Solving the uncapacitated facility location problem with semi-Lagrangian relaxation[*]

C. Beltran-Royo[†] J.-Ph. Vial[‡] A. Alonso-Ayuso[§]

March 1, 2007

Abstract

The Semi-Lagrangian Relaxation (SLR) method has been introduced in [BTV06] to solve the p-median problem. In this paper we apply the method to the Uncapacitated Facility Location (UFL) problem. We perform computational experiments on two main collections of UFL problems with unknown optimal values. On one collection, we manage to solve to optimality 16 out of the 18 instances. On the second collection we solve 2 out of 18 instances and improve the Lagrangian lower bound. In this way, we prove that the Hybrid Multistart heuristic of [RW06] provides near optimal solutions.

Keywords: Lagrangian relaxation, combinatorial optimization, uncapacitated facility location (UFL) problem.

1 Introduction

The Semi-Lagrangian Relaxation (SLR) method has been introduced in [BTV06] to solve the p-median problem. SLR applies to problems with equality constraints. Like Lagrangian Relaxation (LR), the equality constraints are relaxed, but the definition of the semi-Lagrangian dual problem incorporates those constraints under the weaker form of inequalities. On combinatorial problems with positive coefficients, it has the strong property of achieving a zero duality gap. The method has been used with success to solve large instances of the p-median problem. In this paper we revisit the method and apply it to the celebrated Uncapacitated Facility Location (UFL) problem. We propose two different approaches to solve the semi-Lagrangian dual problem and perform computational experiments on two main collections of UFL problems with unknown optimal values. On one collection, we manage to solve to optimality 16 out of the 18 problems. On the second collection we only solve 2 out of 18 instances. Nevertheless, we are able to improve the Lagrangian lower bound in this collection and thereby confirm that the

[*]This work was partially supported by the Spanish government, MCYT subsidies TIC2003-05982-c05-05 and MTM2006-14961-C05-05.

[†]Corresponding author, cesar.beltran@urjc.es, Statistics and Operations Research, Rey Juan Carlos University, Madrid, Spain.

[‡]jpvial@ordecsys.com, Profesessor Emeritus, University of Geneva (Switzerland) and Director of ORDECSYS, www.ordecsys.com.

[§]antonio.alonso@urjc.es, Statistics and Operations Research, Rey Juan Carlos University, Madrid, Spain.

Hybrid Multistart heuristic of [RW06] provides near optimal solutions (over 99% optimal in most cases).

The UFL problem (also called the simple plant location problem) is a basic but important problem in location science [NW88]. Due to this importance many works were published in the period 1960-1990. See [MF90] for a survey. This period culminates with the paper [Koe89], which represents the state of the art regarding the UFL exact solving methods. [Koe89] improves the famous dual-based procedure of Donald Erlenkotter [Erl78], where the (condensed) dual of the LP relaxation for the UFL problem is heuristicaly maximized by the coordinate ascent method. This procedure is followed by a Branch-and-Bound if necessary. [CC90] propose an projection method to improve the coordinate ascent method in order to compute an exact optimum of the condensed dual problem.

On the other hand, after 1990 contributions to the UFL problem mainly dealt with heuristic methods. Some of the most successful methods are: Tabu search in [Gho03], a combination of Tabu search, Genetic Algorithm and Greedy Randomized Adaptative Search in [KI05] and Hybrid Multistart heuristic in [RW06].

Lagrangian relaxation [Gui03] has also been used to solve the UFL problem, though with less success than the LP dual-based procedures. For exemple [Gui88] proposes to strengthen the Lagrangian relaxation (equivalent to the LP relaxation) by using Benders inequalities. From a more theoretical point of view, [MBH06] studies the duality gap of the UFL problem. A complete study of valid inequalities, facets and lifting theorems for the UFL problem can be found in [CJPR83, CPR83].

In the standard Lagrangian approach to UFL, the equality constraints are dualized. The value of this dual function is obtained by solving a linear programming subproblem, which, in the case of UFL, is separable and simple enough to be solved by inspection. In SLR, the equality constraints are dualized as before, but they also enter the definition of the subproblem under the weaker form of inequalities. This makes the subproblem substantially more difficult than in the standard approach. In particular, the problem is no more separable and it is even NP-hard. However, it is well-known that making the subproblem more constrained reduces the duality gap. In the SLR, it even fully closes the gap. The main issue is thus our ability to obtain exact solutions for the subproblems in a reasonable time. This turns out to be possible on many problem instances because the SLR subproblems have a very interesting property: their dimension is usually much smaller than the dimension of the original problem. Therefore, they can be solved by integer programming solvers that are currently available. In our case, we use CPLEX.

A main issue of SLR is to keep the dimension of the SLR subproblems as low as possible, so as to make them as easy as possible. To this end, we use and compare two different approaches to solve the semi-Lagrangian dual problem (the so-called master program): a variant of the dual ascent method (see Section 3.3 for details) and Proximal-ACCPM [BBH$^+$06]. The dual ascent method we use is a variant of the *dual multi-ascent* procedure designed in [Koe89] to solve the Erlenkotter's 'condensed' dual of the UFL problem [Erl78]. The method increases the dual variables by finite amounts according to the local subgradient information. The way it is done allows convergence to an optimal solution while keeping the number of subproblem variables fixed to zero as large as possible. We prove finite convergence for the method in the SLR case, whereas the dual multi-ascent method does not necessarily converge in the LR case. The other method is the Proximal Analytic Center Cutting Plane Method (Proximal-ACCPM), which is a generic tool for convex nondifferentiable optimization. With a proper setting of the proximal term it is possible to avoid fast increase of the Lagrangian dual variables: in this way we also manage to keep the subproblems relatively easy in most cases. The method has been previously used in the SLR context to solve the p-median problem [BTV06].

This paper is organized as follows: In Section 2 we review the main properties of the SLR and discuss two algorithms to solve it: the dual ascent algorithm and Proximal-ACCPM. In Section 3, we apply the SLR to the UFL problem, develop some related theoretical properties and detail a specialization of the dual ascent algorithm for the UFL case. Section 4 reports our empirical study. Our conclusions are given in a last section.

2 Semi-Lagrangian relaxation

The concept of semi-Lagrangian relaxation was introduced in [BTV06]. We present it on a general model and summarize the main results obtained in that paper.

Consider the problem, to be named "primal" henceforth.

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

Assumption 1 *The components of $A \in \mathbb{R}^m \times \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ are nonnegative.*

Assumption 2 $0 \in X$ *and (1) is feasible.*

Assumptions 1 and 2 together imply that (1) has an optimal solution.

The semi-Lagrangian relaxation consists in adding the inequality constraint $Ax \leq b$ and relaxing $Ax = b$ only. We thus obtain the dual problem

$$\max_{u \in \mathbb{R}^m} \mathcal{L}(u), \tag{2}$$

where $\mathcal{L}(u)$ is the semi-Lagrangian dual defined by

$$\begin{aligned} \mathcal{L}(u) = \min_x \quad & c^T x + u^T(b - Ax) && \text{(3a)} \\ \text{s.t.} \quad & Ax \leq b, && \text{(3b)} \\ & x \in S. && \text{(3c)} \end{aligned}$$

Note that with our assumptions the feasible set of (3) is bounded. We also have that $x = 0$ is feasible to (3); hence (3) has an optimal solution. $\mathcal{L}(u)$ is well-defined, but the minimizer in (3) is not necessarily unique. With some abuse of notation, we write

$$x(u) = \arg\min_x \{c^T x + u^T(b - Ax) \mid Ax \leq b,\ x \in S\} \tag{4}$$

to denote one such minimizer. With this notation we may write $\mathcal{L}(u) = (c - A^T u)^T x(u) + b^T u$. Finally, we denote U^* the set of optimal solutions of problem (2).

We summarize in one theorem the main results obtained in [BTV06].

