International Journal of Operations Research

International Journal of Operations Research Vol. 4, No. 3, 172-180 (2007)

A Simple Stabilizing Method for Column Generation Heuristics: An Application to *P*-Median Location Problems

Edson L. F. Senne[1,*], Luiz A. N. Lorena[2], and Marcos A. Pereira[1]

[1]FEG/UNESP – São Paulo State University, 12516-410 – P.O. Box 205 – Guaratinguetá, SP – Brazil

[2]LAC/INPE – Brazilian Space Research Institute, 12.201-970 – P.O. Box 515 – São José dos Campos, SP –Brazil

Received May 2006; *Revised September 2006*; *Accepted January 2007*

Abstract—The Lagrangean/surrogate relaxation has been explored as a faster computational alternative to traditional Lagrangean heuristics. In this work the Lagrangean/surrogate relaxation and traditional column generation approaches are combined in order to accelerate and stabilize primal and dual bounds, through an improved reduced cost selection. The Lagrangean/surrogate multiplier modifies the reduced cost criterion, resulting in the selection of more productive columns for the *p*-median problem, which deals with the localization of *p* facilities (medians) on a network in order to minimize the sum of all the distances from each demand point to its nearest facility. Computational tests running *p*-median instances taken from the literature are presented.

Keywords—*P*-median, Location, Column generation, Large-scale optimization, Integer programming

1. INTRODUCTION

This work describes the use of the Lagrangean/ surrogate relaxation as a stabilizing method for the column generation process for linear programming problems. The Lagrangean/surrogate relaxation uses the local information of a surrogate constraint relaxed in the Lagrangean way, and has been used to accelerate subgradient-like methods. A local search is conducted at some initial iteration of subgradient methods, adjusting the step sizes. The reduction of computational times can be substantial for large-scale problems (Narciso and Lorena (1999), Senne and Lorena (2000)).

Column generation is a powerful tool for solving large-scale linear programming problems that arise when the columns of the problem are not known in advance and a complete enumeration of all columns is not an option, or the problem is rewritten using Dantzig-Wolfe decomposition (Dantzig and Wolfe (1960)). Column generation is a natural choice in several applications, such as the well-known cutting-stock problem, vehicle routing and crew scheduling (Gilmore and Gomory (1961), Gilmore and Gomory (1963), Desrochers and Soumis (1989), Desrochers et al. (1992), Vance (1993), Vance et al. (1994), Day and Ryan (1997), Valério de Carvalho (1999)).

In a classical column generation process, the algorithm iterates between a restricted master problem and a column generation subproblem. Solving the master problem yields a dual solution, which is used to update the cost coefficients for the subproblem that can produce new incoming columns.

The equivalence between Dantzig-Wolfe decomposition, column generation and Lagrangean relaxation optimization is well known. Solving a linear programming by Dantzig-Wolfe decomposition is equivalent to solving the Lagrangean dual by Kelley's cutting plane method (Kelley (1960)). However, in many cases a straightforward application of column generation may result in slow convergence. This paper shows how to use the Lagrangean/surrogate relaxation to accelerate the column generation process, generating new productive sets of columns at each iteration of the algorithm.

Other attempts to stabilize dual solutions have appeared before, like the Boxstep method (Marsten et al. (1975)), where the optimization in the dual space is explicitly restricted to a bounded region with the current dual solution as the central point. Bundle methods (Neame (1999)) define a trust region combined with penalties to prevent significant changes between consecutive dual solutions. The Analytic Center Cutting Plane method (du Merle et al. (1998)) considers the current analytic center of the dual function in the next iteration, instead of assuming the optimal dual solution, avoiding drastic oscillations on the dual multipliers. Other recent alternative methods to stabilize dual solutions have been considered in du Merle et al. (1999). See also Lübbecke and Desrosiers (2002) for selected topics in column generation.

The search for *p*-median nodes on a network with n vertices is a classical location problem. The objective is to locate $p < n$ facilities (medians) such that the sum of the distances from each demand point to its nearest facility is minimized. The problem is well known to be *NP*-hard and several heuristics have been developed for *p*-median

* Corresponding author's email: elfsenne@feg.unesp.br

problems. The combined use of Lagrangean/surrogate relaxation and subgradient optimization in a primal-dual viewpoint revealed to be a good solution approach to the problem (Senne and Lorena (2000)).

Initial attempts of using column generation to solve *p*-median problems appear in Garfinkel et al. (1974) and Swain (1974). These authors report convergence problems for instances where the ratio p/n tends to 0. This observation was also confirmed later in Galvão (1981). The solution of large-scale instances using a stabilized approach is reported in du Merle et al. (1999). The use of Lagangean/surrogate as an alternative to stabilize the column generation process applied to capacitated *p*-median problems appeared in Lorena and Senne (2004).

In this paper, the use of Lagrangean/surrogate relaxation as a simple, but effective, stabilization method for the column generation technique to the *p*-median problem is presented. The paper is organized as follows. Section 2 presents *p*-median formulations and the traditional column generation process. The next section presents the Lagrangean/surrogate relaxation and how it can be used in conjunction with the column generation process. Section 4 presents the algorithms and the next section shows some computational results evidencing the benefits of the new approach.

