Computational Optimization and Applications, 35, 239–260, 2006

DOI: 10.1007/s10589-006-6513-6

Solving the p-Median Problem with a Semi-Lagrangian Relaxation*

C. BELTRAN† cesar.beltran@urjc.es
Logilab, HEC, University of Geneva, Switzerland

C. TADONKI tadonki@embl.fr
Centre Universitaire Informatique, University of Geneva, Switzerland

J.-PH.VIAL jean-philippe.vial@hec.unige.ch
Logilab, HEC, University of Geneva, Switzerland

Received December 10, 2004; Revised August 15, 2005

Published Online: 5 June 2006

Abstract. Lagrangian relaxation is commonly used in combinatorial optimization to generate lower bounds for a minimization problem. We study a modified Lagrangian relaxation which generates an optimal integer solution. We call it semi-Lagrangian relaxation and illustrate its practical value by solving large-scale instances of the p-median problem.

Keywords: Lagrangian relaxation, combinatorial optimization, p-median problem

1. Introduction

Lagrangian relaxation (LR) is commonly used in combinatorial optimization to generate lower bounds for a minimization problem [8]. For a given problem, there may exist different Lagrangian relaxations. The higher the optimal value of the associated Lagrangian dual function, the stronger the relaxation and the more useful in solving the combinatorial problem in a branch-and-bound framework [11, 21]. Ideally, one would like to work with the strongest possible Lagrangian, one that closes the integrality gap. Are there combinatorial problems for which such a Lagrangian relaxation exists, and if yes, are the associated subproblems computationally tractable?

In this paper we give a positive answer to the first question: we study a relaxation, which we call semi-Lagrangian relaxation (SLR), that closes the integrality gap for any (linear) combinatorial problem with equality constraints. Regarding the second question, we also give a positive answer for the p-median problem, a well-studied combinatorial problem [3, 17]. For this integer programming problem, the standard Lagrangian

*This work was partially supported by the Fonds National Suisse de la Recherche Scientifique, grant 12-57093.99 and the Spanish government, MCYT subsidy dpi2002-03330.

†To whom correspondence should be addressed.

relaxation consists in relaxing the equality constraints that ensure that each "customer" is assigned to exactly one median. This yields a maxmin optimization problem, in which the binding equality constraints are no longer present. The inner minimization problem is thus separable and easy. Unfortunately, the Lagrangian relaxation yields the same optimal value as the linear relaxation. To strengthen the standard Lagrangian relaxation, we insert into the inner minimization problem the equality constraint, as a "less than or equal" inequality. We call this process, a semi-Lagrangian relaxation. It is quite general, as it applies to any problem with equality constraints.

The semi-Lagrangian relaxation has the theoretical property that its optimal value is the same as for the original problem. This relaxation thus closes the integrality gap. Unfortunately, the associated subproblem may be as difficult as the original problem itself. However, for the p-median problem, the subproblem is an easy variant of the uncapacitated facility location problem, in which only "profitable" customers must be assigned to a facility. This subproblem is NP-hard, but when the Lagrange multipliers are small, it turns out that very few customers are "profitable" and the subproblem becomes easy. This suggests that a method that would approach the set of optimal Lagrangian multipliers with small multiplier values, may keep the subproblem relatively easy until it produces an optimal (dual) solution. Moreover, at this solution, the subproblem outputs a feasible integer solution that is optimal for the original problem.

We have implemented this scheme for the p-median problem and tested it on a collection of the large-scale instances studied in [2, 13]. We have been able to improve the best known dual bounds for five of the six non solved difficult problems in that collection (one of them is solved up to optimality by our method). We also improved the computation time on a few difficult problems by a significant amount. However, in the average, our approach does not improve the results of Avella et al., which are, in our opinion, among the best reported computational results on the p-median problem. This is not surprising, since these authors exploit the combinatorial structure of the p-median problem to construct specific efficient cuts (lifted odd-hole inequalities, cycle inequalities, etc.) to be used in a sophisticated branch-cut-and-price (BCP) algorithm. In contrast our simpler method is general and does not resort to a branch-and-bound scheme. Of course, the integer programming solver (CPLEX in our case) that handles the semi-Lagrangian subproblems relies on sophisticated branch-and-bound schemes, but the remarkable fact is that this solver cannot handle the p-median problems in their initial formulation, unless they have small dimension. Since the semi-Lagrangian relaxed problem is a variant of the uncapacitated facility location problem, usually very sparse, its special structure could be exploited to improve the solution time: any such improvement will directly translate into the same improvement in the overall procedure.

Shortly after we completed the paper, we have been informed of a closely related work [19]. This paper is concerned with sensitivity analysis for integer programming based on the subadditive dual problem [16]. The author of [19] shows that it is not necessary to work on the full class of subadditive functions but in what he calls the family of *generator subadditive functions*. He proves that this family contains an optimal subadditive function (OSF) that can be used to perform sensitivity analysis and to compute *all* the solutions for an integer programming problem. The author applies his algorithm to solve small to medium size instances of the set partitioning problem [20]. It turns out that maximizing the semi-Lagrangian dual function is equivalent to computing a generator OSF, but there are many possible methods to perform this operation.

Our work differs from Klabjan's on the following aspects. First, we recognize that the semi-Lagrangian dual function is concave but non-differentiable and thus difficult to optimize. To perform this task, we use Proximal-ACCPM [6], an enhancement of ACCPM (Analytic Center Cutting Plane Method) [9], which is discussed with more detail in Section 4.2. Second, we are essentially concerned with numerical issues. We exploit the remarkable fact that, in the p-median problem, the computation of the semi-Lagrangian dual function often breaks down into several independent subproblems. For this reason, we are able to solve p-median instances which have up to seven times the size CPLEX can handle successfully. Third, we compare our approach with a state-of-the art branch-cut-and-price method [2] for the p-median problem. Our numerical results on very large instances are surprising as they show that our branch-and-bound free approach is quite effective and able to close the integrality gap on some problems that were not previously solved. Our approach is conceptually simple and easy to implement since it is based on standard tools: Proximal-ACCPM to maximize the SLR dual function and CPLEX to solve the associated subproblems.

The paper is organized as follows: In Section 2 we study the semi-Lagrangian relaxation (concept and properties) for the case of linear integer programming problems. In Sections 3 and 4 we apply the semi-Lagrangian relaxation to the p-median problem. In Section 5 we test the semi-Lagrangian relaxation by solving large scale p-median problems. Conclusions are given in Section 6. case of general functions.

2. Semi-Lagrangian relaxation

Consider the primal problem

$$
\begin{aligned}
z^* = \min_x \quad & c^T x \\
\text{s.t.} \quad & Ax = b, && \text{(1a)}\\
& x \in S := X \cap \mathbb{N}^n. && \text{(1b)}
\end{aligned}
$$

Assumption 1. All components in A, b and c are non-negative and $X \subset \mathbb{R}^n$ is a cone (thus, $0 \in X$). Note that, since A and b are non-negative, the set $\{x \in S \mid Ax \leq b\}$ is bounded (finite).