Theorem 1 *The following statements hold.*

1. *$\mathcal{L}(u)$ is concave and $b - Ax(u)$ is a supergradient at u.*
2. *$\mathcal{L}(u)$ is monotone and $\mathcal{L}(u') \geq \mathcal{L}(u)$ if $u' \geq u$, with strict inequality if $u' > u$ and $u' \notin U^*$.*
3. *$U^* = U^* \cup \mathbb{R}^m_+$; thus U^* is unbounded.*
4. *If $x(u)$ is such that $Ax(u) = b$, then $u \in U^*$ and $x(u)$ is optimal for problem (1).*
5. *Conversely, if $u \in$ int(U^*), then any minimizer $x(u)$ is optimal for problem (1).*
6. *The SLR closes the duality gap for problem (1).*

Since the proofs are short we replicate them below.

Proof: From the definition of the SLR function as a pointwise minimum, the inequality

$$\begin{aligned}\mathcal{L}(u') &\leq (c - A^T u)^T x(u) + b^T u' \\ &= \mathcal{L}(u) + (b - Ax(u))^T (u' - u), \end{aligned} \tag{5}$$

holds for any pair u, u'. This shows that $\mathcal{L}(u)$ is concave and that $(b - Ax(u))$ is a supergradient at u. This proves statement 1.

To prove statement 2, we note, in view of $b - Ax(u') \geq 0$ and $u' \geq u$, that we have the chain of inequalities

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

This proves the first part of the third statement. If $u' \notin U^*$, then $0 \notin \partial\mathcal{L}(u')$ and one has $(b - Ax(u'))_j > 0$ for some j. Thus, $u < u'$ implies $(b - Ax(u'))^T(u' - u) > 0$. Hence, $\mathcal{L}(u') > \mathcal{L}(u)$.

The third statement is an immediate consequence of the monotone property of $\mathcal{L}(u)$.

To prove the fouth statement, we note that $Ax(u) = b$ implies $0 \in \partial\mathcal{L}(u)$, a necessary and sufficient condition of optimality for problem (2). Hence $u \in U^*$. Finally, since $x(u)$ is feasible to (1) and optimal for the relaxation (3) of problem (1), it is also optimal for (1). Assume now $u \in$ int(U^*). In this case there exists $u' \in U^*$ such that $u' < u$; thus $(b - Ax(u))^T (u - u') \geq 0$, with strict inequality if $b - Ax(u) \neq 0$. In view of (5),

$$0 \geq (b - Ax(u))^T (u' - u) \geq \mathcal{L}(u') - \mathcal{L}(u).$$

Thus $Ax(u) = b$, and $x(u)$ is optimal to (1). It follows that the original problem (1) and the semi-Lagrangian dual problem (2) have the same optimal value (the last statement). ■

Theorem 1 tells us that if we can solve the SLR problem and exhibit some $u \in$ int(U^*), we can find a solution of problem (1) by taking $x = x(u)$. Assuming that we are able to solve the subproblem (4), also called *oracle*, it remains to solve the semi-Lagrangian dual problem itself. Since the SLR dual problem is concave, we can resort to one of the many methods for non-differentiable convex optimization. Any such method can be paraphrased as follows. The method queries the oracle (3) at successive points u^k. The oracle subproblem solution $x(u^k)$ at u^k provides the value of the semi-Lagrangian dual $(c - A^T u^k)^T x(u^k) + b^T u^k$ and one supergradient $b - Ax(u^k)$. In view of (5), one can build a piece-wise linear approximation of the semi-Lagrangian dual function. The approximation is increasingly refined and the sequence of query points eventually converges to U^*.

The overall complexity of the procedure depends on how many points are required to achieve convergence and on the computational effort in solving the oracle subproblem at each such point. We know that the original problem is NP-hard, so the SLR approach must also be NP-hard. Since there exist methods for non-differentiable convex optimization that converge in a polynomial number of iterations, we cannot expect that the oracle subproblem be polynomially solvable. From a theoretical point of view, we are hitting a dead end, but from a practical point of view, things often look differently.

The effort to solve the oracle subproblem is far from being the same at each iteration. This is particularly true in the application problems we shall consider in this paper. This assertion is based on the following observation. When solving the oracle at some u, we can assert that we have a solution $x(u)$ such that $x(u)_j = 0$ for all $(c - A^T u)_j \geq 0$. The dimension of the oracle subproblem can be reduced to the negative components $(c - A^T u)_j < 0$. If the sequence of query points is initiated at $u^0 = 0$, the optimal value of the subproblem oracle is trivially 0 and the supergradient is b. On the other hand, as u increases, the number of negative components in $c - A^T u$ increases as well. For very large u, $A \geq 0$ implies that all components in $c - A^T u$ are strictly negative. The dimension of the oracle problem is the same as the original problem; most likely, it is as difficult. Between these extreme cases, the oracle subproblem may hopefully remain easy enough to be tackled successfully by an efficient MIP solver.

The above discussion shows that one cannot rely on obvious principles to design an efficient solution method for solving the original problem via semi-Lagrangian relaxation. The solution method should compromise between contradictory objectives: keeping the iterates u^k small to have a maximum of non-negative terms $c - A^T u$ and increasing u^k fast enough to reach U^*. In this paper we propose two different methods. The first one consists in choosing a theoretically and practically efficient general method for solving the semi-Lagrangian dual problem. As for the p-median problem [BTV06] we select the proximal analytic center cutting plane method (proximal-ACCPM). This ensures a small number of iterations, and seems to keep the oracle subproblem simple enough during the solution process. The other method is a variant of the dual ascent method (e.g., [Belar]) based on finite increases of the components of u. In fact, the dual ascent algorithm we use is in essence the dual multi-ascent procedure used in [Koe89] to solve the Erlenkotter's 'condensed' dual of the UFL problem [Erl78]. In the case of LR, the dual multi-ascent method does not necessarily converge, but we prove finite convergence in the SLR case.

To close this Section we present the two methods we use to solve the SLR dual problem (2).

2.1 The dual ascent method

We state the algorithm first and then prove finite convergence.

Algorithm 1: Dual ascent algorithm - basic iteration.

```
1. The current iterate is $u^k$.  $\Delta > 0$ is a fixed number.

2. Compute $x^k = \arg\min_x \{c^T x + (u^k)^T (b - Ax) \mid Ax \le b,\ x \in S\}$.

3. If $s^k := b - Ax^k = 0$, stop.
   $x^k$ is optimal for the original problem.

4. Otherwise, let $u^{k+1}$ be defined by
   $u_j^{k+1} = \begin{cases} u_j^k + \delta_j^k & \text{if } s_j^k > 0, \\ u_j^k & \text{otherwise}, \end{cases}$

   where $\delta_j^k \ge \Delta$.
```

We now prove finite convergence.

Theorem 2 *The following statements hold.*

1. *Algorithm 1 is a dual ascent method.*
2. *Let us suppose that $u^0 \ge 0$ and that $U^* \neq \emptyset$. Algorithm 1 converges to an optimal point $u \in U^*$ after finitely many iterations.*

Proof: Let us prove statement 1 first. The updating procedure of Algorithm 1 consists in increasing some components of the current dual point u^k (step 4). Thus, $u^{k+1} \ge u^k$ and by Theorem 1.2 we have that $\mathcal{L}(u^{k+1}) \ge \mathcal{L}(u^k)$.

The proof of statement 2 goes as follows. Let us consider the sequence $\{s^k\}$ of subgradients generated by the algorithm. We have two exclusive cases.

Case 1: There exists k_0 such that $s^{k_0} = 0$. Then $0 \in \partial\mathcal{L}(u^{k_0})$ and $u^{k_0} \in U^*$.

Case 2: At least for one component of s^k, say the 1st, there exists a subsequence $\{s_1^{k_i}\} \subset \{s_1^k\}$ such that $s_1^{k_i} \neq 0$ for all $i = 0, 1, 2, \ldots$. We will prove by contradiction that this case cannot happen.

By definition of the algorithm we will have:

$$u_1^{k_i} \ge u_1^{k_0} + i\Delta. \tag{6}$$

Let us show that the subsequence $\{\mathcal{L}(u^{k_i})\}$ is unbounded from above which contradicts the hypothesis $U^* \neq \emptyset$. Let us define $J^{k_i} = \{j \mid s_j^{k_i} > 0\}$. Since x is a binary vector, it implies, by Assumption 1, that there exists an absolute constant $\eta > 0$ such that

$$\min_j \min_x \{s_j = (b - Ax)_j \mid (b - Ax)_j > 0\} = \eta.$$

Thus $s_j^{k_i} \ge \eta$ for all $j \in J^{k_i}$ and for all i.