2. *P*-MEDIAN FORMULATIONS AND COLUMN GENERATION

The *p*-median problem considered in this paper can be formulated as the following binary integer programming problem:

$$Pmed: v(Pmed) = Min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

$$\text{subject to } \sum_{j=1}^{n} x_{ij} = 1, \textit{ for } i \in N \tag{1}$$

$$\sum_{j=1}^{n} x_{jj} = p \tag{2}$$

$$x_{ij} \leq x_{jj}, \textit{ for } i, j \in N \tag{3}$$

$$x_{ij} \in \{0, 1\}, \textit{ for } i, j \in N \tag{4}$$

where

- n is the number of nodes in the network and $N = \{1, ..., n\}$;
- p is the number of facilities (medians) to be located;
- $[d_{ij}]_{n\times n}$ is a symmetric cost (distance) matrix, with $d_{ii} = 0$, for $i \in N$;
- $[x_{ij}]_{n\times n}$ is the allocation matrix, with $x_{ij} = 1$ if node i is assigned to median j, and $x_{ij} = 0$, otherwise; $x_{jj} = 1$ if node j is a median and $x_{jj} = 0$, otherwise.

Constraints (1) and (3) ensure that each node i is allocated to only one node j, which must be a median. Constraint (2) determines that exact p nodes must be selected for the localization of the medians and (4) gives the integer conditions. Any feasible solution for *Pmed* partitions the set N into p disjoint subsets, each one defining a cluster containing one median and the nodes allocated to it.

Pmed is a classical formulation and has been explored in other papers. Garfinkel et al. (1974) and Swain (1974) applied the Dantzig-Wolfe decomposition to *Pmed* obtaining the following set partition problem with cardinality constraint:

$$SP\text{-}Pmed: \ v(SP\text{-}Pmed) = Min \sum_{k=1}^{m} c_k y_k$$

$$\text{subject to } \sum_{k=1}^{m} A_k y_k = 1 \tag{5}$$

$$\sum_{k=1}^{m} y_k = p, \quad y_k \in \{0, 1\} \tag{6}$$

where

- $S = \{S_1, S_2, ..., S_m\}$, is the set of all subsets of N,
- $M = \{1, 2, ..., m\}$,
- $A_k = [a_i]_{n\times 1}$, for $k \in M$; with $a_i = 1$ if $i \in S_k$, and $a_i = 0$ otherwise,
- $c_k = \underset{i \in S_k}{Min} (\sum_{j \in S_k} d_{ij})$, for $k \in M$, and
- y_k is the decision variable, with $y_k = 1$ if the subset S_k is selected, and $y_k = 0$ otherwise.

For each subset S_k, the median node is decided when the cost c_k is calculated. So, the columns of *SP-Pmed* implicitly consider the constraints set (3) in *Pmed*. Constraints (1) and (2) are conserved and respectively updated to (5) and (6), according the Dantzig-Wolfe decomposition principle. The same formulation is found in Minoux (1987).

The cardinality of M can be huge, so a partial set of columns $K \subset M$ is considered instead. In this case, problem *SP-Pmed* is also known as the restricted master problem in the column generation context (Barnhart et al. (1998)).

The search for exact solutions of *SP-Pmed* is not the objective of this paper. So, the problem to be solved by column generation is the linear programming relaxation of the corresponding set covering formulation for *Pmed*, as follows:

$$SC\text{-}Pmed: \ v(SC\text{-}Pmed) = Min \sum_{k=1}^{m} c_k y_k$$

$$\text{subject to } \sum_{k=1}^{m} A_k y_k \geq 1 \tag{7}$$

$$\sum_{k=1}^{m} y_k = p, \quad y_k \in [0, 1] \tag{8}$$

Problem *SC-Pmed* is a relaxed version of *SP-Pmed*, so $v(SC\text{-}Pmed) \leq v(SP\text{-}Pmed)$. But problem *SC-Pmed* is easier to be solved than *SP-Pmed*.

After defining an initial pool of columns, problem *SC-Pmed* is solved and the final dual costs μ_i (i = 1, ..., n)

and ρ are used to generate new columns $\alpha_j = [\alpha_{ij}]_{n\times 1}$ as solutions of the following subproblem:

$$SubPmed:\ v(SubPmed) = \underset{j\in N}{Min}\left[\underset{\alpha_{ij}\in\{0,1\}}{Min}\sum_{i=1}^{n}(d_{ij}-\mu_i)\alpha_{ij}\right]$$

SubPmed is easily solved, considering each $j \in N$ as a median node, and setting $\alpha_{ij} = 1$, if $(d_{ij} - \mu_i) \le 0$ and $\alpha_{ij} = 0$, if $(d_{ij} - \mu_i) > 0$. The new sets S_j are defined as $\{i \mid \alpha_{ij} = 1$ in $SubPmed\}$.