The standard Lagrangian relaxation consists in relaxing the (linear) equality constraints and solving the dual problem

$$z_{LR} = \max_u \mathcal{L}_{LR}(u), \tag{2}$$

where

$$\mathcal{L}_{LR}(u) = b^T u + \min_x \{(c - A^T u)^T x \mid x \in S\}. \tag{3}$$

The optimal solution of the Lagrangian dual yields a lower bound for the original problem $z_{LR} \leq z^*$.

The semi-Lagrangian relaxation consists in relaxing the equality constraint as in (3), but keeping in the meantime a weaker form of the equality constraint in the oracle (subproblem). Namely,

$$z_{SLR} = \max_u \mathcal{L}_{SLR}(u), \tag{4}$$

where

$$\mathcal{L}_{SLR}(u) = b^T u + \min_x \{(c - A^T u)^T x \mid Ax \leq b, x \in S\}. \tag{5}$$

The oracle (5) is more constrained than (3). Its minimum value is thus higher

$$z_{LR} \leq z_{SLR} \leq z^*.$$

The semi-Lagrangian relaxation is thus stronger than the Lagrangian relaxation. However, solving the oracle (5) may be (much) more difficult than solving (3). Actually, the difficulty in solving (5) depends on the particular values of u. We shall consider two extreme cases. First, assume that $u = 0$. By Assumption 1, $b \geq 0$ and $0 \in S$, therefore 0 is feasible to (5). Also by Assumption 1, $c \geq 0$ and then 0 is a trivial solution to (5). The second case occurs when all components of u are positive and very large. If we write (5) as

$$\mathcal{L}_{SLR}(u) = \min_x \{c^T x + (b - A^T x)^T u \mid Ax \leq b, x \in S\}$$

the very large penalty on $b - A^T x$ imposes to chose x such that $Ax \geq b$. Since x is explicitly constrained by $Ax \leq b$, the optimal solution of (5) meets the original constraint $Ax = b$. Solving the oracle may be just as difficult as solving the original problem (2). This is the bad side of the situation, but it also has a positive side: it gives indication that the optimal solution of (4) may be strictly bigger that (2), thereby reducing the integrality gap.

We want to argue that there may exist intermediary situations, where the oracle (5) is not too difficult to solve, and thus is practical. To this end, we cast our previous discussions into formal propositions.

Theorem 1. *The original problem* (1) *and the semi-Lagrangian dual problem* (4) *have the same optimal value.*

Proof: By [8] we know that

$$\begin{aligned} z_{SLR} &= \min\{c^T x : Ax = b, x \in \operatorname{conv}(Ax \leq b, x \in S)\} && \text{(6a)}\\ &= \min\{c^T x : Ax = b, Ax \leq b, x \in S\} && \text{(6b)}\\ &= \min\{c^T x : Ax = b, x \in S\} && \text{(6c)}\\ &= z^*. && \text{(6d)} \end{aligned}$$

Equality (6b) comes from the fact that the faces of an integer polytope are integral. □

Theorem 1, already proved in [19], shows that the suggested oracle is the strongest possible relaxation. The next theorem and its corollary show that the optimal set of $\mathcal{L}_{SLR}(u)$ is unbounded.

Theorem 2. *The function $\mathcal{L}_{SLR}$ is non-decreasing.*

Proof: Let $u \geq u'$ and let $x(u)$ and $x(u')$ be optimal solutions of (5) at u and u' respectively. Let us show that $\mathcal{L}_{SLR}(u) \geq \mathcal{L}_{SLR}(u')$. Using the fact that $Ax(u) \leq b$ and that $x(u')$ minimizes $c^T x + (b - Ax)^T u'$, we have

$$\begin{aligned}\mathcal{L}_{SLR}(u) &= c^T x(u) + (b - Ax(u))^T u,\\ &= c^T x(u) + (b - Ax(u))^T u' + (b - Ax(u))^T (u - u'),\\ &\geq c^T x(u) + (b - Ax(u))^T u',\\ &\geq c^T x(u') + (b - Ax(u'))^T u' = \mathcal{L}_{SLR}(u').\end{aligned}$$

□

Theorem 2 induces a domination criterion in the set of optimal multipliers u. Formally, we state the following corollary.

Corollary 2.1. *Let u^* be an optimal solution of* (4). *The optimal set contains the unbounded set* $\{u \mid u \geq u^*\}$.

We can define the set of non dominated optimal solution as the Pareto frontier of the optimal set U^*. Let us picture in Figure 1 the set U^* and a possible trajectory of multipliers from the origin to the set.

We observe that we can choose an optimal solution x in the oracle (5) with the property that $x_j = 0$ if the reduced cost $(c - A^T u)_j$ is nonnegative. At the origin O ($u = 0$) of Figure 1, the oracle has the trivial solution $x = 0$. At point C, far inside the optimal set, all reduced costs are negative. The oracle is difficult, since it is essentially equivalent

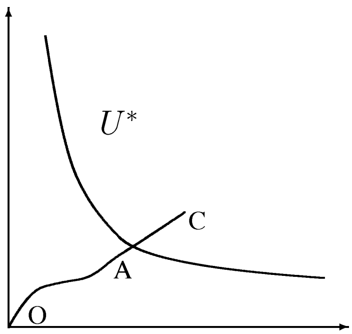

Figure 1. Path to the optimal set of dual multipliers.

to the original problem. The difficulty in solving the oracle increases as one progresses along the path OAC. At A, close to the origin O, the oracle problem involves only few variables and might thus be easy.

The key issue is whether the oracle (5) is easy enough to solve at points on the Pareto frontier of U^* and near of it. If yes, we have at hand a procedure to find an exact solution of the original problem by solving a sequence of moderately difficult problems. Let us show here that if $u \in \text{int}(U^*)$ one can get an optimal solution to (1).

Theorem 3. *Let $x(u)$ be an optimal solution of the semi-Lagrangian relaxation problem* (4) *at u. If $x(u)$ satisfies* (1a) *at $u \in U^*$, then it is optimal for the original problem* (1). *Moreover, $x(u)$ at any $u \in int(U^*)$ satisfies* (1a) *and is thus optimal for* (1).

Proof: The first statement of the theorem is a standard result [7]. To prove the second statement, let $u \in \text{int}(U^*)$. Since $u \in \text{int}(U^*)$, there exists $u' \in U^*$ such that $u' < u$. Then,

$$z^* = c^T x(u') + (b - A^T x(u'))^T u' = c^T x(u) + (b - A^T x(u))^T u.$$

Thus

$$\begin{aligned} z^* &= c^T x(u') + (b - A^T x(u'))^T u' \\ &\leq c^T x(u) + (b - A^T x(u))^T u' \quad \text{(since } x(u) \text{ is suboptimal for (5) at } u'\text{)} \\ &= c^T x(u) + (b - A^T x(u))^T u + (b - A^T x(u))^T (u' - u) \\ &= z^* + (b - A^T x(u))^T (u' - u). \end{aligned}$$

Thus, $(b - A^T x(u))^T (u' - u) \geq 0$. Since $u' < u$, and $A^T x(u) \leq b$ in (5), one has $A^T x(u) = b$. This concludes the proof of the theorem. □