Using the fact that $c^T x \ge 0$ and that $u^{k_i} \ge 0$, we have

$$\begin{aligned} \mathcal{L}(u^{k_i}) &= c^T x(u^{k_i}) + (b - Ax(u^{k_i}))^T u^{k_i} \\ &= c^T x(u^{k_i}) + {(s^{k_i})}^T u^{k_i} \end{aligned}$$

$$
\begin{aligned}
&\geq (s^{k_i})^T u^{k_i} \\
&= \sum_{j \in J^{k_i}} s_j^{k_i} u_j^{k_i} \\
&\geq u_1^{k_i} \eta \\
&\geq (u_1^{k_0} + i\Delta)\eta.
\end{aligned}
$$

Thus $\lim_{i \to \infty} \mathcal{L}(u^{k_i}) = +\infty$. ■

2.2 The proximal analytic center cutting plane method

Function $\mathcal{L}(u)$ in problem (10) is by construction concave and nonsmooth, since it is implicitly defined as the pointwise minimum of linear functions in u. Extensive numerical experience shows that Proximal-ACCPM, is an efficient tool for solving (10). See, for instance, [GV02] and references therein included; see also [BTV06] for experiments with the p-median problem.

In the cutting plane procedure, we consider a sequence of points $\{u^k\}_{k \in K}$ in the domain of $\mathcal{L}(u)$. 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.

As already said, 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

$$L_K = \{(u, z) \in \mathbb{R}^{n+1} \mid u \in \mathbb{R}^n, \quad z \leq \mathcal{L}^k(u) \quad \forall k \leq K, \quad z \geq \theta_l\}, \tag{7}$$

where K is the current iteration number. The basic iteration of a cutting plane method can be summarized as follows:

1. Select $(\overline{u}, \overline{z})$ in the localization set L_K.
2. Call the oracle at $\overline{u}$. The oracle returns one or several optimality cuts and a new lower bound $\mathcal{L}(\overline{u})$.
3. Update the bounds:
   (a) $\theta_l \leftarrow \max\{\mathcal{L}(\overline{u}), \theta_l\}$.
   (b) Compute an upper bound θ_u to the optimum[1] of problem (10).

[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.

4. `Update the lower bound` θ_l `and add the new cuts in the definition of the localization set (7).`

These steps are repeated until a point is found such that $\theta_u - \theta_l$ falls below a prescribed optimality tolerance. The first step in the above algorithm sketch 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 $(\overline{u}, \overline{z})$ that maximizes z in the localization set [Kel60] may prove disastrous, because it over-emphasizes the global approximation property of the localization set. Safer methods, as for example bundle methods [HUL96] or Proximal-ACCPM [GHV92, GV02, dMV02], 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 $(\overline{u}, \overline{z})$ 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.

3 SLR applied to the UFL problem

In the Uncapacitated Facility Location (UFL) problem we have a set of 'facilities' indexed by $I = \{1, \ldots, m\}$ and a set of 'customers' indexed by $J = \{1, \ldots, n\}$. The purpose is to open facilities relative to the set of 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 plus the fix costs of opened facilities f_i. The distance is sometimes weighed by an appropriate factor, e.g., the demand at a customer node. The objective is to minimize this sum. The UFL problem is NP-hard [MF90] and can be formulated as follows.

$$z^* = \min_{x,y} \quad z(x,y) \tag{8a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = 1, \quad j \in J, \tag{8b}$$

$$x_{ij} \leq y_i, \quad i \in I, j \in J, \tag{8c}$$

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

where

$$z(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i. \tag{9}$$

[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

$$(\overline{u}, \overline{z}) = \text{argmin}_{u,z} F_L(u,z) + \rho \|u - \hat{u}\|^2,$$

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).

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

Following the ideas of the preceding section, we formulate the semi-Lagrangian relaxation of the UFL problem. We obtain the dual problem

$$\max_{u \in \mathbb{R}^n} \mathcal{L}(u) \tag{10}$$

and the oracle

$$\begin{aligned} \mathcal{L}(u) = \quad \min_{x,y} \quad & f(u,x,y) && \text{(11a)} \\ \text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} \leq 1, \,, \quad j \in J, && \text{(11b)} \\ & x_{ij} \leq y_i, \quad i \in I, j \in J, && \text{(11c)} \\ & x_{ij}, y_i \in \{0,1\}, && \text{(11d)} \end{aligned}$$

where

$$\begin{aligned} f(u,x,y) \quad &= \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i + \sum_{j=1}^{n} u_j(1 - \sum_{i=1}^{m} x_{ij}) \\ &= \quad \sum_{i=1}^{m}\sum_{j=1}^{n} (c_{ij} - u_j)x_{ij} + \sum_{i=1}^{m} f_i y_i + \sum_{j=1}^{n} u_j. \end{aligned} \tag{12}$$

As in the previous section, we denote $(x(u), y(u))$ an optimal pair for (11).

3.1 Properties of the SLR dual problem

In this section we show that, in order to solve the SLR dual problem, we can restrict our dual search to a box. To this end, we define the *best combined costs* as $\tilde{c}_j := \min_i\{c_{ij} + f_i\}$ for all $j \in J$. The vector of best combined costs is thus $\tilde{c} := (\tilde{c}_1, \dots, \tilde{c}_n)$.

Theorem 3 $u \geq \tilde{c} \Rightarrow u \in U^*$ *and* $u > \tilde{c} \Rightarrow u \in$ *int*(U^*).

Proof: Consider the oracle

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{m}(f_i y_i + \sum_{j=1}^{n}(c_{ij} - u_j)x_{ij}) + \sum_{j=1}^{n} u_j \\ & \sum_{i=1}^{m} x_{ij} \leq 1, \ \forall j \\ & x_{ij} \leq y_i, \ \forall i,j \\ & x_{ij}, y_i \in \{0,1\}. \end{aligned}$$

Assume $u \geq \tilde{c}$. If there exists an optimal solution of the oracle such that $\sum_i x_{ij} = 1, \ \forall j$, then, by Theorem 1, this solution is optimal for the original problem. Assume we have an oracle solution with $\sum_i x_{ij} = 0$ for some j. Let i_k be such that $\tilde{c}_j = f_{i_k} + c_{i_k j}$. By hypothesis,

$\tilde{c}_j - u_j \leq 0$. Thus, $f_{i_k} + (c_{i_k j} - u_j) \leq 0$ and one can set $x_{i_k j} = 1$ and $y_{i_k} = 1$ without increasing the objective value. The modified solution is also optimal. Hence, there exists an optimal oracle solution with $\sum_i x_{ij} = 1,\ \forall j$ and $u \in U^*$. The second statement of the theorem follows from $\tilde{c} \in U^*$ and statement 3 of Theorem 1. ■

Theorem 4 *If $u \in$ int(U^*), then $u \geq c^1$.*

Proof: Let us assume assume that $u_{j_0} < c^1_{j_0}$ for some $j_0 \in J$ and see that this contradicts $u \in$ int(U^*). If $u_{j_0} < c^1_{j_0}$ then $c^k_{j_0} - u_{j_0} > 0$ for all $k \in I$. Any optimal solution $x(u)$ is such that $x_{ij_0}(u) = 0$, for all $i \in I$. Hence, $1 - \sum_{i \in I} x_{ij_0}(u) = 1$ and by Theorem 1, u is not in int(U^*). ■

Corollary 1 *For any $\epsilon > 0$, one has*

$$\textit{int}(U^*) \cap \{u \mid c^1 < u \leq \tilde{c} + \bar{\epsilon}\} \neq \emptyset,$$

where $\bar{\epsilon}$ is a vector where each component is equal to ϵ.

The corollary shows that the search for an optimal dual solution can be confined to a box. In particular, taking $u = \tilde{c} + \bar{\epsilon}$ and solving the oracle for u yields a primal optimal solution in one step. As pointed out earlier, it is likely to be impractical because the oracle is too difficult at that u. Surely enough, that point is optimal for the SLR dual problem, but almost surely there is a smaller $u \in U^*$ for which the oracle subproblem would be easier (with less binary variables) and hopefully tractable by an integer programming solver. This justifies the use of an iterative search that increases the components of u in a very controlled way.