The reduced cost is $rc = v(SubPmed) - \rho$ and $rc < 0$ is the condition for incoming columns. Let j^* be the node index reaching the overall minimum for $v(SubPmed)$. Then, the column $\left[\frac{\alpha_{j^*}}{1}\right]$ is added to *SC-Pmed* if $v(SubPmed) < \rho$. But it is well known (Barnhart (1998)) that every column $\left[\frac{\alpha_j}{1}\right]$ $(j = 1, ..., n)$ satisfying:

$$\left[\underset{\alpha_{ij}\in\{0,1\}}{Min}\sum_{i=1}^{n}(d_{ij}-\mu_i)\alpha_{ij}\right] < \rho, \qquad (9)$$

can be added to the pool of columns, possibly accelerating the column generation process.

3. LAGRANGEAN/SURROGATE AND COLUMN GENERATION

The Lagrangean relaxation for problem *Pmed* is:

$$L_{\pi,\lambda}Pmed:\ v(L_{\pi,\lambda},\ Pmed) = Min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n}(d_{ij}-\pi_i)x_{ij} + \lambda\left(\sum_{j=1}^{n}x_{jj}-p\right)+\sum_{i=1}^{n}\pi_i\right\}$$

subject to (3) and (4)

where $\pi \in R^n$ and $\lambda \in R$ are the Lagrangean multipliers of constraints (1) and (2), respectively.

Solving $L_{\pi,\lambda}Pmed$ generates new cutting planes on the Kelley's method. If $\mu \in R_+^n$ and $\rho \in R$ are dual variables associated to constraints (7) and (8) of *SC-Pmed*, respectively, this is equivalent to finding the column j^* solving subproblem *SubPmed*. The column $\left[\frac{\alpha_{j^*}}{1}\right]$, as well as all the corresponding columns $\left[\frac{\alpha_j}{1}\right]$ satisfying expression (9), can be added to *SC-Pmed*.

The Lagrangean/surrogate relaxation for the *p*-median problem was presented in Senne and Lorena (2000). As the number of medians is not implicitly considered in *SubPmed*, we can relax only the constraints (1) in the Lagrangean sense with multipliers $\pi \in R^n$. Doing this, for a given $t \in R$ and $\pi \in R^n$, the Lagrangean/surrogate relaxation of problem *Pmed* can be formulated as:

$$LS_{\pi,t}Pmed:\ v(L_{\pi,t},\ Pmed) = Min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n}(d_{ij}-t\pi_i)x_{ij} + t\sum_{j=1}^{n}x_{jj}\right\}$$

subject to (2)–(4)

$LS_{\pi,t}Pmed$ can be solved considering constraint (2) implicitly and decomposing the problem for index j, obtaining the following n subproblems:

$$Min\sum_{i=1}^{n}(d_{ij}-t\pi_i)x_{ij}$$

subject to (3) and (4).

Each subproblem is easily solved, calculating $\beta_j = \sum_{i=1}^{n}[\min\{0, d_{ij}-t\pi_i\}]$ and defining J as the index set for the p smallest β_j (here constraint (2) is considered implicitly). Then, a solution x_{ij}^{π} to $LS_{\pi,t}Pmed$ is:

$$x_{jj}^{\pi} = \begin{cases}1, & \text{if } j \in J\\ 0, & \text{otherwise}\end{cases}$$

and for all $i \ne j$,

$$x_{ij}^{\pi} = \begin{cases}1, & \text{if } j \in J \text{ and } d_{ij}-t\pi_i < 0\\ 0, & \text{otherwise}\end{cases}$$

The solution value is calculated as:

$$v(LS_{\pi,t}Pmed) = \sum_{j=1}^{n}\beta_j x_{jj}^{\pi} + t\sum_{i=1}^{n}\pi_i$$

Note that x_{jj}^{π} is always candidate to be 1, since $(d_{jj} - t\pi_i) = -t\pi_i \le 0$, and this allows one or more x_{ij}'s to be 1 if the corresponding $(d_{ij} - t\pi_i)$ are negative.

For a fixed multiplier π, the usual Lagrangean relaxation is obtained from $LS_{\pi,t}Pmed$ by setting $t = 1$. The best value for t can be obtained as optimal solution of the local Lagrangean dual problem:

$$D_{\pi,t}:\ v(D_{\pi,t}) = \underset{t\in R}{Max}\{v(LS_{\pi,t}Pmed)\}$$

It is well known that the function l: R → R, $(t, v(LS_{\pi,t}Pmed))$ is concave and piecewise linear. Then, an exact solution to $D_{\pi,t}$ may be obtained by a dichotomous search over different values of t (Senne and Lorena (2000)).

The Lagrangean/surrogate relaxation can be integrated to the column generation process transferring the multipliers μ_i $(i = 1, ..., n)$ of problem *SC-Pmed* to the

Lagrangean dual problem $\underset{t\geq 0}{Max}\ v(LS_{\mu,t}Pmed)$. The median (and allocated non-medians) will be determined as the node with the smallest contribution to $v(D_{\pi,t})$ in the cluster that corresponds to the incoming column on the new subproblem:

$$Sub_tPmed:\ v(Sub_tPmed) = \underset{j\in N}{Min}\left[\underset{\alpha_{ij}\in\{0,1\}}{Min}\sum_{i=1}^{n}(d_{ij}-t\mu_i)\alpha_{ij}\right]$$

Let $\bar{j}$ be the node index reaching the overall minimum on $v(Sub_tPmed)$. The new sets S_j are $\{i \mid \alpha_{ij} = 1 \text{ in } Sub_tPmed\}$ and the column $\begin{bmatrix}\alpha_{\bar{j}}\\ 1\end{bmatrix}$, as well as all the corresponding columns $\begin{bmatrix}\alpha_j\\ 1\end{bmatrix}$ satisfying expression (9), can be added to *SC-Pmed*. Note that the columns generated can be different from the ones generated using *SubPmed*, but they are incoming columns only if they satisfy the usual reduced cost tests.