The above discussion suggests a procedure to solve the original problem (1) via a semi-Lagrangian relaxation. The dual problem in the semi-Lagrangian relaxation is a concave non-differentiable one that can be solved by a specialized method of the cutting plane type. The difficulty in this approach is that the oracle is potentially difficult, possibly as difficult as the original problem (1). To make the overall procedure workable, the oracle should be solved exactly. An enumeration technique or an advanced commercial solver must be used. The cutting plane method must therefore be particularly efficient so as to require as few solvings of (5) as possible. In that respect, a good starting point might be of a great help. The natural suggestion is to use the optimal point of the dual problem of the standard Lagrangian relaxation (see Section 5.1)

$$\max_u \left\{ b^T u + \min_x \{(c - A^T u)^T x \mid x \in S\} \right\}.$$

3. Semi-Lagrangian relaxation for the p-median problem

In the p-median problem the objective is to open p 'facilities' from a set of m candidate facilities relative to a set of n 'customers', and to assign each customer to a single facility. The cost of an assignment is the sum of the shortest distances c_{ij} from a customer to a facility. The distance is sometimes weighed by an appropriate factor, e.g., the demand at a customer node. The objective is to minimize this sum. Applications of the p-median problem can be found in cluster analysis [12, 22], facility location [4], optimal diversity management problem [3], etc. The p-median problem can be formulated as follows

$$z^* = \min_{x,y} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{7a}$$

$$\text{s.t.} \sum_{i=1}^{m} x_{ij} = 1, \quad \forall j, \tag{7b}$$

$$\sum_{i=1}^{m} y_i = p, \tag{7c}$$

$$x_{ij} \leq y_i, \quad \forall i, j, \tag{7d}$$

$$x_{ij}, y_i \in \{0, 1\}, \tag{7e}$$

where $x_{ij} = 1$ if facility i serves the customer j, otherwise $x_{ij} = 0$ and $y_i = 1$ if we open facility i, otherwise $y_i = 0$.

The p-median is a NP-hard problem [17] for which polyhedral properties and some families of valid inequalities have been studied in [1, 5]. For this reason the p-median problem has been solved either by heuristic methods, such as the variable neighborhood decomposition method [13], or by exact methods, such as the branch-and-cut approach [3] and the branch-cut-and-price approach [2]. As far as we know, the latter represents the state of the art regarding exact solution methods to solve the p-median problem.

Following the ideas of the preceding section, we formulate the standard Lagrangian relaxation of the p-median problem, and two semi-Lagrangian relaxations.

3.1. Standard relaxation

The constraints (7b) and (7c) are both relaxed to yield the dual problem

$$z_1 = \max_{u,v} \mathcal{L}_1(u, v)$$

and the oracle

$$\mathcal{L}_1(u, v) = \min_{x,y} f(u, v, x, y) \tag{8a}$$

$$\text{s.t. } x_{ij} \leq y_i, \quad \forall i, j, \tag{8b}$$

$$x_{ij}, y_i \in \{0, 1\}, \tag{8c}$$

where

$$\begin{aligned} f(u, v, x, y) &= \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} u_j\left(1 - \sum_{i=1}^{m} x_{ij}\right) + v\left(p - \sum_{i=1}^{m} y_i\right) \\ &= \sum_{i=1}^{m}\left(\sum_{j=1}^{n}(c_{ij} - u_j)x_{ij} - vy_i\right) + \sum_{j=1}^{n} u_j + vp. \end{aligned}$$

We name *Oracle* 1 this oracle; it is trivially solvable. Its optimal solution is also optimal for its linear relaxation. Consequently, the optimum of $\mathcal{L}_1$ coincides with the optimum of the linear relaxation of (7) [24].

It is not possible to make this relaxation stronger by keeping the constraint on the number of medians (7c) in the oracle. Indeed, one can easily check that the linear relaxation of the ensuing oracle has an integer optimal solution. Therefore, keeping the constraint (7c) in the oracle, does not make the Lagrangian relaxation stronger than $\mathcal{L}_1$.

3.2. Partial semi-Lagrangian relaxation

To strengthen $\mathcal{L}_1$ we introduce the constraints $\sum_i x_{ij} \leq 1$, $j = 1, \ldots, n$ in the oracle. We obtain the dual problem

$$z_2 = \max_{u,v} \mathcal{L}_2(u, v)$$

and the new oracle

$$\begin{aligned} \mathcal{L}_2(u, v) = \min_{x,y}\ & f(u, v, x, y) && \text{(9a)} \\ \text{s.t.}\ & \sum_i x_{ij} \leq 1, \quad \forall j, && \text{(9b)} \\ & x_{ij} \leq y_i, \quad \forall i, j, && \text{(9c)} \\ & x_{ij}, y_i \in \{0, 1\}. && \text{(9d)} \end{aligned}$$

We name *Oracle* 2 this oracle. In view of the cost component in the y variables in the objective (9a), the problem resembles the well-known uncapacitated facility location (UFL) problem. However, the oracle differs from UFL on one important point. The standard cover (all customers must be assign to one facility) is replaced by a subcover inequality (9b). It implies the necessary condition that a customer j may be served by facility j only if the reduced cost $c_{ij} - u_j$ is negative. This fact will be used extensively in the procedure to solve the oracle: all variables x_{ij} with a non-negative cost are automatically set to zero, thereby reducing the size of the problem to solve dramatically.

The oracle for $\mathcal{L}_2$ is more difficult than the one for $\mathcal{L}_1$, but z_2 may be larger than z_1 when there exists an integrality gap between the optimal integer solution of (7) and its linear relaxation.

3.3. *Semi-Lagrangian relaxation*

If the solution of the strong oracle problem evaluated at the optimum of $\mathcal{L}_2$ is feasible for (7), then this solution is optimal for (7). In our numerical experiments, it has been the case on many instances. On other instances, the solution produced by the oracle only violates the constraint (7c) on the number of medians. To cope with this difficulty, we consider the strongest relaxation,

$$z_3 = \max_{u,v} \mathcal{L}_3(u, v)$$

with the new oracle

$$\mathcal{L}_3(u, v) = \min_{x,y} \; f(u, v, x, y) \tag{10a}$$

$$\text{s.t.} \; \sum_i x_{ij} \le 1, \quad \forall j, \tag{10b}$$

$$\sum_i y_i \le p, \tag{10c}$$

$$x_{ij} \le y_i, \quad \forall i, j, \tag{10d}$$

$$x_{ij}, y_i \in \{0, 1\}. \tag{10e}$$

We name *Oracle* 3 this oracle. In view of (10c), relaxation $\mathcal{L}_3$ is stronger than $\mathcal{L}_2$. It is also more difficult to solve.

4. p-Median solved by semi-Lagrangian relaxation

To solve the p-median problem by means of the semi-Lagrangian relaxation, we use the following general procedure.

Step 1. Solve the LR dual problem

$$(u^1, v^1) = \arg\max_{u,v} \mathcal{L}_1(u, v).$$

Let (x^1, y^1) be an optimal solution of (8) at (u^1, v^1). If (x^1, y^1) is feasible to (7), STOP: (x^1, y^1) is an optimal solution to (7).

Step 2. Solve the intermediate dual problem by using (u^1, v^1) as starting point

$$(u^2, v^2) = \arg\max_{u,v} \mathcal{L}_2(u, v).$$

1. Let (x^2, y^2) be an optimal primal solution associated to (u^2, v^2). If (x^2, y^2) is feasible to (7), STOP: (x^2, y^2) is an optimal solution to (7).
2. Let $(\hat{x}^2, \hat{y}^2)$ be a heuristic solution for problem (7). If this heuristic solution closes the primal-dual gap, STOP: $(\hat{x}^2, \hat{y}^2)$ is an optimal solution to (7).