3.2 Structure of the oracle: the core subproblem

In this section, we study the structure of the UFL oracle as a function of the dual variable u. We shall show how some primal variables x_{ij} can be set to zero when u_j is large enough. This operation, which reduces the size of the oracle, is quite common in Lagrangian relaxation applied to combinatorial optimization. There, using some appropriate argument, on fixes some of the oracle variables and one obtains a reduced-size oracle called the *core subproblem*. (See, for instance, [BN04, ASV07].) We now define the core subproblem in the SLR case.

Let $c(u)$ be the matrix of reduced costs such that $c(u)_{ij} = c_{ij} - u_j$. Let $G = (V \times W, E)$ be the bipartite graph associated to the UFL problem, such that each node in V (W) represents a facility (customer) and the edge e_{ij} exists if facility i can serve customer j. $c(u)_{ij}$ is thus the reduced cost associated to e_{ij}. Let $E(u) \subset E$ be the subset of edges with strictly negative reduced cost for a given dual vector u. Let $V(u) \subset V$ and $W(u) \subset W$ be the adjacent vertices to $E(u)$. Let $G(u) = (V(u) \times W(u), E(u))$ be the induced bipartite subgraph. We call $G(u)$ the *core graph*.

As already mentioned, for any $c(u)_{ij} \geq 0$ there exists $x(u)$ such that $x(u)_{ij} = 0$. Therefore, we can restrict our search to the core graph $G(u)$ to compute $x(u)$. Solving oracle (11) with core graph $G(u)$ is equivalent to using the whole graph G. However, the first option is easier in general: Usually in $G(u)$ we have (much) less edges (variables x_{ij}) and less vertices (variables y_j), as we will see in Section 4. A further advantage of solving the core subproblem is that, even though G is usually a connected graph, $G(u)$ may be decomposable into independent subgraphs and then we can break the core subproblem into smaller (easier) subproblems.

We now study the inverse image of the mapping $G(u)$. In other words, we want to describe the set of u's that have the graph G as a common image through the mapping $G(u)$. It is easy to see that it is a simple box. To be more precise, we introduce a new definition. For each customer j, we sort its costs c_{ij}, and get the *sorted costs*

$$c_j^1 \le c_j^2 \le \ldots \le c_j^m.$$

To define an elementary box, we use the sorted costs to partition the domain of each coordinate u_j into intervals of the form $c_j^k < u_j \le c_j^{k+1}$, with the understanding that $c_j^{m+1} = +\infty$. Note that some boxes may be empty (if $c_j^k = c_j^{k+1}$). These partitions induce a partition of the space of $u > c^1$ into elementary boxes having each a unique graph image. The dual ascent algorithm to be presented in the next section will exploit this structure.

3.3 The dual ascent algorithm to solve the SLR dual problem

We now present a specialized version of the dual ascent method (Algorithm 1) for the UFL problem. We know by Corollary 1, that the search can be confined to the box $\{u \mid c^1 < u \le \tilde{c} + \bar{\epsilon}\}$. The discussion of the previous section further shows that it is enough to restrict the search to one representative point per elementary box. Our choice is to take that point as small as possible within the box. Since an elementary box is made of semi intervals open from the left, we choose the representative point to be $u_j = c_j^{l(j)} + \epsilon$, for some fixed $\epsilon > 0$.

Our algorithmic scheme is as follows. For some $l \in I$ and some small $\epsilon > 0$, start the dual search at $u_j^0 := c_j^l + \epsilon$ (each customer node will have l core edges). Query the oracle, and if the current dual iterate is not optimal, update it. To maintain the number of core edges as low as possible, at each iteration, we add at most one edge per customer, that is, we update u_j from $c_j^{l'} + \epsilon$ to $c_j^{l'+1} + \epsilon$, for some $l' \in I$. We only update the components of u whose corresponding subgradient component is not null. The details of this simple procedure are as follows.

Algorithm 2: Dual ascent algorithm - UFL case.

```
1. Initialization: Set $k = 0$ and $\epsilon > 0$. For each customer $j \in J$,
   set
   (a) $u_j^0 = c_j^{l(j)} + \epsilon$ for some $l(j) \in I$,
   (b) $\tilde{c}_j = \min_i\{c_{ij} + f_i\}$,
   (c) $c_j^{n+1} = +\infty$.
2. Oracle call: Compute $\mathcal{L}(u^k)$, $(x(u^k), y(u^k))$ and $s^k$ where
       $$s_j^k = 1 - \sum_{i=1}^{m} x_{ij}^k,$$
   for all $j \in J$. (Note that $s_j^k \in \{0, 1\}$.)
```

```
3. *Stopping criterion*: If $s^k = 0$ then stop. The pair $(u^k; (x(u^k), y(u^k))$
   is a primal-dual optimal solution.
4. *Multiplier updating*: For each $j \in J$ such that $s_j^k = 1$, set
   $u_j^{k+1} = \min\{c_j^{l(j)+1}, \tilde{c}_j\} + \epsilon$ and $l(j) = \min\{l(j) + 1, n\}$.
5. Set $k = k + 1$ and go to Step 2.
```

By construction, the iterates are monotonic with at least one strictly increasing coordinate. The algorithm converges in a finite number of iterations. The semi-Lagrangian dual function is also monotonic, which makes the algorithm a dual-acent.

4 Empirical study

The objective of our numerical test is twofold: first, we study the quality of the solution given by the semi-Lagrangian relaxation (SLR) and second we study the SLR performance in terms of CPU time, either when one uses Proximal-ACCPM or the dual ascent algorithm as dual optimizer. We compare our results to the current state-of-the-art results reported in [RW06] and in the *Uncapacitated Falicity Location Library* (UflLib) [Hoe06].

4.1 Parameter settings for the algorithms

Programs have been written in MATLAB 7.0 and run in a PC (Pentium-IV, 3.0 GHz, with 3 GB of RAM memory) under the Windows XP operating system. On the other hand, the reader should bear in mind that results reported in [RW06] were found on a different machine (SGI Challenge with 28 196-MHz MIPS R10000 processors, but each execution was limited to a single processor) and programs were implemented in C++.

In our context and using the terminology of oracle based optimization [BBH$^+$06], to call or to solve an *oracle* means to compute $\mathcal{L}(u^k)$: Oracle 1 for the Lagrangian relaxation and Oracle 2 for the semi-Lagrangian relaxation. To solve Oracle 2 (a large-scale mixed integer program) we have intensively used CPLEX 9.1 (default settings) interfaced with MATLAB [Bao04].

The dual ascent algorithm is implemented as stated in the previous section. The parameter ϵ is set to 10^{-3}. Proximal-ACCPM is used in its generic form as described in [BBH$^+$06]. With the SLR oracle, we choose not to update the the reference proximal point to the best point, but to keep it fixed at the initial point u^0. As already pointed out in [BTV06], a point slightly larger than the optimal Lagrangian multipliers is a very good starting point for the SLR. In the UFL case, it turns out that, in many cases, this point is already in int(U^*), so that the algorithm terminates in one iteration (columns (5.b-c)). Besides, this point can be computed in short CPU time (column (5.d)) with Proximal-ACCPM. To be complete, we mention that the parameters in Proximal-ACCPM are as follows. In the search for the initial point (the LR phase), we use a proximal weight ρ that at each Proximal-ACCPM iteration is updated within the range $[10^{-6}, 10^4]$ and the required precision is 10^{-6}. In the SLR phase, the proximal weight ρ is fixed to 10^{-4} for the K-median instances and to 10^{-3} for the Koerkel-Ghosh instances.

Finally, at each iteration of the SLR phase, we use the following heuristic to compute a primal feasible solution: the oracle solution proposes a set of open facilities and a subset of assigned

Table 1: Instance description: For the K-median instances, the fix cost is the same for all facilities. For the Koerkel-Ghosh instances the fix cost is randomly chosen for each facility according to a uniform distribution.