Rewriting Sub_tPmed:

$$v(Sub_tPmed) = \underset{j\in N}{Min}\left[\underset{\alpha_{ij}\in\{0,1\}}{Min}\sum_{i=1}^{n}(d_{ij}-t\mu_i)\alpha_{ij}\right]$$
$$= \underset{j\in N}{Min}\left\{\underset{y_{ij}\in\{0,1\}}{Min}\left[\sum_{i=1}^{n}(d_{ij}-\mu_i)\alpha_{ij} + (1-t)\mu_i\sum_{i=1}^{n}\alpha_{ij}\right]\right\}$$

and the multiplier $(1 - t)$ can be assumed as the dual variable corresponding to the following additional constraint in the master problem *SC-Pmed*:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\mu_i A_k y_k \geq \sum_{i=1}^{n}\mu_i \qquad (10)$$

Constraint (10) is formulated using the dual solution $\mu \in R_+^n$ of the current master problem. The new *SC-Pmed* is:

$$SC\text{-}Pmed^{\mu}:\ v(SC\text{-}Pmed^{\mu}) = Min\sum_{k=1}^{m}c_k y_k$$

$$\text{subject to } \sum_{k=1}^{m}A_k y_k \geq 1$$

$$\sum_{k=1}^{m}x_k = p$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\mu_i A_k y_k \geq \sum_{i=1}^{n}\mu_i$$

$$y_k \in [0, 1]$$

Constraint (10) is a surrogate constraint derived of constraints (7) in *SC-Pmed* and is considered only implicitly by the dual variable $(1 - t)$. It follows, by linear programming duality, that $(1 - t) \geq 0$. As t is the Lagrangean multiplier associated with the surrogate constraint derived from constraints (7), it is defined nonnegative, following that the multiplier t is always situated in the interval [0, 1].

The implicit consideration of (10) is beneficial to the column generation process because some columns can be anticipated in the process. These new identified columns can be more productive for the column generation process than the ones generated by *SubPmed.*

Comparing problems Sub_tPmed and *SubPmed* it is easy to see that, for $t \in [0, 1]$, if $d_{ij} - \mu_i > 0$ then $d_{ij} - t\mu_i > 0$ and in the column $\begin{bmatrix}y_j\\ 1\end{bmatrix}$ the corresponding $\alpha_{ij} = 0$ is not modified by using multiplier t. On the other hand, if $d_{ij} - \mu_i \leq 0$ then $d_{ij} - t\mu_i \leq 0$ or $d_{ij} - t\mu_i > 0$ and in the column $\begin{bmatrix}y_j\\ 1\end{bmatrix}$ some $\alpha_{ij} = 1$ can be flipped to $\alpha_{ij} = 0$. A direct consequence is that, for the same multipliers μ_i, the column cost $c_k = \underset{i\in S_k}{Min}(\sum_{j\in S_k} d_{ij})$ calculated for problem *SC-Pmed* can be smaller using the Lagrangean/surrogate approach. This effect is best shown on computational tests of section 5 and results on faster convergence, even when multiple columns are added to the pool at each iteration of the process.

4. ALGORITHM IMPLEMENTATION

The column generation algorithm proposed in this paper can be stated as:

Algorithm *CG*(*t*)

(i) Set an initial pool of columns to *SC-Pmed*;
(ii) Solve *SC-Pmed* and obtain the dual prices μ_i ($i = 1, ..., n$) and ρ;
(iii) Solve approximately a local Lagrangean/surrogate dual $\underset{t\geq 0}{Max}\ v(LS_{\mu,t}Pmed)$, returning the corresponding columns of Sub_tPmed;
(iv) Append the columns $\begin{bmatrix}y_j\\ 1\end{bmatrix}$ satisfying expression (9) to *SC-Pmed*;
(v) If no columns are found in Step (iv) then stop;
(vi) Perform tests to remove columns and return to Step (ii).

The case $t = 1$ gives the algorithm *CG*(1), the traditional column generation process. In this case, the search for t in the step (iii) is not executed, and the usual Lagrangean bound $v(LS_{\mu,1}Pmed)$ implicitly solves problem Sub_1Pmed. In any case, the bounds $v(SC\text{-}Pmed)$ and $v(LS_{\mu,t}Pmed)$ are calculated at each iteration.
The following procedure RC is used in Step (vi):

Procedure *RC*

Let:

mean_rc be the average of the reduced costs for the initial pool of columns of *SC-Pmed*;

tot_cols be the total number of columns in the current *SC-Pmed*;

rc_i be the reduced cost of the columns in the current *SC-Pmed* (i = 1, ..., *tot_cols*);

rc_factor be a parameter to control the strength of the test.

