
<td>33,293.40</td>
<td>600.00</td>
<td>33,270.94</td>
<td>44.08</td>
<td>33,270.94</td>
<td>112.81</td>
<td>33,270.94</td>
<td>2.19</td>
</tr>
<tr>
<td>sjc3a</td>
<td>300</td>
<td>25</td>
<td>45,338.02</td>
<td>2,307.00</td>
<td>45,335.16</td>
<td>8,580.30</td>
<td>45,335.16</td>
<td>940.75</td>
<td>45,335.16</td>
<td>23.22</td>
</tr>
<tr>
<td>sjc3b</td>
<td>300</td>
<td>30</td>
<td>40,635.90</td>
<td>2,308.00</td>
<td>40,635.90</td>
<td>2,292.86</td>
<td>40,635.90</td>
<td>1,887.97</td>
<td>40,635.90</td>
<td>2.02</td>
</tr>
<tr>
<td>sjc4a</td>
<td>402</td>
<td>30</td>
<td>61,925.52</td>
<td>6,109.00</td>
<td>61,925.51</td>
<td>4,221.47</td>
<td>61,928.72</td>
<td>2,885.11</td>
<td>61,925.51</td>
<td>52.56</td>
</tr>
<tr>
<td>sjc4b</td>
<td>402</td>
<td>40</td>
<td>52,531.46</td>
<td>6,106.00</td>
<td>52,469.96</td>
<td>3,471.44</td>
<td>52,531.27</td>
<td>7,626.33</td>
<td>52,458.02</td>
<td>5.57</td>
</tr>
</tbody>
</table>

* Intel Celeron 2.2 GHz  
* Pentium IV 3.2 GHz and CPLEX 10.1  
* Pentium IV 3.02 GHz  
* ASUS S5 notebook 1.6 GHz and CPLEX 9.0  
* Intel i5-2300 2.80GHz and CPLEX 12.3

The table shows that the computational time of reduced model is significantly smaller than the other approaches, despite the different hardware platforms. The reduced model with $\alpha = 2.2$ find the optimal solutions for all instances of the set.

### 5.2. Results for instance set 2

Table 4 shows the results for the second set of instances. The value of the objective function of the solution found in the second phase and the total computational time are shown in the fourth and fifth columns, respectively. The value of objective function to the best solution found and average, maximum, standard deviation and average of computational time in phase 3 are shown in the last columns of this table. Furthermore, the first column gives the instance name followed by the objective function values and computational times reported by Díaz and Fernández (2006).

### Table 4: Computational results for instance set 2.

<table>
<thead>
<tr>
<th>Instance</th>
<th>SS-PR</th>
<th>Phase 2</th>
<th>Phase 3</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ss-PR</td>
<td></td>
<td>Z</td>
<td>Time(s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spain</td>
<td></td>
<td>8,967</td>
<td>9,717.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spain2</td>
<td></td>
<td>8,970</td>
<td>57,239.12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spain3</td>
<td></td>
<td>6,012</td>
<td>43,361.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spain4</td>
<td></td>
<td>6,009</td>
<td>17,714.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Sun Blade 1000/750  
* Intel i5-2300 2.80GHz and CPLEX 12.3

We observed that the computing time for instances with 148 medians were smaller than the other two instances with 74 medians. That is due to the fact that the phase 2 obtains good results in the same time, showing more favorable in instances with ratio $p/n$ low.

In comparison with the reported by Díaz and Fernández (2006), we have improved the results for all instances. Solutions found in the first round were already better for instances with 148 medians. Although the data reported are from different hardware platforms, we can infer that the difference between computational time of the reduced model and SS-PR is representative.
5.3. Results for instance set 3

The Table 5 details the average of 10 executions of the phase 1, 2 and 3 for the third set of instances. The first columns give the instance name, the number of nodes and the number of medians, respectively. The column generation algorithm proposed by Lorena and Senne (2004) shows the values and the computational time to compute the lower bound to this set of instances. The minimum, average, maximum, and standard deviation values of objective function, plus an average runtime in seconds to compute all phases of our procedure are shown in the last columns.

<table>
<thead>
<tr>
<th>Instance</th>
<th>n</th>
<th>p</th>
<th>Lower Bound</th>
<th>Total Time (s)</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>SD</th>
<th>Avg (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3038_600</td>
<td>3038</td>
<td>600</td>
<td>122,023.66</td>
<td>59,993.02</td>
<td>122,711.17</td>
<td>122,724.79</td>
<td>122,743.15</td>
<td>11.44</td>
<td>2,685.38</td>
</tr>
<tr>
<td>p3038_700</td>
<td>3038</td>
<td>700</td>
<td>108,685.59</td>
<td>46,705.53</td>
<td>109,677.30</td>
<td>109,695.61</td>
<td>109,729.94</td>
<td>17.04</td>
<td>2,239.84</td>
</tr>
<tr>
<td>p3038_800</td>
<td>3038</td>
<td>800</td>
<td>98,530.99</td>
<td>33,844.27</td>
<td>100,064.94</td>
<td>100,084.41</td>
<td>100,105.03</td>
<td>13.05</td>
<td>2,819.26</td>
</tr>
<tr>
<td>p3038_900</td>
<td>3038</td>
<td>900</td>
<td>90,239.65</td>
<td>25,306.54</td>
<td>92,310.09</td>
<td>92,317.78</td>
<td>92,335.74</td>
<td>9.18</td>
<td>1,578.17</td>
</tr>
<tr>
<td>p3038_1000</td>
<td>3038</td>
<td>1000</td>
<td>83,231.58</td>
<td>20,210.25</td>
<td>85,854.05</td>
<td>85,856.85</td>
<td>85,864.74</td>
<td>3.02</td>
<td>1,874.08</td>
</tr>
</tbody>
</table>

We can observe that the average objective function value is very close to the best solutions found shown in column Min, maintaining relatively low values of standard deviation.

Regarding the average computational time, we can conclude that the proposed procedure is efficient, despite the hardware configurations. Compared with the lower bound provided by the Column Generation approach, the smallest gap is 0.57% for instance p3038_600 and the largest gap is 3.15% for instance p3038_1000.

6. Conclusions

In this work we proposed strategies to eliminate heuristically variables from the CPMP mathematical model and a iterative local optimization algorithm based on reduction solved by a MIP solver.

The computational experiments were performed on three set of instances in order to validate the performance of the proposed methods, which enabled the resolution of large instances. Concerning the instances of small and medium sizes, for which current state-of-art MIP solvers find the optimal solution for the vast majority of instances using the full model, the reduced model provides a small computational time with small loss in solution quality over the full model. The strategies proposed in this work were very efficient for instance’s set 1 because they were able to find the optimal solutions in much less computational time than the best methods from the literature. Regarding the instance’s set 2, we not only found solutions with better objective function value but also obtained lower computational time. The optimization algorithm provided the resolution of large instances with an amount of nodes that had not yet been addressed in the literature.

As a future work, we’ll replace the phase 2 by a fast local search to solve large scale instances. Also, we plan creating a new set of instances to evaluate our approach.
Acknowledgements

The first author was partially supported by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), Brazil. We also thanks the anonymous referees for their valuable comments and suggestions to improve this paper.

References