K-median Instances	Nb. of customers	Fix cost	Koerkel-Ghosh Instances	Nb. of customers	Fix cost
500-1000	500	224	gs00250a-1	250	$\mathcal{U}$[100,200]
500-100	500	2236	gs00250b-1	250	$\mathcal{U}$[1000,2000]
500-10	500	22361	gs00250c-1	250	$\mathcal{U}$[10000,20000]
1000-1000	1000	316	gs00500a-1	500	$\mathcal{U}$[100,200]
1000-100	1000	3162	gs00500b-1	500	$\mathcal{U}$[1000,2000]
1000-10	1000	31623	gs00500c-1	500	$\mathcal{U}$[10000,20000]
1500-1000	1500	387	gs00750a-1	750	$\mathcal{U}$[100,200]
1500-100	1500	3873	gs00750b-1	750	$\mathcal{U}$[1000,2000]
1500-10	1500	38730	gs00750c-1	750	$\mathcal{U}$[10000,20000]
2000-1000	2000	447	ga00250a-1	250	$\mathcal{U}$[100,200]
2000-100	2000	4472	ga00250b-1	250	$\mathcal{U}$[1000,2000]
2000-10	2000	44721	ga00250c-1	250	$\mathcal{U}$[10000,20000]
2500-1000	2500	500	ga00500a-1	500	$\mathcal{U}$[100,200]
2500-100	2500	5000	ga00500b-1	500	$\mathcal{U}$[1000,2000]
2500-10	2500	50000	ga00500c-1	500	$\mathcal{U}$[10000,20000]
3000-1000	3000	548	ga00750a-1	750	$\mathcal{U}$[100,200]
3000-100	3000	5477	ga00750b-1	750	$\mathcal{U}$[1000,2000]
3000-10	3000	54772	ga00750c-1	750	$\mathcal{U}$[10000,20000]

customers; all unassigned customers are then assigned to their closest facility to form a feasible solution. If the SLR algorithm does not converge (within the CPU time limit) to an optimal primal solution, we take as primal solution the best obtained among the previous iterations.

4.2 Instance description

For our test we use 36 *unsolved* UFL instances that come from two public sets of instances from UflLib (see Table 1). The first set is called K-median. These are 18 large scale instances, which were translated from benchmarks for the K-median problem [ACCF88] to be used as UFL instances in [BC99]. In the Euclidian plane n points are generated. Each point simultaneously represents a facility and a customer $(m = n)$, with $n = 500, 1000, ..., 3000$. The connection costs c_{ij} are determined by the Euclidian distance. In each instance all the fix costs f_i are equal and calculated by $\sqrt{n}/l$ with $l = 10, 100$ or 1000. All values are rounded up to 4 singnificant digits and made integer [Hoe06]. We use the label $n - l$ to name these instances.

The second set is called Koerkel-Ghosh. A benchmark of this type was initially proposed by [Koe89]. Here, connection costs are drawn uniformly at random from $[1000, 2000]$. Fixed costs are drawn uniformly at random from [100, 200] in class ‘a’, from $[1000, 2000]$ in class ‘b’ and from $[10000, 20000]$ in class ‘c’. Furthermore symmetric and asymmetric connection matrices are created. UflLib provides instances of the 3 largest sizes presented in [Gho03] with $n = m = 250, 500$ and 750. Of the 90 instances proposed in the UflLib, we took 18 representative ones. We use the label gX00YZ-1 to name these instances, where X can be either ‘s’ or ‘a’ for the symmetric or asymmetric case respectively. Y is equal to ‘n’ and Z is the class (a, b or c).

4.3 Dual solution quality

In this section we study the dual solution quality and the duality gap (see Table 2). Column (2.a) displays the best upper bound (UB) to the optimal primal cost as displayed in columns (3.b), (3.d) and (3.f). (In some cases the UB is the best one since it is the primal optimum.)

In columns (2.b) and (2.c) we have the Lagrangian relaxation lower bound (LB): (2.b) reported in [Hoe03] and (2.c) obtained by Proximal-ACCPM. For the K-median instances, we give the LB reported in [Hoe03] as a fraction of our LB (2.c). We observe that Proximal-ACCPM produces slightly more accurate bounds. For the Koerkel-Ghosh instances we only report our LB's, since no other results have been reported in literature.

In column (2.d) we have an upper bound to the duality gap computed from columns (2.a) and (2.c). For the K-median instances the average gap bound is 0.0184% and for the Koerkel-Ghosh instances is 1.3980%. This partially explains that Koerkel-Ghosh instances are more difficult to solve than the K-median ones. In columns (2.e) and (2.f) we have the semi-Lagrangian relaxation LB: (2.e) obtained by the dual ascent algorithm and (2.f) by Proximal-ACCPM. As we can see in the 'Global average' line, column (2.g), Proximal-ACCPM produces, on average, slightly more accurate bounds in less time (64% of the dual ascent CPU time as reported in Table 5.e-f). Furthermore, as expected the semi-Lagrangian relaxation produces stronger LB's than the Lagrangian relaxation (columns (2.c) and (2.f)).

4.4 Primal solution quality

In this section we study the primal solution quality (see Table 3). Column (3.a) displays the SLR LB bound computed by Proximal-ACCPM. In column (3.b), K-median instances, we have the UB reported in [RW06] and obtained as the average of several runs (average values). In column (3.b), Koerkel-Ghosh instances, we have the UB reported in UflLib (the results reported in [RW06] for these instances do not correspond to sigle instances but for groups of 5 instances). In columns (3.d) and (3.f) we have the UB obtained by the dual ascent method and Proximal-ACCPM, respectively. We measure the quality of each UB (solution) in terms of the following optimality index: $100 \times [1 - (UB - LB)/LB]\%$.

From our set of 36 *unsolved* UFL instances, we have that the heuristic procedure in [RW06], dual ascent and Proximal-ACCPM solve up to optimality, respectively, 3, 16 and 18 instances. Recall that with the heuristic [RW06] one cannot check for optimality (even if the computed solution is optimal). The other instances remain unsolved and in general the sophisticated heuristic used in [RW06] performs much better than our naive primal heuristic. Finally, note that although the Koerkel-Ghosh instances are smaller, they result much harder to solve than the K-median ones.

4.5 Instance reduction

Very often Lagrangian relaxation is used to decompose difficult problems. The decomposition induced by SLR is more coarse-grained than the Lagrangian relaxation one. Thus for example, in the UFL problem, after Lagrangian relaxation, one has one subproblem per facility. However,

Table 2: Dual solution quality: ‘UB’ stands for upper bound, ‘LB’ for lower bound, ‘PACCPM’ for Proximal-ACCPM, ‘duality gap bound’ is computed comparing the UB and the LB given by the Lagrangian relaxation (column (2.b)), ‘DA’ for dual ascent. For the K-median instances, we give the LB reported in [Hoe03] as a fraction of our LB (2.c). For the Koerkel-Ghosh instances, no LB has ever been reported before.