For i = 1, ..., *tot_cols* do
  Delete column i from the current *SC-Pmed* if rc_i > *rc_factor* * *mean_rc*.
End_For;

5. COMPUTATIONAL TESTS

The algorithms presented in the previous section were implemented in *C* and executed on a Sun Ultra 30 workstation. The *p*-median instances from OR-Library (Beasley (1990)) were used in the initial tests. The results are reported in the following tables (note that the symbol "–" in these tables means "null gap"). In these tables, all the computer times do not include the time needed to setup the problem.

Table 1 reports the results for *CG*(*t*) and *CG*(1) (in parenthesis) obtained for *rc_factor* = 1.0 and maximum number of 1000 iterations. Table 1 contains:

- the number of nodes in the network and the number of medians to be located;
- the optimal integer solution for the instance (available in OR-Library);
- the total number of iterations;
- the total number of columns generated;
- the number of columns effectively used in the process;
- primal gap = 100 × |(*v*(*SC-Pmed*) – *optimal*)|/*optimal*, or the percentage deviation from *optimal* to the best
- primal solution value *v*(*SC-Pmed*) found by *CPLEX*;
- dual gap = 100 × (*optimal* – $v(LS_{\pi,t}Pmed)$)/*optimal*, or the percentage deviation from *optimal* to the best relaxation value $v(LS_{\pi,t}Pmed)$ found;
- the total computational time (in seconds).

The combined use of Lagrangean/surrogate and column generation can be very interesting, especially for large-scale problems. Algorithm *CG*(*t*) is faster and found the same results of *CG*(1) generating a smaller number of columns. Figure 1 shows that the typical behaviors of the Lagrangean bound $v(LS_{\pi,1}Pmed)$ and the Lagrangean/ surrogate bound $v(LS_{\pi,t}Pmed)$ are conserved using column generation. The figure shows the values obtained at each iteration of *CG*(*t*) and *CG*(1) for a problem instance with n = 900 and p = 300.

The results of Table 1 also show that, for a given number of nodes, the smaller the number of medians in the instance, the harder is the problem to be solved using the column generation approaches *CG*(*t*) or *CG*(1).

Table 2 includes the *LS* algorithm, presented in Senne and Lorena (2000), which uses the Lagrangean/surrogate relaxation embedded on a dual optimized by a subgradient method. This table shows the results obtained for the set of the most time consuming instances (for *LS*) from OR-Library in order to compare the *CG* approaches discussed here and the *LS* approach. The results presented in Table 2 were obtained for *rc_factor* = 1.0 and a maximum number of 50 iterations. The columns *CG* show the results for *CG*(*t*) and *CG*(1) (in parenthesis). For the *LS* algorithm, the primal gap is calculated as 100 × (*feasible solution* – *optimal*)/ *optimal*, where the *feasible solution* value is obtained after a local search procedure for the best allocation solution was performed on each cluster.

The instances in Table 2 seem to be easy for *CG* approaches. For these instances, the computational tests have confirmed the superiority of the combined use of Lagrangean/surrogate and column generation compared to the Lagrangean/surrogate embedded in a subgradient search method. Note that the *LS* approach was already shown to be faster than Lagrangean heuristics in Senne and Lorena (2000).

The results from Table 1 show that *CG*(*t*) is able to generate fewer and higher quality columns than *CG*(1). This becomes evident when the number of useful columns is limited by decreasing *rc_factor*, as reported by Table 3 and shown by Figure 2, for an instance with n = 200 and p = 5.

The results from Table 3 and Figure 2 shows that the column generation procedure including the Lagrangean/ surrogate method, *CG*(*t*), is able to produce high quality approximate solutions even if only a few number of columns is used. The traditional approach, *CG*(1), keeps on several iterations with no improvement on the optimal value of the master problem, or it can stay unchanged all the time (see Figure 2 for *rc_factor* = 0.3).

The computational tests proceeded now considering a large-scale instance. The PCB3038 instance in the TSPLIB, compiled by Reinelt (1994), was considered for the tests. The results are given in Table 4, Table 5 and Table 6. In these tables, primal gap and dual gap are calculated as following:

- primal gap = 100 × |(*v*(*SC-Pmed*) – *best known solution*)|/*best known solution*
- dual gap = 100 × (*best known solution* – $v(LS_{\pi,t}Pmed)$)/ best known solution

The results from Tables 4, 5 and 6 confirm that *CG*(*t*) is really able to generate better quality columns than *CG*(1). Evidently, if more columns are deleted by *RC* algorithm, more iterations are necessary to reach the same results, which highlights the superiority of *CG*(*t*) as compared to *CG*(1). The *rc_factor* can be understood as a trade-off parameter to decide between computational time and storage availability.