Step 3. Solve the SLR dual problem by using (u^2, v^2) as starting point

$$(u^3, v^3) = \arg\max_{u,v} \mathcal{L}_3(u, v).$$

Let (x^3, y^3) be an optimal solution of (10) at (u^3, v^3). If (x^3, y^3) is feasible (satisfies (7b–7c)), STOP: (x^3, y^3) is an optimal solution to (7).

Step 4. Compute $\mathcal{L}_3(u^4, v^4)$, with $u_i^4 = u_i^3 + \delta$, $v_i^4 = v_i^3 + \delta$, for a small positive perturbation δ $(i = 1, \ldots, n)$. Let (x^4, y^4) be an optimal solution of (10) at (u^4, v^4). STOP: (x^4, y^4) is an optimal solution to (7).

The stopping criterion in Step 3 is justified by Theorem 3. It may happen that this stopping criterion is not met when (u^3, v^3) belongs to the boundary of U^*. But, by Theorem 3 akin, the stopping criterion will be met at any $(u^4, v^4) > (u^3, v^3)$, i.e., $(u^4, v^4) \in \text{int}(U^*)$. This is the motivation for Step 4. In our computational experience (see Section 5) we have never attained Step 4 either because we obtained a primal optimal point in Step 3 or because the algorithm exceeded the allowed CPU time.

In Step 2.2 we first use a simple heuristic method (*Heuristic* 1). If Heuristic 1 does not close the primal-dual gap, then we use a second and more sophisticated heuristic method (*Heuristic* 2). As Heuristic 1 we use the following simple method. We distinguish two cases after computing (x^2, y^2): (a) If the number of open medians is less than p, say p', then we set as new medians the $p - p'$ most expensive customers. (b) If the number of open medians is greater than p, say p'', then we close the $p'' - p$ medians with least number of assigned customers. We reassign these customers to their closest open median. As Heuristic 2 we use the 'Variable Neighborhood Decomposition Search' (VNDS) [13].

4.1. *Solving the dual problems*

The point is now how to solve the dual problems

$$\max_u \mathcal{L}_r(u) \quad r = 1, 2, 3, \tag{11}$$

associated to the semi-Lagrangian relaxation. For the ease of notation we drop the v component of $\mathcal{L}_r(u, v)$, $r = 1, 2, 3$, with no loss of generality. Functions $\mathcal{L}_r(u)$, $r = 1, 2, 3$, are implicitly defined as the pointwise minimum of linear functions in u. By construction they are concave and nonsmooth. Extensive numerical experience shows that ACCPM and in particular Proximal-ACCPM, is an efficient tool for solving (11). See, for instance, [10] and references therein included; see also [6] for experiments with the linear relaxation of the p-median problem.

In the cutting plane procedure, we consider a sequence of points $\{u^k\}_{k \in K}$ in the domain of $\mathcal{L}(u)$ (for ease of notation we drop the r index of $\mathcal{L}_r(u)$). We denote by s^k a subgradient of $\mathcal{L}(u)$ at u^k, that is, $s^k \in \partial\mathcal{L}(u^k)$, the subdifferential of $\mathcal{L}(u)$ at u^k (properly speaking, we should use the terminology *supergradient* and *superdifferential* given that $\mathcal{L}(u)$ is concave). We consider the linear approximation to $\mathcal{L}(u)$ at u^k, given by $\mathcal{L}^k(u) = \mathcal{L}(u^k) + s^k \cdot (u - u^k)$ and have

$$\mathcal{L}(u) \leq \mathcal{L}^k(u)$$

for all u.

The point u^k is referred to as a *query point*, and the procedure to compute the objective value and subgradient at a query point is called an *oracle*. Furthermore, the hyperplane that approximates the objective function $\mathcal{L}(u)$ at a feasible query point and defined by the equation $z = \mathcal{L}^k(u)$, is referred to as an *optimality cut*.

A lower bound to the maximum value of $\mathcal{L}(u)$ is provided by:

$$\theta_l = \max_k \mathcal{L}(u^k).$$

The *localization set* is defined as

$$L = \{(u, z) \in \mathbb{R}^{n+1} \mid u \in \mathbb{R}^n, \quad z \leq \mathcal{L}^k(u) \quad \forall k \in K, \quad z \geq \theta_l\}. \tag{12}$$

The basic iteration of a cutting plane method can be summarized as follows

1. Select $(\bar{u}, \bar{z})$ in the localization set L.
2. Call the oracle at $\bar{u}$. The oracle returns one or several optimality cuts and a new lower bound $\mathcal{L}(\bar{u})$.
3. Update the bounds:

   (a) $\theta_l \leftarrow \max\{\mathcal{L}(\bar{u}), \theta_l\}$.
   (b) Compute an upper bound θ_u to the optimum [1] of problem (11).

4. Update the lower bound θ_l in the definition of the localization set (12) and add the new cuts.

These steps are repeated until a point is found such that $\theta_u - \theta_l$ falls below a prescribed optimality tolerance. The reader may have noticed that the first step in the summary is not completely defined. Actually, cutting plane methods essentially differ in the way one chooses the query point. For instance, the intuitive choice of the Kelley point $(\bar{u}, \bar{z})$ that maximizes z in the localization set [18] may prove disastrous, because it overemphasizes the global approximation property of the localization set. Safer methods, as for example bundle methods [15] or Proximal-ACCPM [6, 9, 10], introduce a regularizing scheme to avoid selecting points too "far away" from the best recorded point. In this paper we use Proximal-ACCPM (*Proximal Analytic Center Cutting Plane Method*) which selects the *proximal analytic center* of the localization set. Formally, the proximal analytic center is the point $(\bar{u}, \bar{z})$ that minimizes the logarithmic barrier function [2] of the localization set plus a quadratic proximal term which ensures the existence of a unique minimizer.[3] This point is relatively easy to compute using the standard artillery of Interior Point Methods. Furthermore, Proximal-ACCPM is robust, efficient and particularly useful when the oracle is computationally costly—as is the case in this application.

4.2. *Solving the oracle $\mathcal{L}_2$*

Step 2 of the cutting plane method calls the oracle $\mathcal{L}$ at $\bar{u}$. This amounts to compute $\mathcal{L}(\bar{u})$ and one $s \in \partial\mathcal{L}(\bar{u})$. In our case, this implies to solve the relaxed problems of Section 3 (*solve the oracles*, in our terminology). Solving the oracle $\mathcal{L}_2$ is by no means a trivial matter. However, problem (9) has many interesting features that makes it possible to solve by a direct approach with an efficient solver such as CPLEX. We can reduce the size of the problem and decompose it by taking into account the following three observations.

Our first observation is that all variables x_{ij} with reduced cost $c_{ij} - u_j \geq 0$ are set to zero (or simply eliminated). Associated to the p-median problem, there is a underlying graph with one link (i, j) connecting facility i with customer j, which has a positive cost c_{ij}. After the Lagrangian relaxation, the p-median graph may become very sparse, since only links (i, j) with negative reduced costs are kept in the graph.