Instance	UB	LB Lagrangian relaxation		% Duality gap bound	LB semi-Lag. relaxation		
		[Hoe03]	PACCPM		DA	PACCPM	(2.e) / (2.f)
	(2.a)	(2.b)	(2.c)	(2.d)	(2.e)	(2.f)	(2.g)
500-1000	99169	99.94 %	99160	0.0091	99169	99169	100 %
500-100	326790	99.99 %	326790	0	326790	326790	100 %
500-10	798577	99.98 %	798577	0	798577	798577	100 %
1000-1000	220560	99.96 %	220559	0.0003	220560	220560	100 %
1000-100	607878	99.96 %	607814	0.0105	607878	607878	100 %
1000-10	1434154	99.95 %	1433389	0.0534	1434154	1434154	100 %
1500-1000	334962	99.97 %	334944	0.0054	334962	334962	100 %
1500-100	866454	99.97 %	866453	0.0001	866454	866454	100 %
1500-10	2000801	99.90 %	1999284	0.0759	2000696	2000801	99.99 %
2000-1000	437686	99.97 %	437678	0.0018	437686	437686	100 %
2000-100	1122748	99.97 %	1122746	0.0002	1122748	1122748	100 %
2000-10	2558118	99.96 %	2557753	0.0143	2558118	2558118	100 %
2500-1000	534405	99.95 %	534395	0.0019	534405	534405	100 %
2500-100	1347516	99.96 %	1347494	0.0016	1347516	1347516	100 %
2500-10	3100225	99.94 %	3096856	0.1088	3097647	3097647	100 %
3000-1000	643463	99.97 %	643432	0.0048	643463	643463	100 %
3000-100	1602335	99.93 %	1601652	0.0426	1602063	1602120	100 %
3000-10	3570766	99.90 %	3570752	0.0004	3570766	3570766	100 %
Partial average	1200367	99.94 %	1199985	0.0184	1200203	1200212	100 %
gs00250a-1	257964	-	257639	0.1261	257899	257964	99.97 %
gs00250b-1	276761	-	273693	1.1209	275363	275574	99.92 %
gs00250c-1	332935	-	322696	3.1729	329135	330559	99.57 %
gs00500a-1	511229	-	510408	0.1608	510408	510408	100 %
gs00500b-1	537931	-	533020	0.9214	534029	533477	100.10 %
gs00500c-1	620041	-	601962	3.0034	607260	609333	99.66 %
gs00750a-1	763671	-	762562	0.1455	762562	762562	100 %
gs00750b-1	797026	-	790334	0.8467	790917	790917	100 %
gs00750c-1	900454	-	875340	2.8690	879430	879343	100.01 %
ga00250a-1	257957	-	257618	0.1317	257882	257957	99.97 %
ga00250b-1	276339	-	273296	1.1135	275080	275200	99.96 %
ga00250c-1	334135	-	322958	3.4609	329839	331171	99.60 %
ga00500a-1	511422	-	510587	0.1635	510587	510587	100 %
ga00500b-1	538060	-	533334	0.8862	534416	533837	100.11 %
ga00500c-1	621360	-	602852	3.0701	608518	609975	99.76 %
ga00750a-1	763576	-	762462	0.1461	762462	762462	100 %
ga00750b-1	796480	-	790112	0.8059	790631	790630	100 %
ga00750c-1	902026	-	875593	3.0189	880301	879601	100.08 %
Partial average	555520	-	547581	1.3980	549818	550087	99.93 %
Global average	877944	-	873783	0.7082	875010	875149	99.96 %

Table 3: Primal solution quality: 'SLR' stands for semi-Lagrangian relaxation, 'PACCPM' for Proximal-ACCPM, 'Opt.' for optimality, 'LB' for lower bound, 'UB' for upper bound and 'DA' for dual ascent.

Instance	SLR PACCPM	[RW06]		DA		PACCPM	
	LB	UB	% Opt.	UB	% Opt.	UB	% Opt.
	(3.a)	(3.b)	(3.c)	(3.d)	(3.e)	(3.f)	(3.g)
500-1000	99169	99169.0	100	99169	100	99169	100
500-100	326790	326805.4	99.9953	326790	100	326790	100
500-10	798577	798577.0	100	798577	100	798577	100
1000-1000	220560	220560.9	99.9996	220560	100	220560	100
1000-100	607878	607880.4	99.9996	607878	100	607878	100
1000-10	1434154	1434185.4	99.9978	1434154	100	1434154	100
1500-1000	334962	334973.2	99.9967	334962	100	334962	100
1500-100	866454	866493.2	99.9955	866454	100	866454	100
1500-10	2000801	2001121.7	99.9840	2000801	100	2000801	100
2000-1000	437686	437690.7	99.9989	437686	100	437686	100
2000-100	1122748	1122861.9	99.9899	1122748	100	1122748	100
2000-10	2558118	2558120.8	99.9999	2558118	100	2558118	100
2500-1000	534405	534426.6	99.9960	534405	100	534405	100
2500-100	1347516	1347577.6	99.9954	1347516	100	1347516	100
2500-10	3097647	3100224.7	99.9168	3122045	99.2124	3122045	99.2124
3000-1000	643463	643541.8	99.9878	643463	100	643463	100
3000-100	1602120	1602530.9	99.9744	1602335	99.9866	1602397	99.9827
3000-10	3570766	3570818.8	99.9985	3570766	100	3570766	100
Partial average	1200212	1200420.0	99.9903	1201579	99.9555	1201583	99.9553
gs00250a-1	257964	257964	100	258137	99.9329	257964	100
gs00250b-1	275574	276761	99.5693	279416	98.6058	278337	98.9974
gs00250c-1	330559	332935	99.2812	337270	97.9698	333617	99.0749
gs00500a-1	510408	511229	99.8392	513038	99.4847	513038	99.4847
gs00500b-1	533477	537931	99.1651	551716	96.5811	551716	96.5811
gs00500c-1	609333	620041	98.2427	641659	94.6949	641659	94.6949
gs00750a-1	762562	763671	99.8545	767269	99.3827	767269	99.3827
gs00750b-1	790917	797026	99.2275	810239	97.5569	810239	97.5569
gs00750c-1	879343	900454	97.5992	971616	89.5066	971616	89.5066
ga00250a-1	257957	257969	99.9953	258363	99.8426	257957	100
ga00250b-1	275200	276339	99.5861	280695	98.0033	278442	98.8219
ga00250c-1	331171	334135	99.1050	346296	95.4329	337398	98.1197
ga00500a-1	510587	511422	99.8365	513673	99.3956	513673	99.3956
ga00500b-1	533837	538060	99.2090	546253	97.6743	546253	97.6743
ga00500c-1	609975	621360	98.1335	670597	90.0616	643140	94.5629
ga00750a-1	762462	763576	99.8539	767965	99.2783	767965	99.2783
ga00750b-1	790630	796480	99.2601	808105	97.7898	808105	97.7898
ga00750c-1	879601	902026	97.4505	1000972	86.2016	1008497	85.3461
Partial average	550087	555521	99.1783	573516	96.5220	571494	97.0149
Global average	875149	877971	99.5843	887547	98.2387	886538	98.4851

after SLR, one may have from several to only one subproblem. In Table 4, columns (4.a) and (4.b) we have the average number of subproblems per Oracle 2 call, for the dual ascent method and for Proximal-ACCPM respectively. In 30 cases the SLR does not decompose the instance (ANSO2 = 1). The remaining 6 instances are decomposed by the SLR and this decomposition is slightly better with the dual ascent algorithm than with Proximal-ACCPM.

An important advantage of the SLR is that usually it drastically reduces the number of relevant variables (otherwise said, we can fix many variables). In columns (4.c) and (4.f) we display the total number of variables (x_{ij} and y_i, respectively). On average, in this test the SLR only uses 2.5% of the x_{ij} variables and 46% of the y_i variables when we use the dual ascent algorithm (columns (4.d) and (4.g)). In the Proximal-ACCPM case we use a slightly higher number of variables (columns (4.e) and (4.h)).

4.6 Performance

Finally, in Table 5 we display the performance of the dual ascent method and Proximal-ACCPM in terms of number of SLR iterations and CPU time. Comparing columns (5.b) and (5.c) we observe that, on average, Proximal-ACCPM reduces more than 60% the dual ascent number of iterations. For the K-median instances, one remarkable fact is that a few SLR iterations (in some cases only one) are enough to compute an optimal solution. The exception are instances 2500-10 and 3000-100 where SLR stops before convergence because an excess of CPU time. The CPU time limit is set as follows: we stop the SLR algorithm after the first completed SLR iteration that produces a cumulated CPU time above 7200 seconds. If that iteration, say the kth one, goes beyond 10000 seconds, we stop the SLR procedure and report the results of the $(k-1)$th iteration. For the Koerkel-Ghosh instances tings are different: when the SLR algorithm stops after only 1 or 0 completed iterations, it is because an excess of CPU time.

As already pointed out in [BTV06] the optimal set of Lagrangian multipliers is a very good starting point for the SLR: because it is computationally cheap in terms of CPU time (column (5.d)) and because from that starting point a few SLR iterations are enough to reach optimality (1 iteration in many cases). See columns (5.b) and (5.c).

In columns (5.e) to (5.g) we have the total CPU time. The first observation is that, on average, Proximal-ACCPM reduces approximately a 35% the dual ascent CPU time. This is mainly due to the effective use that Proximal-ACCPM makes of the available information: cutting planes and the box defined in Section 3. The main advantage of the dual ascent algorithm is its extreme simplicity.