Table 1. Computational results for instances from OR-Library

n	*p*	optimal solution	iter	columns generated	columns used	primal gap	dual gap	total time
100	5	5819	184 (155)	5458 (5969)	3861 (3775)	– (–)	– (–)	36.35 (36.31)
200	5	7824	399 (381)	16929 (23630)	11763 (12533)	– (–)	– (–)	902.77 (1625.63)
200	10	5631	936 (757)	24375 (24483)	20584 (18701)	– (–)	– (–)	996.00 (864.83)
300	5	7696	1000 (919)	39299 (48431)	38173 (42704)	0.246 (–)	1.796 (–)	17889.12 (23337.79)
300	10	6634	731 (1000)	33342 (55200)	26638 (36864)	– (0.108)	– (0.215)	10749.91 (13214.36)
300	30	4374	198 (1000)	12040 (40166)	8016 (30381)	– (–)	– (0.118)	831.22 (1057.43)
400	5	8162	1000 (1000)	60624 (85762)	53181 (64266)	0.686 (0.832)	1.662 (1.022)	52807.93 (83877.77)
400	10	6999	675 (627)	41156 (66680)	26561 (26070)	– (–)	– (–)	36829.25 (41202.98)
400	40	4809	195 (191)	18160 (24213)	13130 (13101)	– (–)	– (–)	1055.20 (1078.27)

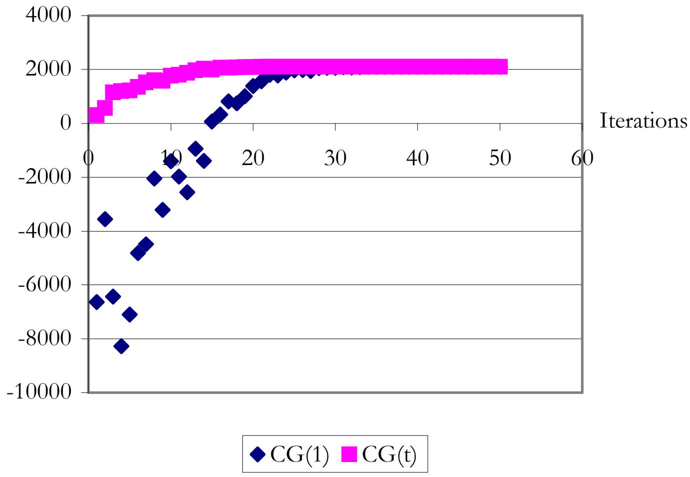
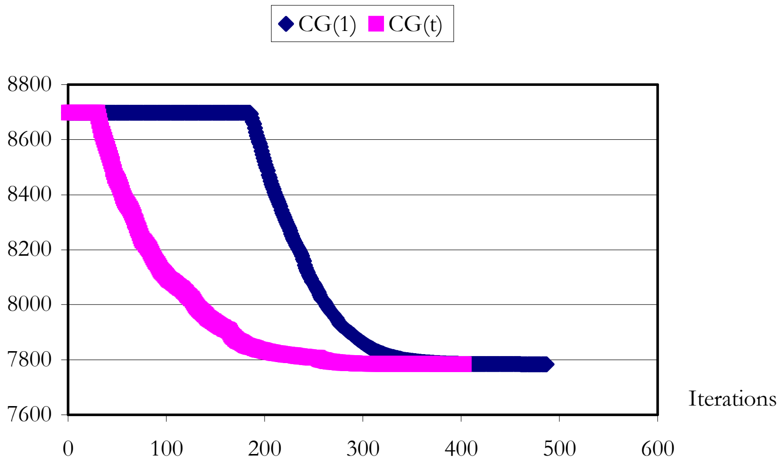


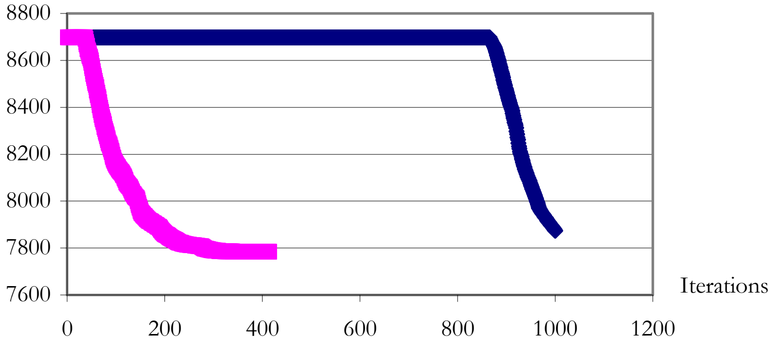
Figure 1. Typical computational behavior of the dual bounds $v(LS_{\pi,1}Pmed)$ and $v(LS_{\pi,t}Pmed)$.