Our second observation is that the above elimination of links in the p-median graph, not only reduces the problem size, but, may break the p-median graph into K smaller independent subgraphs. In that case, to compute $\mathcal{L}_2(u, v)$ we end up solving K independent subproblems. The important point is that the union of all these problems is much easier for CPLEX to solve than the larger instance collecting all the smaller problems into a single one. It seems that CPLEX does not detect this decomposable structure, while it is easy for the user, and almost costless, to generate the partition.

Our third observation is that if a column $\bar{j}$ in the array $\{(c_{ij} - u_j)^-\}$ of negative reduced costs has a single negative entry $c_{\bar{\imath}\bar{j}} - u_{\bar{j}}$, then we may enforce the equality $x_{\bar{\imath}\bar{j}} = y_{\bar{\imath}}$.

4.3. *Solving the oracle $\mathcal{L}_3$*

Solving (10) is more challenging, though it just suffices to add the constraint $\sum_i y_i = p$ to (9). This certainly makes the problem more difficult for CPLEX. Moreover, this constraint links all blocks in the graph partition discussed above. This is particularly damaging if the partition contains many small blocks. When this situation occurs, it often appears that the solution (x^2, y^2) in Step 2 of our algorithm violates the constraint (10c) by few units, say for example 3, i.e.,

$$\sum y_i^2 = p + 3.$$

Let $I = \{1, \ldots, n\} = \cup_{k=1}^K I_k$ be the partition resulting from the graph decomposition. (Note that one set, say I_K may collect all the indices of rows of the reduced cost matrix with no negative entry.) Let $p_k = \sum_{i \in I_k} y_i^2$. For each k we solve the subproblem associated with graph I_k, with the added constraint

$$\sum_{i \in I_k} y_i^2 \leq b_k,$$

for $b_k = p_k,\ p_k - 1,\ p_k - 2,\ p_k - 3$ (if b_k becomes zero or negative, we do not solve the corresponding subproblem). We then solve a knapsack auxiliary problem to combine the solutions of the independent blocks to generate an optimal solution to (10) (such that $\sum y_i^2 \leq p$.).

5. Numerical experiments

The objective of our numerical experiments is threefold: first we whish to study the influence of using a good starting point to maximize $\mathcal{L}_2$, second we study the solution quality of the semi-Lagrangian relaxation and third we will study its performance.

To test the semi-Lagrangian relaxation we use data from the *traveling salesman problem* (TSP) library [25], to define p-median instances, as already used in the p-median literature [6], especially in [2]. In the tables of this paper the name of the instance indicates the number of customers (e.g. vm1748 corresponds to a p-median instance with 1748 customers).

Programs have been written in MATLAB 6.1 [14] and run in a PC (Pentium-IV, 2.4 GHz, with 6 Gb of RAM memory) under the Linux operating system. The program that solves Oracle 1 has been written in C. To solve Oracles 2 and 3 we have intensively used CPLEX 8.1 (default settings) interfaced with MATLAB [23, 26]. To make our approach as general as possible, we have used the same set of parameters for Proximal-ACCPM in all the instances.

5.1. *Influence of the starting point*

Considering that Oracle 2 is a strengthened version of Oracle 1, our hypothesis is that the set of dual optimizers associated to Oracle 2 may be close to the optimal set associated to Oracle 1. As a matter of fact, in our tests we have observed that the more accurate the dual optimizer associated to Oracle 1, the easier the solving of the Oracle 2 dual problem. To illustrate this empirical observation we display the results obtained for a set of 10 medium p-median instances with data from the TSP library.

We compare the results obtained by using two different starting points for Oracle 2. In the first approach we use, as starting point, a low accuracy optimal point obtained by using Oracle 1 (Proximal-ACCPM stopping criterion threshold equal to 10^{-3}). In the second approach we use 10^{-6}. In the two cases the maximum number of Oracle 1 iterations has been set equal to 500. We also set a limit of 30000 seconds for the CPU time.

In Table 1 we have the optimal values and in Table 2 the number of iterations and CPU time in seconds. We can observe that the extra iterations spent to compute an accurate starting point for the Oracle 2 is largely compensated by cutting down the number of very expensive Oracle 2 iterations. On average the ‘High accuracy’ approach is over six times faster (12030/1937) than the ‘Low accuracy’ one. In Table 2 we observe that instances vm1748 need a great amount of CPU time compared to the other instances. As we will see in Section 5.3, the CPU time strongly depends on the ANSO2 parameter and we also will see that instances vm1748 have a disadvantageous ANSO2 value.

Table 1. Starting point accuracy: Optimal values and relative increase in %. Labels 'Low accuracy' and 'High accuracy' correspond to use 10^{-3} and 10^{-6} respectively, in the stopping criterion when solving Oracle 1.

Instance		Low accuracy			High accuracy		
Label	p	Oracle 1	Oracle 2	%	Oracle 1	Oracle 2	%
r11304	100	491356.2	491639	0.06	491487.3	491639	0.03
rl1304	300	177270.5	177326	0.03	177317.4	177326	0.00
rl1304	500	96986.4	97024	0.04	97008.9	97024	0.02
fl1400	100	15946.3	15962	0.10	15960.6	15962	0.01
fl1400	200	8787.4	8806	0.21	8792.2	8806	0.16
u1432	20	588424.5	588766	0.06	588719.7	588766	0.01
vm1748	10	2979175.9	2983645	0.15	2982731	2983645	0.03
Vm1748	20	1894608.6	1899152[a]	0.24	1898775.2	1899680	0.05
vm1748	50	1002392.9	1004331	0.19	1004205.2	1004331	0.01
vm1748	100	635515.8	636515	0.16	636324.1	636515	0.03
Average		789046.4	790317	0.12	790132.2	790369	0.03

[a]Optimal value not attained because the maximum CPU time was reached while solving Oracle 2.

Table 2. Starting point accuracy: Iterations and CPU time (seconds). Labels 'Low accuracy' and 'High accuracy' correspond to use 10^{-3} and 10^{-6} respectively, in the stopping criterion when solving Oracle 1.

Instance		Low accuracy			High accuracy		
Label	p	Iter. Or. 1	Iter. Or. 2	CPU (s)	Iter. Or. 1	Iter. Or. 2	CPU (s)
r11304	100	124	44	429	256	40	268
r11304	300	88	16	26	161	8	20
r11304	500	90	16	67	133	15	48
fl1400	100	107	15	886	442	13	572
fl1400	200	121	14	877	500	16	916
u1432	20	126	15	1548	346	9	192
vm1748	10	235	41	27774	500	21	3945
vm1748	20	220	100	82424	500	38	10768
vm1748	50	154	51	3675	462	19	551
vm1748	100	133	60	2596	500	40	2085
Average		140	30	12030	380	20	1937

5.2. 'Easier' instances

By 'easier' TSP instances we mean the instances that in [2] required less than 7500 seconds to be solved. The remaining instances, which required at least 28000 seconds, are called 'difficult' and studied in Section 5.3. In this section we solve the 'easier'TSP instances. As we can see in Table 3, easier instances range form 1304 to 3795 customers and each problem is solved for different values of p. By no means these easier instances are easy since commercial solvers, as CPLEX, are able to solve instances up to 400 customers. The maximal CPU time CPU_{max} for each instance is set equal to the minimum between: ten times the reported CPU time in [2] for each case and 30000 seconds. The maximal number of Oracle 1 iterations is set equal to 500.