Proximal-ACCPM reduces a 60% the average number of dual ascent iterations. One would expect a similar reduction in the CPU time, instead of the mentioned 35%. The reason for this mismatch, as we can see in (5.e) and (5.f), is that the average CPU time per Oracle 2 call is greater for Proximal-ACCPM than for the dual ascent algorithm. This is because Proximal-ACCPM perform a faster increase of the Lagrange multiplier values, which produces a faster increase in the number of relevant variables, as observed in Table 4. This produces harder (CPU time consuming) Oracle 2 calls. (See Fig. 1 and Fig. 2.)

As usual, a heuristic method reduces the CPU time of an exact method at the price of no information about the solution quality. If we compare the best SLR results (column (5.f)) versus the results reported in [RW06] (column (5.g)) we observe that: for the K-median instances, the

Table 4: Semi-Lagranian reduction and decomposition: ‘ANSO2’ stands for average number of subproblems per Oracle 2 call, ‘DA’ for dual ascent, ‘PACCPM’ for proximal-ACCPM, ‘ANRXVO2’ for average number of relevant x_{ij} variables per Oracle 2 call and ‘ANRYVO2’ for average number of relevant y_i variables per Oracle 2 call.

Instance	ANSO2		Nb. of x_{ij}	ANRXVO2 (%)		Nb. of y_i	ANRYVO2 (%)	
	DA	PACCPM	variables	DA	PACCPM	variables	DA	PACCPM
	(4.a)	(4.b)	(4.c)	(4.d)	(4.e)	(4.f)	(4.g)	(4.h)
500-1000	108	84	250000	0.3	0.5	500	56	98
500-100	1	1	250000	1.2	1.2	500	66	66
500-10	1	1	250000	2.8	2.8	500	35	35
1000-1000	106	29	1000000	0.3	0.3	1000	89	94
1000-100	1	1	1000000	0.8	0.9	1000	55	61
1000-10	1	1	1000000	2.6	3.4	1000	39	51
1500-1000	11	7	2250000	0.3	0.3	1500	86	87
1500-100	1	1	2250000	0.6	0.6	1500	50	50
1500-10	1	1	2250000	1.8	2.6	1500	30	44
2000-1000	2	2	4000000	0.2	0.2	2000	82	82
2000-100	1	1	4000000	0.6	0.6	2000	50	50
2000-10	1	1	4000000	1.2	2.2	2000	25	42
2500-1000	3	3	6250000	0.2	0.2	2500	80	80
2500-100	1	1	6250000	0.5	0.6	2500	46	58
2500-10	1	1	6250000	1.7	1.7	2500	35	35
3000-1000	2	2	9000000	0.2	0.2	3000	78	78
3000-100	1	1	9000000	0.5	0.6	3000	53	56
3000-10	1	1	9000000	1.0	1.0	3000	23	23
Partial average	14	8	3791667	0.9	1.1	1750	54	61
gs00250a-1	1	1	62500	2.4	4.6	250	60	87
gs00250b-1	1	1	62500	5.6	6.3	250	50	54
gs00250c-1	1	1	62500	14.4	18.4	250	43	52
gs00500a-1	1	1	250000	1.3	1.3	500	48	48
gs00500b-1	1	1	250000	1.8	2.2	500	24	30
gs00500c-1	1	1	250000	5.1	6.1	500	23	27
gs00750a-1	1	1	562500	1.0	1.0	750	48	48
gs00750b-1	1	1	562500	1.6	1.6	750	26	26
gs00750c-1	1	1	562500	3.0	3.0	750	17	17
ga00250a-1	1	1	62500	2.5	4.3	250	62	85
ga00250b-1	1	1	62500	5.5	5.9	250	48	50
ga00250c-1	1	1	62500	13.5	16.9	250	40	47
ga00500a-1	1	1	250000	1.4	2.4	500	53	31
ga00500b-1	1	1	250000	2.4	1.4	500	31	24
ga00500c-1	1	1	250000	5.2	6.3	500	23	30
ga00750a-1	1	1	562500	1.0	1.0	750	44	44
ga00750b-1	1	1	562500	1.3	1.5	750	23	24
ga00750c-1	1	1	562500	2.9	2.7	750	16	15
Partial average	1	1	291667	4.0	4.8	500	38	41
Global average	7	4	2041667	2.5	3.0	1125	46	51

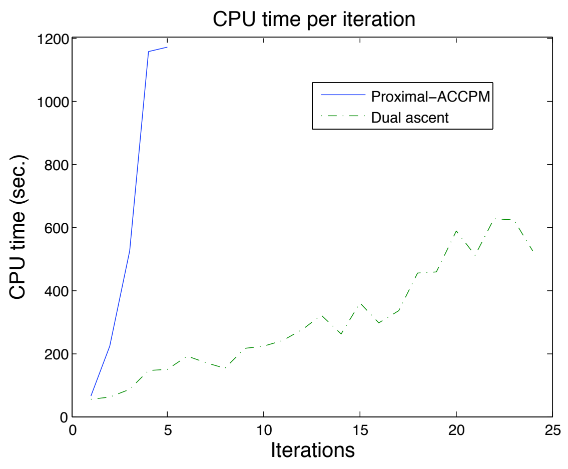

Figure 1: CPU per iteration in the K-median instance 1500-10.

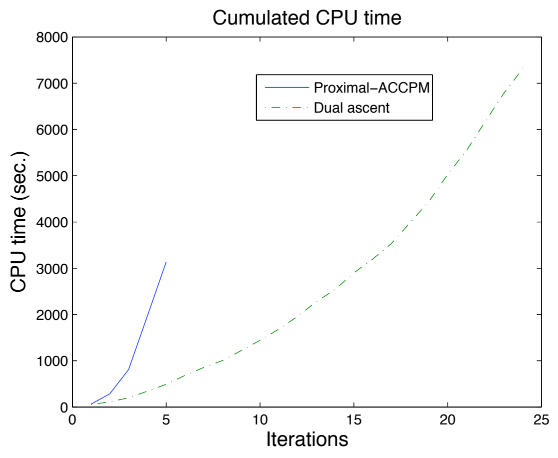

Figure 2: Cumulated CPU for the K-median instance 1500-10.

SLR CPU time is very competitive, since at the price of an extra 64% of CPU time, we solve up to optimality 16 instances. For the two remaining instances we can evaluate its quality by the SLR lower bound. For the Koerkel-Ghosh instances SLR is less competitive (quality solution and CPU time) compared to [RW06]. However, for these instances and by using the SLR bounds we can determine for the first time that, on average, the [RW06] solutions are at least 99.18% optimal.

5 Conclusions

This paper is a continuation of [BTV06], which introduced the semi-Lagrangian relaxation (SLR). In that paper large-scale instances of the p-median problem were successfully solved by SLR. Two key ingredients were present: first, the SLR drastically reduced the number of

Table 5: Performance: 'PACCPM' stands for proximal-ACCPM and 'DA' for dual ascent.

Instance	Nb. of completed iterations			Total CPU time (sec.)			
	Oracle 1	Oracle 2		Oracle 1	Oracle 1 + Oracle 2		[RW06]
	PACCPM (5.a)	DA (5.b)	PACCPM (5.c)	PACCPM (5.d)	DA (5.e)	PACCPM (5.f)	(5.g)
500-1000	128	1	1	2.7	3	4	33
500-100	100	1	1	1.9	2	2	33
500-10	177	1	1	4.1	5	5	24
1000-1000	121	1	1	3.5	6	6	174
1000-100	276	2	2	11.8	16	19	149
1000-10	535	29	5	46.1	4024	1061	142
1500-1000	128	1	1	5.0	9	10	348
1500-100	311	1	1	19.0	22	22	379
1500-10	327	24	5	19.3	7438	3156	387
2000-1000	137	1	1	7.0	12	12	718
2000-100	245	1	1	16.7	23	23	651
2000-10	408	40	4	37.7	2256	661	760
2500-1000	136	1	1	9.4	18	18	1420
2500-100	342	2	2	36.1	55	89	1128
2500-10	754	1	1	177.7	8214	8214	1309
3000-1000	160	5	3	14.0	76	63	1621
3000-100	427	10	8	64.1	7355	8428	1978
3000-10	413	1	1	60.5	90	90	2081
Partial average	285	7	2	29.8	1646	1216	741
gs00250a-1	209	11	5	7.1	8173	5765	6
gs00250b-1	165	11	3	2.9	9145	3845	8
gs00250c-1	252	37	13	5.2	7831	7929	7
gs00500a-1	195	0	0	9.5	10	10	40
gs00500b-1	191	2	1	10.2	2771	515	52
gs00500c-1	143	14	5	7.1	7241	7573	57
gs00750a-1	192	0	0	12.9	13	13	118
gs00750b-1	143	1	1	11.3	5436	5234	127
gs00750c-1	140	8	2	10.7	7274	1868	137
ga00250a-1	177	9	4	5.9	7389	6567	5
ga00250b-1	229	12	3	4.6	7283	2223	8
ga00250c-1	156	36	13	2.5	7675	7804	8
ga00500a-1	233	0	0	9.9	10	10	44
ga00500b-1	229	3	1	4.6	8899	478	53
ga00500c-1	150	16	5	5.3	7936	6604	51
ga00750a-1	199	0	0	14.1	14	14	113
ga00750b-1	137	1	1	10.6	3134	3181	126
ga00750c-1	126	10	2	7.7	9018	1554	130
Partial average	181	10	3	7.9	5514	3399	61
Global average	233	8	3	18.8	3580	2307	401