Table 2. Comparison of *LS* and *CG* approaches

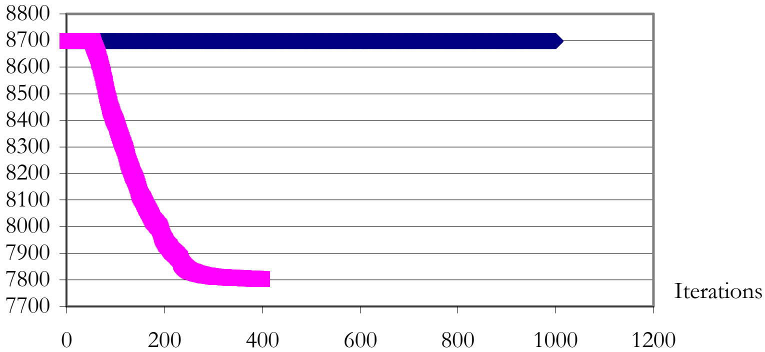
n	*p*	optimal solution	primal gap *LS*	primal gap *CG*	dual gap *LS*	dual gap *CG*	total time *LS*	total time *CG*
100	33	1355	–	– (–)	–	– (–)	0.58	0.37 (0.35)
200	67	1255	–	– (–)	–	– (0.667)	4.00	1.29 (1.89)
300	100	1729	–	0.116 (–)	–	0.058 (–)	16.78	4.55 (4.90)
400	133	1789	–	0.112 (–)	–	0.950 (0.783)	51.80	6.21 (6.04)
500	167	1828	–	0.055 (0.036)	–	0.310 (0.210)	127.60	11.00 (12.91)
600	200	1989	–	0.302 (0.101)	–	0.285 (0.235)	257.02	15.81 (17.59)
700	233	1847	–	0.081 (0.325)	–	0.379 (0.785)	482.97	21.50 (21.41)
800	267	2026	–	0.518 (0.222)	–	0.346 (0.271)	1374.74	26.14 (27.95)
900	300	2106	0.047	0.518 (0.607)	0.004	0.827 (0.443)	3058.65	33.37 (49.99)



(a) Results for *rc_factor* = 0.5



(b) Results for *rc_factor* = 0.4



(c) Results for *rc_factor* = 0.3

Figure 2. *SC-Pmed* values at each iteration.

Table 3. Limiting useful columns by *rc_factor*

rc_factor	iter	columns generated	columns used	primal gap	dual gap	total time
0.5	403	18493	7543	–	–	619.63
	(487)	(47634)	(7364)	(–)	(–)	(971.59)
0.4	414	20395	6627	–	–	613.79
	(1000)	(167247)	(3270)	(0.631)	(4.635)	(1370.99)
0.3	400	23521	3886	–0.276	2.010	532.27
	(1000)	(186267)	(421)	(11.171)	(65.181)	(905.67)

Table 4. Computational results for PCB3038 instances (*rc_factor* = 1.0)

p	best known solution	iter	columns generated	columns used	primal gap	dual gap	total time
300	187723.46	42	58339	44599	0.043	0.044	22235.02
		(48)	(65007)	(44081)	(0.043)	(0.043)	(35132.76)
350	170973.34	47	58758	45576	0.044	0.045	10505.93
		(37)	(65545)	(43956)	(0.044)	(0.045)	(20457.59)
400	157030.46	33	50807	37318	0.008	0.008	4686.27
		(35)	(60287)	(39563)	(0.008)	(0.008)	(8962.82)
450	145422.94	32	45338	32637	0.052	0.053	1915.84
		(30)	(52515)	(33544)	(0.052)	(0.052)	(3241.71)
500	135467.85	22	31778	22854	0.036	0.036	597.86
		(21)	(36386)	(22839)	(0.035)	(0.036)	(787.46)

Table 5. Computational results for PCB3038 instances (*rc_factor* = 0.5)

p	best known solution	iter	columns generated	columns used	primal gap	dual gap	total time
300	187723.46	79	96798	40053	0.043	0.044	19371.01
		(67)	(111597)	(39448)	(0.043)	(0.043)	(36029.23)
350	170973.34	65	86113	29179	0.044	0.045	7077.99
		(53)	(90651)	(31664)	(0.044)	(0.044)	(12905.94)
400	157030.46	53	77174	22857	0.008	0.008	2872.48
		(49)	(94716)	(30101)	(0.008)	(0.008)	(5682.90)
450	145422.94	40	55870	18662	0.052	0.052	1288.56
		(41)	(80631)	(23767)	(0.052)	(0.053)	(2568.56)
500	135467.85	34	45092	16750	0.036	0.036	716.78
		(53)	(79338)	(22956)	(0.036)	(0.044)	(1425.33)

Table 6. Computational results for PCB3038 instances (*rc_factor* = 0.2)

p	best known solution	iter	columns generated	columns used	primal gap	dual gap	total time
300	187723.46	617	958984	28718	0.043	0.044	36333.01
		(834)	(1655221)	(93535)	(0.043)	(0.043)	(117707.31)
350	170973.34	393	576789	24475	0.044	0.044	10823.10
		(719)	(1232357)	(74005)	(0.044)	(0.044)	(49874.03)
400	157030.46	235	330475	15973	0.008	0.008	4529.20
		(586)	(1232440)	(54724)	(0.008)	(0.008)	(39883.02)
450	145422.94	155	176348	13489	0.052	0.052	2356.97
		(363)	(843026)	(20517)	(0.052)	(0.052)	(12990.88)
500	135467.85	121	119884	12997	0.035	0.035	1682.15
		(210)	(420737)	(24254)	(0.036)	(0.036)	(4340.33)

Based on the computational tests we can draw the following overall conclusions:

- Instances with small number of medians are hard to column generation approaches and easy for Lagrangean/surrogate and subgradient methods. On the other hand, instances with large number of medians are easy to column generation and hard to Lagrangean/ surrogate and subgradient methods. It seems that they are companion methods in this sense.
- Algorithm *CG*(t) can be used as a substitute of *CG*(1), specially on hard instances.