The main two factors to evaluate are first, the quality of the solutions, as expressed by the optimality gap and second, the computing time. Let us discuss first the issue of the quality of the solutions computed by the semi-Lagrangian relaxation. In Table 3 we can observe that in general the semi-Lagrangian relaxation gives tighter dual bounds than the Lagrangian relaxation. On average, the Oracle 2 lower bounds are 0.03% larger than the Oracle 1 ones.

In most cases (see Tables 3 and 4) the procedure stops with an optimal integer solution obtained by Oracle 2. In few cases, the time limit is reached while solving Oracle 2. The use of a heuristic yields a bound on the integrality gap. We notice that the percentage of optimality is higher than 98.98%. The remaining cases concerns the use of Oracle 3. Indeed, Oracle 2 sometimes produces an integer solution that is feasible for all constraints but $\sum y_i = p$. If the heuristic does not produce an optimal integer solution, then we must resort to Oracle 3 which corresponds to the full semi-Lagrangian relaxation. Then, on the easier instances Oracle 3 always terminates with an optimal solution.

In summary, 27 of the 33 ‘easier’ instances (82%) are solved up to optimality by the semi-Lagrangian approach (label SLR in column ‘Upper bound/Method’). In the remaining instances (18%) we stopped the semi-Lagrangian procedure because of an excess of CPU time. Nevertheless, these instances are almost completely solved by computing a quasi optimal primal solution by using the VNDS heuristic [13]. For these instances, the solution quality is no worse than 98.98% of optimality gap. In [2] all these instances are fully solved (solution quality equal to 100% in all cases).

The performance of the semi-Lagrangian relaxation for this test can be found in Table 4. On average, the number of Proximal-ACCPM iterations is 378, 19 and 0.4, for the Oracles 1, 2 and 3 respectively. As we have seen in Section 5.1, the use of an effective convex optimization solver, as Proximal-ACCPM, is important to limit the number of calls to the very expensive Oracle 2 (19 calls on average). Oracle 3 is called 2 times at most. A possible explanation for this low number of Oracle 3 calls, is that $\mathcal{L}_2(u, v)$ and $\mathcal{L}_3(u, v)$ only differ in constraint (10c), and therefore it is likely that the respective associated optimal sets are similar. Furthermore, as we have seen in Corollary 2.1, the optimal set of $\mathcal{L}_3(u, v)$, i.e. U^*, is an unbounded set and therefore it should not take too many iterations to find one of the infinitely many optimal solutions, once we are close to U^*.

On average, the CPU time is 108 seconds, 2283 seconds and 464 seconds for the Oracles 1, 2 and 3 respectively, which shows that most of the time corresponds to Oracle 2. The average CPU time for the semi-Lagrangian relaxation is 2859 seconds which is 3.62 times[4] the averaged CPU time reported in [2]. In our opinion, the main reason that explains this difference in performance is that in [2] the polyhedral structure of the p-median problem is exploited in a a branch-cut-and-price (BCP) algorithm. In contrast, our algorithm is based on the semi-Lagrangian relaxation, a general purpose simple method independent of the p-median problem.

Table 3. Easier instances: Solution quality. Symbols: 'Or.' stands for Oracle, 'VNDS' for variable neighborhood decomposition search, 'SLR' for semi-Lagrangian relaxation. '% Optimality' is computed as 100 × [1−('optimal primal value'—'optimal dual value')/'optimal dual value'].

Instance		Lower bounds			Upper bound		Optimality
Label	p	Or. 1	Or. 2	Or. 3	Value	Method	(%)
rl1304	10	2131787.5	2133534	–	2134295	VNDS	99.96
rl1304	100	491487.3	491639	–	491639	SLR	100
rl1304	300	177317.4	177326	–	177326	SLR	100
rl1304	500	97008.9	97024	–	97024	SLR	100
fl1400	100	15960.6	15962	15962	15962	SLR	100
fl1400	200	8792.2	8806	8806	8806	SLR	100
u1432	20	588719.7	588766	–	588766	SLR	100
u1432	100	243741.0	243793	–	243793	SLR	100
u1432	200	159844.0	159885	–	160504	VNDS	99.61
u1432	300	123660.0	123689	123689	123689	SLR	100
u1432	500	93200.0	–	–	93200	VNDS	100
vm1748	10	2982731.0	2983645	–	2983645	SLR	100
vm1748	20	1898775.2	1899390	–	1899680	VNDS	99.98
vm1748	50	1004205.2	1004331	–	1004331	SLR	100
vm1748	100	636324.1	636515	–	636515	SLR	100
vm1748	300	286029.5	286039	–	286039	SLR	100
vm1748	400	221522.2	221526	–	221526	SLR	100
vm1748	500	176976.2	176986	176986	176986	SLR	100
d2103	10	687263.3	687321	–	687321	SLR	100
d2103	20	482794.9	482926	–	482926	SLR	100
d2103	200	117730.8	117753	–	117753	SLR	100
d2103	300	90417.2	90471	90471	90471	SLR	100
d2103	400	75289.3	75324	75324	75324	SLR	100
d2103	500	63938.4	64006	64006	64006	SLR	100
pcb3038	5	1777657.0	1777677	–	1777835	VNDS	99.99
pcb3038	100	351349.1	351461	–	353428	VNDS	99.44
pcb3038	150	280034.0	280128	–	280128	SLR	100
pcb3038	200	237311.0	237399	–	237399	SLR	100
pcb3038	300	186786.4	186833	–	186833	SLR	100
pcb3038	400	156266.6	156276	–	156276	SLR	100
pcb3038	500	134771.8	134798	134798	136179	VNDS	98.98
fl3795	400	31342.5	31354	–	31354	SLR	100
fl3795	500	25972.0	25976	25976	25976	SLR	100
Average[a]		498243.9	498392		498554		99.94

[a] Average figures do not take into account problem u1432 with $p = 500$.

Table 4. Easier instances: Performance. Symbols: 'Or.' stands for Oracle, 'ANSO2' for average number of subproblems per Oracle 2 call, (*)'Total CPU time' includes 100 seconds of the VNDS heuristic.