relevant variables; and second, the SLR decomposed the relaxed p-median instances into many smaller (easier) subproblems. In the current paper, the decomposition of the uncapacitated facility location (UFL) instances induced by the SLR is far less important. Nonetheless, the variable reduction is very high. In this way, SLR is able to solve many previously unsolved ULF instances: our implementation of SLR often provides an optimal primal solution, because it benefits of the strong combinatorial property of a zero duality gap.

We proposed an extension of Koerkel dual multi-ascent method to solve the general SLR problem. We proved convergence, a property that does not hold for the LR problem. We studied the properties of the SLR dual problem in the UFL case. From the properties of the SLR, we know that the optimal set of the SLR dual problem is unbounded. We have shown that for the UFL problem we can restrict our search to a box whose definition depends on the problem costs. This property does not turn out to be very helpful in solving practical instances with the dual ascent method. This is not the case with with the proximal analytic center cutting plane method: the explicit use of this box improves the performance of this algorithm.

An interesting finding of our investigations is that the two methods we used to handle the master program, produce dual variables that keep the dimension of the UFL oracle very low. With the dual ascent method, 97.5% of the primal subproblem variables are fixed (to zero) in average. The same figure for Proximal-ACCPM, is 97.0%. The difference seems to be negligible, yet the computations show that it is sufficient to make the subproblem substantially harder with Proximal-ACCPM. This would give an edge to the dual ascent, but Proximal-ACCPM still achieves a 65% reduction of the total CPU, because it requires nearly 3 times less subproblem solves.

The SLR approach has solved in two hours or less, 18 out of 36 previously unsolved UFL instances from the UflLib. For the remaining 18 still unsolved UFL instances, we have improved the Lagrangian lower bound and confirmed that the Hybrid Multistart heuristic of [RW06] provides near optimal solutions (over 99% optimal). In this study, the Proximal-ACCPM has slightly better performance than the dual ascent: it has produced similar and sometimes better solutions with a CPU time reduction of 35%. Within the 2 hours CPU time limit, Proximal-ACCPM and the dual ascent method have fully solved 18 and 16 UFL instances, respectively. The advantage of the dual ascent method is its extreme simplicity compared to Proximal-ACCPM.

The final conclusion of this paper is that in the framework of the SLR, the combination of an efficient MIP solver (as CPLEX) and an effective dual solver (as Proximal-ACCPM), represents a competitive approach to tackle some combinatorial problems.

Acknowledgments

We are thankful to Mato Baotic (Swiss Federal Institute of Technology) for its Matlab/Cplex interface. We are also thankful to Martin Hoefer, (Max Planck Institut Informatik) for the UFL data from the UflLib.

References

[ACCF88] S. Ahn, C. Cooper, G. Cornuejols, and A. M. Frieze. Probabilistic analysis of a relaxation for the k-median problem. *Mathematics of Operations Research*, 13:1–31, 1988.

[ASV07] P. Avella, A. Sassano, and I. Vasil'ev. Computational study of large-scale p-median problems. *Mathematical Programming*, 109(1):89–114, 2007.

[Bao04] M. Baotic. Matlab/cplex interface. http://control.ee.ethz.ch/ hybrid/cplexint.php, 2004.

[BBH$^+$06] F. Babonneau, C. Beltran, A. B. Haurie, C. Tadonki, and J.-Ph. Vial. *Proximal-ACCPM: a versatile oracle based optimization method*, pages 200–224. Advances in Computational Economics, Finance and Management Science, volume of books series on Advances on Computational Management Science, Kontoghiorghes, E. J. and Gatu, C. (Editors). Springer, 2006.

[BC99] F. Barahona and F. Chudak. Near-optimal solutions to large scale facility location problems technical report. Technical Report RC21606, IBM Watson Research Center, 1999.

[Belar] C. Beltran-Royo. A conjugate Rosen's gradient projection method with global line search for piecewise linear concave optimization. *European Journal of Operational Research*, 2007 (to appear).

[BN04] O. Briant and D. Naddef. The optimal diversity management problem. *Operations Research*, 52(4), 2004.

[BTV06] C. Beltran, C. Tadonki, and J.-Ph. Vial. Solving the p-median problem with a semi-Lagrangian relaxation. *Computational Optimization and Applications*, 35(2):239–260, 2006.

[CC90] A. R. Conn and G. Cornuéjols. A projection method for the uncapacitated facility location problem. *Mathematical programming*, 46:373–398, 1990.

[CJPR83] D. Ch. Cho, E. L. Johnson, M. W. Padberg, and M. R. Rao. On the uncapacitated facility location problem I: Valid inequalities and facets. *Mathematics of Operations Reserach*, 8(4):590–612, 1983.

[CPR83] D. Ch. Cho, M. W. Padberg, and M. R. Rao. On the uncapacitated facility location problem II: Facets and lifting theorems. *Mathematics of Operations Reserach*, 8(4):590–612, 1983.

[dMV02] O. du Merle and J.-Ph. 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.

[Erl78] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26:992–1009, 1978.

[Gho03] D. Ghosh. Neighborhood search heuristics for the uncapacitated facility location problem. *European Journal of Operational Research*, 150:150–162, 2003.

[GHV92] J.-L. Goffin, A. Haurie, and J.-Ph. Vial. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science*, 37:284–302, 1992.

[Gui88] M. Guignard. A Lagrangean dual ascent algorithm for simple plant location problems. *European Journal of Operational Research*, 35:193–200, 1988.

[Gui03] M. Guignard. Lagrangian relaxation. *TOP (Journal of the "Spanish Society of Statistics and Operations Research")*, 11(2):151–228, 2003.

[GV02] J.-L. Goffin and J.-Ph. Vial. Convex nondifferentiable optimization: A survey focussed on the analytic center cutting plane method. *Optimization Methods and Software*, 17(805-867), 2002.

[Hoe03] M. Hoefer. Experimental comparison of heuristic and approximation algorithms for uncapacitated facility location. In *Lecture Notes in Computer Science, Vol. 2647*, pages 165–178. (WEA: experimental and efficient algorithms (2nd International workshop)), Springer-Verlag, 2003.

[Hoe06] M. Hoefer. Ufllib, 2006.
http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/.

[HUL96] J. B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*, volume I and II. Springer-Verlag, Berlin, 1996.

[Kel60] J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.

[KI05] Y. Kochetov and D. Ivanenko. Computationally difficult instances for the uncapacitated facility location problem. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: progress as real solvers*. Springer, 2005.

[Koe89] M. Koerkel. On the exact solution of large-scale simple plant location problems. *European Journal of Operational Research*, 39(2):157–173, 1989.

[MBH06] N. Mladenovic, J. Brimberg, and P. Hansen. A note on duality gap in the simple plant location. *European Journal of Operational Research*, 174(1):11–22, 2006.

[MF90] P. Mirchandani and R. Francis. *Discrete Location Theory*. John Wiley and Sons, 1990.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, 1988.

[RW06] M. G. G. Resende and R. F. Werneck. A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174(1):54–68, 2006.