6. COMMENTS AND CONCLUSION

Column generation has been recognized as a useful method for modeling and solving large-scale linear programming problems. Despite that, the column generation application may have some computational problems, when the subproblem generates too many columns that not improve the master problem bound.

The combined use of Lagrangean/surrogate relaxation and column generation shows some improvement to the traditional column generation process. Depending on the instance, both methods, the column generation and the Lagrangean/surrogate embedded with subgradient like methods, can be improved.

Algorithm *CG*(*t*) also calculates lower bounds, the Lagrangean/surrogate bound, that can be used, in similar way to other bounds (Farley (1990)), to stop the process at a convenient iterations limit. It also can be useful to branch-and-price methods (Vance et al. (1994), Barnhart et al. (1998)). The *CG*(*t*) application to *p*-median problems is an alternative to Lagrangean heuristics, especially on hard instances.

ACKNOWLEDGMENTS

The authors acknowledge CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico (processes 305346/2003-2, 304598/2003-8 and 380758/2002-4, respectively) for partial financial research support.

REFERENCES

1. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., and Vance, P.H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46: 316-329.
2. Beasley, J.E. (1990). OR-library: Distributing test problems by electronic mail. *Journal Operational Research Society*, 41: 1069-1072.
3. Dantzig, G.B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8: 101-111.
4. Day, P.R. and Ryan, D.M. (1997). Flight attendant rostering for short-haul airline operations. *Operations Research*, 45: 649-661.
5. Desrochers, M. and Soumis, F.A. (1989). Column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23: 1-13.
6. Desrochers, M., Desrosiers, J., and Solomon, M.A. (1992). New optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40: 342-354.
7. Du Merle, O., Goffin, J.L., and Vial, J.P. (1998). On improvements to the analytic centre cutting plane method. *Computational Optimization and Applications*, 11: 37-52.
8. Du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, 194: 229-237.
9. Farley, A.A. (1990). A note on bounding a class of linear programming problems, including cutting stock problems. *Operations Research*, 38: 992-993.
10. Galvão, R.D.A. (1981). Note on garfinkel, neebe and Rao's LP decomposition for the *P*-Median problem. *Transportation Science*, 15(3): 175-182.
11. Garfinkel, R.S., Neebe, W., and Rao, M.R. (1974). An algorithm for the *M*-median location problem. *Transportation Science*, 8: 217-236.
12. Gilmore, P.C. and Gomory, R.E. (1961). A linear programming approach to the cutting stock problem. *Operations Research*, 9: 849-859.
13. Gilmore, P.C. and Gomory, R.E. (1963). A linear programming approach to the cutting stock problem - Part ii. *Operations Research*, 11: 863-888.
14. CPLEX 6.5. (1999). ILOG Inc., Cplex Division.
15. Kelley, J.E. (1960). The cutting plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8: 703-712.
16. Lorena, L.A.N. and Senne, E.L.F. (2004). A column generation approach to capacitated *P*-Median problems. *Computers & Operations Research*, 31(6): 863-876.
17. Lübbecke, M.E. and Desrosiers, J. (2002). *Selected Topics in Column Generation*, Les Cahiers du GERAD, G-2002-64, HEC Montreal, Canada.
18. Marsten, R.M., Hogan, W., and Blankenship, J. (1975). The boxstep method for large-scale optimization. *Operations Research*, 23: 389-405.
19. Minoux, M.A. (1987). Class of combinatorial problems with polynomially solvable large scale set covering/set partitioning relaxations. *RAIRO: Operations Research*, 21(2): 105–136.
20. Narciso, M.G. and Lorena, L.A.N. (1999). Lagrangean/surrogate relaxation for generalized assignment problems. *European Journal of Operational Research*, 114(1): 165-177.
21. Neame, P.J. (1999). *Nonsmooth Dual Methods in Integer Programming*, PhD Thesis, Department of Mathematics and Statistics, The University of Melbourne.
22. Reinelt, G. (1994). *The Traveling Salesman Problem: Computational Solutions for TSP Applications*, Lecture Notes in Computer Science 840, Springer-Verlag, Berlin.
23. Senne, E.L.F. and Lorena, L.A.N. (2000). Lagrangean/surrogate heuristics for *p*-median problems. In: M. Laguna and J.L. Gonzalez-Velarde (Eds.), *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, Kluwer Academic Publishers, pp. 115-130.
24. Swain, R.W. (1974). A parametric decomposition approach for the solution of uncapacitated location problems. *Management Science*, 21: 955-961.
25. Valério de Carvalho, J.M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86: 629-659.
26. Vance, P. (1993). *Crew Scheduling, Cutting Stock and Column Generation: Solving Huge Integer Programs*, PhD Thesis, Georgia Institute of Technology.
27. Vance, P.H., Barnhart, C., Johnson, E.L., and Nemhauser, G.L. (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, 3: 111-130.