Instance		Oracle calls				CPU time (seconds)				
Label	p	Or. 1	Or. 2	Or. 3	ANSO2	Or. 1	Or. 2	Or. 3	SLR	BCP[a]
rl1304	10	390	35	0	1	95	17141	0	(*)17336	1614
rl1304	100	256	40	0	11	26	242	0	268	40
rl1304	300	161	8	0	69	11	9	0	20	18
rl1304	500	133	15	0	143	8	40	0	48	20
fl1400	100	442	13	2	28	89	483	558	1130	378
fl1400	200	500	16	2	48	119	797	5384	6300	191
u1432	20	346	9	0	1	54	138	0	192	101
u1432	100	297	21	0	2	36	940	0	976	119
u1432	200	500	21	0	5	114	1219	0	(*)1433	58
u1432	300	500	26	1	19	112	310	78	500	43
u1432	500	158	2	0	11	11	362	0	(*)473	36
vm1748	10	500	21	0	1	174	3771	0	3945	478
vm1748	20	500	23	0	1	162	5045	0	(*)5307	341
vm1748	50	462	19	0	2	152	399	0	551	36
vm1748	100	500	40	0	4	157	1928	0	2085	136
vm1748	300	230	25	0	51	30	39	0	69	24
vm1748	400	158	7	0	93	16	24	0	40	155
vm1748	500	146	15	2	131	14	61	22	97	74
d2103	10	241	7	0	2	41	504	0	545	260
d2103	20	389	11	0	2	109	2682	0	2791	733
d2103	200	500	27	0	20	155	2178	0	2333	1828
d2103	300	500	23	1	26	146	1465	535	2146	1133
d2103	400	500	17	1	37	145	618	1426	2189	235
d2103	500	500	26	2	39	143	10086	4309	14538	5822
pcb3038	5	341	5	0	1	111	1888	0	(*)2099	1114
pcb3038	100	464	21	0	2	188	32714	0	(*)33002	7492
pcb3038	150	500	11	0	2	202	11292	0	11494	3057
pcb3038	200	500	12	0	5	201	11792	0	11993	2562
pcb3038	300	446	16	0	12	166	3853	0	4019	2977
pcb3038	400	330	14	0	24	99	2874	0	2973	454
pcb3038	500	211	17	2	38	56	3269	3800	(*)7225	704
fl3795	400	500	24	0	26	254	2218	0	2472	6761
fl3795	500	500	38	1	25	259	2531	218	3008	770
[b]Average		378	19	0.4	31	108	2283	464	2859	1053

[a]Times were obtained with a processor Pentium IV-1.8 GHz. We use a Pentium IV-2.4 GHz.
[b]Average figures do not take into account the 5 problems not solved up to optimality (see Table 3).

Each time we call Oracle 2, we solve a relaxed facility location problem. For most Oracle 2 calls the underlying graph is disconnected and then CPLEX solves as many subproblems as the number of graph components. In column 'ANSO2' we have the 'average number of subproblems per Oracle 2 call'. The averaged ANSO2 parameter is 31 subproblems, that is, on average, each time we call Oracle 2, we solve 31 independent combinatorial subproblems.

In general, the difficulty to solve the Oracle 2 increases with the problem size but decreases with the ANSO2 parameter. Clearly ANSO2 is a critical parameter. Thus for example in Tables 3 and 4 we can see that our procedure fails to completely solve the smallest reported problem (rl1304 with $p = 10$) within the time limit (ANSO2 = 1). Nevertheless, the computed solution is 99.96% optimal. On the other extreme, one of the biggest reported instances (fl3795) is fully solved by our procedure (ANSO2 $\geq$ 17 for all the cases).

5.3. *Difficult instances*

In the previous section we have seen that the semi-Lagrangian relaxation does not outperform state-of-art BCP approaches. Nevertheless, in this section we will see that for the instances not solved by [2] (we call them *difficult instances*), the performance of the SLR procedure is similar. The maximal CPU time and maximal number of Oracle 1 calls are set equal to 360000 seconds and 1000 calls, respectively.

Regarding the quality of the (dual) lower bounds, Table 5 shows that except for problem u1432, the SLR lower bounds are equal or tighter than the BCP bounds. Regarding the quality of the best integer solution, Table 6 shows that in cases of partial optimality, the heuristic used in [2] outperforms heuristics we have used (H1 and VNDS). No method has solved up to optimality problem fl1400 with $p = 500$, but our lower bound 3764 combined with the upper bound in [2] solves the problem.

Table 5. Difficult instances: Lower bounds.

Instance		Lower bounds				
Label	p	Or. 1	Or. 2	Or. 3	SLR	BCP
fl1400	300	6091.8	6109	6109	6109	6099
fl1400	400	4635.4	4648	–	4648	4643
fl1400	500	3755.7	3764	3764	3764	3760
u1432	50	361683.6	362005	362057	362057	362072
u1432	400	103403.8	103623	–	103623	103783
d2103	50	301565.4	301705	–	301705	301618
d2103	100	194390.1	194495	–	194495	194409
fl3795	150	65837.6	65868	–	65868	65868
fl3795	200	53908.9	53928	53928	53928	53928
fl3795	300	39576.7	39586	–	39586	39586
Average		113484.9	113573		113578	113577

Table 6. Difficult instances: Best integer solution.

Instance		Best integer solution			% Optimality	
Label	p	Method	SLR	BCP	SLR	BCP
fl1400	300	VNDS	6146	6111	99.39	99.80
fl1400	400	H1	4648	4648	100	99.89
fl1400	500	H1	3765	3764	99.97	99.89
u1432	50	VNDS	362100	362072	99.97	100
u1432	400	VNDS	104735	103979	98.93	99.81
d2103	50	VNDS	302916	302337	99.60	99.76
d2103	100	VNDS	195273	194920	99.60	99.74
fl3795	150	SLR	65868	65868	100	100
fl3795	200	SLR	53928	53928	100	100
fl3795	300	SLR	39586	39586	100	100
Average			113897	113721	99.74	99.89

Table 7. Difficult instances: Oracle calls and average number of subproblems in Oracle 2.

Instance		Oracle calls			
Label	p	Or. 1	Or. 2	Or. 3	ANSO2
fl1400	300	261	16	1	82
fl1400	400	196	17	0	100
fl1400	500	218	16	1	145
u1432	50	423	26	15	1
u1432	400	1000	16	0	29
d2103	50	741	2	0	3
d2103	100	1000	5	0	8
fl3795	150	1000	27	0	17
fl3795	200	1000	32	3	16
fl3795	300	1000	15	0	26
Average		684	17	2	43

Tables 7 and 8 display the number of oracle calls, the average number of subproblems per Oracle 2 call (ANSO2) and the computing times. Both methods have fully solved four of the ten difficult problems. Even considering that our computer is about 33% faster, it is remarkable the SLR time for instances fl3795 and especially instance fl1400 ($p = 400$). The reason for this very good performance probably is the high ANSO2 parameter. However, a high ANSO2 parameter is not enough to guarantee a good SLR performance. See for example the unsolved instance fl400 ($p = 500$) which has the highest ANSO2 parameter (145).

Table 8. Difficult instances: CPU time.

Instance		CPU time (seconds)				
Label	p	Or. 1	Or. 2	Or. 3	SLR	BCP
fl1400	300	27	2632	357341	360000	360000
fl1400	400	16	652	0	678	360000
fl1400	500	20	261715	98265	360000	360000
u1432	50	80	135331	224589	360000	28257
u1432	400	856	359144	0	360000	360000
d2103	50	476	359524	0	360000	360000
d2103	100	928	359072	0	360000	360000
fl3795	150	1100	39199	0	40299	346396
fl3795	200	1459	32975	30125	64559	84047
fl3795	300	1266	2264	0	3530	53352
Average		623	155251	71032	226907	267205

6. Conclusion

In this paper we have studied the semi-Lagrangian relaxation (SLR) which applies to combinatorial problems with equality constraints. In theory it closes the integrality gap and produces an optimal integer solution. This approach has a practical interest if the relaxed problem (oracle) becomes more tractable than the initial one.

We applied this approach to the p-median problem. In this case, the oracle is much easier than the original problem: its size is drastically reduced (most variables are automatically set to zero) and it is often decomposable. The SLR approach has solved most of the tested large-scale p-median instances exactly. This fact is quite remarkable. Of course, the oracle for the strong relaxation is difficult and time consuming, but it is still easy enough to be tackled directly by a general purpose solver such as CPLEX. In sharp contrast, CPLEX is unable to solve these problems in their initial formulation if the size exceeds 400 customers.

The SLR for the p-median problem has two interesting features from a computational point of view: First, if the oracle is decomposable, the SLR is easily parallelizable (one combinatorial subproblem per processor). Second, in contrast with sophisticated branch-cut-and-price implementations, the implementation of SLR is easy, since the complex tasks are all performed by standard tools (CPLEX and Proximal-ACCPM). Any improvement on these tools or similar, could be incorporated without further programming effort.

On the other hand, the relative ease in solving the integer programming subproblems is not sufficient to allow the use of the convex optimization solvers that are commonly used in connection with the standard Lagrangian relaxation or column generation scheme. We believe that a subgradient method or Kelley's cutting plane method, would entail too many queries to the oracle to make the approach workable. The use of an advanced convex solver, such as Proximal-ACCPM, is a must; it turns out that this solver is efficient enough to solve in a short time all the instances we have examined.

This approach opens the road for further investigations. The SLR of the p-median problem is a special variant of the uncapacitated facility location (UFL) problem, with a profit maximizing objective and the additional property that not all customers need to be served. Moreover, the underlying graph of this UFL may be (massively) sparse and disconnected. Those characteristics are attractive enough to justify the search for a dedicated exact algorithm and for powerful and fast heuristics. Progresses in that direction could make the SLR approach more competitive.

Acknowledgments

We thank L. A. Wolsey for pointing out the reference to Klabjan's work and for a short proof of Theorem 1. We also thank P. Hansen, N. Mladenovic and D. Perez-Brito for making available to us the VNDS p-median FORTRAN code [13]. Finally we thank the referees for their comments which have improved the readability of the paper.

Notes

1. For example, $\theta_u = \max\{z \mid (u, z) \in L \cap D\}$ where D is a compact domain defined for example by a set of lower and upper bounds for the components of u.
2. The logarithmic barrier for the half space $\{u \in \mathbb{R}^n \mid a \cdot u \leq b\}$ is $-\log(b - a \cdot u)$.
3. That is, the proximal analytic center of L is the point

$$(\bar{u}, \bar{z}) = \underset{u,z}{\operatorname{argmin}} F_L(u, z) + \rho\|u - \hat{u}\|^2$$

where $F_L(u, z)$is the logarithmic barrier for the localization set L, ρ is the proximal weight, and $\hat{u}$ is the proximal point (current best point).

4. This time ratio has been scaled taking into account that in [2] the authors use a Pentium IV-1.8 GHz, whereas we use a Pentium IV-2.4 GHz, that is, $(2859/1053) \times (2.4/1.8) = 3.62$. It has also to be taken into account that in [2] the code was written in C, a compiled language, whereas our code is written in Matlab, an interpreted language (although our oracles are also coded in C).

References

1. P. Avella and A. Sassano, "On the p-median polytope," Mathematical Programming, vol. 89, pp. 395–411, 2001.
2. P. Avella, A. Sassano, and I. Vasil'ev, "Computational study of large-scale p-median problems," Technical Report, Dipartimento Di Informatica e Sistemistica, Università di Roma "La Sapienza", 2003.
3. O. Briant and D. Naddef, "The optimal diversity management problem," Operations Research, vol. 52, no. 4, 2004.
4. Christofides, Graph Theory: An Algorithmic Approach. Academic Press, New York, 1975.
5. I.R.J. de Farias, "A family of facets for the uncapacitated p-median polytope," Operations Research Letters, vol. 28, pp. 161–167, 2001.
6. O. du Merle and J.-P. Vial, "Proximal-ACCPM, a cutting plane method for column generation and lagrangian relaxation: Application to the p-median problem," Technical Report, Logilab, HEC, University of Geneva, 2002.

7. H. Everett III, “Generalized lagrange multiplier method for solving problems of optimum allocation of resources,” Operations Research, vol. 11, no. 3, pp. 399–471, 1963.
8. A.M. Geoffrion, “Lagrangean relaxation for integer programming,” Mathematical Programming Study, vol. 2, pp. 82–114, 1974.
9. J.L. Goffin, A. Haurie, and J.P. Vial, “Decomposition and nondifferentiable optimization with the projective algorithm,” Management Science, vol. 37, pp. 284–302, 1992.
10. J.-L. Goffin and J. Vial, “Convex nondifferentiable optimization: A survey focussed on the analytic center cutting plane method,” Technical Report 99.02, Geneva University—HEC—Logilab, 1999.
11. M. Guignard and S. Kim, “Lagrangean decomposition: A model yielding stronger Lagrangean bounds,” Mathematical Programming, vol. 39, pp. 215–228, 1987.
12. P. Hansen and B. Jaumard, “Cluster analysis and mathematical programming,” Mathematical Programming, vol. 79, pp. 191–215, 1997.
13. P. Hansen, N. Mladenovic, and D. Perez-Brito, “Variable neighborhood decomposition search,” Journal of Heuristics, vol. 7, pp. 335–350, 2001.
14. D.J. Higham and N.J. Higham, MATLAB Guide, SIAM, Philadelphia, Pennsilvania, USA, 2000.
15. J.B. Hiriart-Urruty and C. Lemaréchal, Convex Analysis and Minimization Algorithms, volume I and II. Springer-Verlag, Berlin, 1996.
16. E. Johnson, “Mathematical programming,” Chapter cyclic groups, cutting planes and shortest path, Academic press, pp. 185–211, 1973.
17. O. Kariv and L. Hakimi, “An algorithmic approach to network location problems. ii: The p-medians,” SIAM Journal of Applied Mathematics, vol. 37, no. 3, pp. 539–560, 1979.
18. J.E. Kelley, “The cutting-plane method for solving convex programs,” Journal of the SIAM, vol. 8, pp. 703–712, 1960.
19. D. Klabjan, “A new subadditive approach to integer programming,” in W. Cook and A.S. Schulz, (eds.), Integer Programming and Combinatorial Optimization, 9th International IPCO Conference, Cambridge, MA, USA, May 27–29, 2002, Proceedings, volume 2337 of Lecture Notes in Computer Science. Springer, 2002.
20. D. Klabjan, “A practical algorithm for computing a subadditive dual function for set partitioning,” Computational Optimization and Applications, vol. 29, pp. 347–368, 2004.
21. C. Lemaréchal and A. Renaud, “A geometric study of duality gaps, with applications,” Mathematical Programming, Ser. A, vol. 90, pp. 399–427, 2001.
22. J.M. Mulvey and H.P. Crowder, “Cluster analysis: An application of lagrangian relaxation,” Management Science, vol. 25, pp. 329–340, 1979.
23. D.R. Musicant, “Matlab/cplex mex-files,” 2000. www.cs.wisc.edu/∼musicant/data/cplex/.
24. G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization, John Wiley and Sons, 1988.
25. G. Reinelt, “Tsplib,” 2001. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.
26. C. Tadonki, “Using cplex with matlab,” 2003. http://www.omegacomputer.com/staff/tadonki/using_cplex_with_matlab.htm.