Comput Optim Appl (2012) 51:387–409
DOI 10.1007/s10589-010-9338-2

Semi-Lagrangian relaxation applied to the uncapacitated facility location problem

C. Beltran-Royo · J.-P. Vial · A. Alonso-Ayuso

Received: 24 July 2009 / Published online: 29 June 2010

Abstract We show how the performance of general purpose Mixed Integer Programming (MIP) solvers, can be enhanced by using the Semi-Lagrangian Relaxation (SLR) method. To illustrate this procedure we perform computational experiments on large-scale instances of the Uncapacitated Facility Location (UFL) problems with *unknown* optimal values. CPLEX solves 3 out of the 36 instances. By combining CPLEX with SLR, we manage to solve 18 out of the 36 instances and improve the best known lower bound for the other instances. The key point has been that, on average, the SLR approach, has reduced by more than 90% the total number of relevant UFL variables.

Keywords Lagrangian relaxation · Mixed integer programming · Uncapacitated facility location (UFL) problem

This work has been partially supported by the Spanish government subsidy MTM2009-14039-C06-03 and by the 'Comunidad de Madrid' subsidy S2009/esp-1594.

C. Beltran-Royo (✉) · A. Alonso-Ayuso
Statistics and Operations Research, Rey Juan Carlos University, Madrid, Spain
e-mail: cesar.beltran@urjc.es

A. Alonso-Ayuso
e-mail: antonio.alonso@urjc.es

J.-P. Vial
University of Geneva, Geneva, Switzerland

J.-P. Vial
ORDECSYS, Geneva, Switzerland
url: www.ordecsys.com

1 Introduction

Implementing an exact method for large scale MIP problems usually requires devising efficient cuts, efficient branching strategies, etc. to exploit the special structure of the problem at hand. In this paper we explore another possibility: to solve large scale MIP problems by combining the SLR approach with a general MIP solver. The idea is to take advantage of the steadily increasing power of general MIP codes. The aim of combining a general MIP solver with SLR is not to obtain the fastest procedure to solve a problem, but to enhance the performance of the MIP solver at low programming cost. Here we use the UFL problem, as a benchmark to illustrate the SLR approach. In order to obtain the best performance, one should use Erlenkotter type algorithms [14], which are especially designed for the UFL problem.

The Semi-Lagrangian Relaxation (SLR) method was introduced in [5] to solve the p-median problem. In order to solve a combinatorial problem, the SLR method has two main advantages compared to the Lagrangian relaxation (LR): The SLR method closes the duality gap and gives an optimal integer solution as a byproduct. The disadvantage of the SLR method is that the relaxed problem is more difficult to solve than in the case of the LR.

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 perform computational experiments on two main collections of UFL problems with unknown optimal values. CPLEX solves 3 out of the 36 instances. On one collection, we manage to solve to optimality 16 out of the 18 problems. On the second collection we solve 2 out of 18 instances. Nevertheless, we are able to improve the Lagrangian lower bound in this collection and thereby confirm that the Hybrid Multistart heuristic of [32] 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 [31]. See [29] for a survey. [27] and [24] improve the famous dual-based procedure of Donald Erlenkotter [14], where the condensed dual of the LP relaxation for the UFL problem is heuristically maximized by the coordinate ascent method. This procedure is followed by a Branch-and-Bound if necessary. [11] propose an projection method to improve the coordinate ascent method in order to compute an exact optimum of the condensed dual problem. [15] adapt Erlenkotter's method to effectively solve the so called two-echelon location problem, which generalizes the UFL 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 [16, 33], a combination of Tabu search, Genetic Algorithm and Greedy Randomized Adaptative Search in [26] and Hybrid Multistart heuristic in [32]. Other contributions have investigated enhanced versions of the UFL problem, as the two-stage UFL [28] or have investigated new computational techniques applied to the UFL problem, as parallel interior point methods [12].

Lagrangian relaxation [20] has also been used to solve the UFL problem, though with less success than the LP dual-based procedures. For example [19] proposes to strengthen the Lagrangian relaxation (equivalent to the LP relaxation) by using Benders inequalities. From a more theoretical point of view, [30] 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 [8–10].

This paper is organized as follows: In Sect. 2 we review the main properties of the SLR and discuss two algorithms to solve its associated dual problem: Proximal-ACCPM and the dual ascent algorithm. In Sect. 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 [5]. In this section, we summarize the main results obtained in that paper and simplify the proofs given there.

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

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

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 X is a polyhedral set, $0 \in S$ 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 function defined as

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

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 $\mathcal{L}(u) = (c - A^T u)^T x(u) + b^T u$. We denote U^* the set of optimal solutions of problem (2). Finally, given two sets S_1 and S_2, its addition corresponds to $S_1 + S_2 = \{s_1 + s_2 : s_1 \in S_1 \text{ and } s_2 \in S_2\}$.

Theorem 1 *The following statements hold* [5].

1. $\mathcal{L}(u)$ *is concave and* $b - Ax(u)$ *is a subgradient 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. $U^* + \mathbb{R}^m_+ = U^*$; *thus* U^* *is an unbounded (convex) set.*
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).

Now, we simplify the proof given in [5].

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

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

holds for any pair u, u'. This shows that $\mathcal{L}(u)$ is concave and that $(b - Ax(u))$ is a subgradient 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 $0 \notin \partial\mathcal{L}(u')$, the subdifferential of $\mathcal{L}$ at u' [22], 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 $\mathcal{L}(u)$. Furthermore, U^* is convex since it is the optimal set of a concave function [22].

To prove the fourth 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).

To prove the fifth statement, assume now $u \in \text{int}(U^*)$. 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). □

To close this section we present the two methods we use to solve the SLR dual problem (2): The proximal analytic center cutting plane method (proximal-ACCPM) and dual ascent method. 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 [5] 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., [6]) 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 [27] to solve the Erlenkotter's 'condensed' dual of the UFL problem [14]. In the case of LR, the dual multi-ascent method does not necessarily converge. In the case of the SLR we prove finite convergence.

2.1 The proximal analytic center cutting plane method

Function $\mathcal{L}(u)$ in problem (2) is by construction, concave and nonsmooth (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 (2). See, for instance, [18] and references therein included; see also [5] 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.

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

$$L_K = \{(u, z) \in \mathbb{R}^{n+1} \mid u \in \mathbb{R}^n,\ z \leq \mathcal{L}^k(u)\ \forall k \leq K,\ z \geq \theta_l\}, \tag{6}$$

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 of problem (2).
4. Update the lower bound θ_l and add the new cuts in the definition of the localization set (6).

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 [25] may prove disastrous, because it overemphasizes the global approximation property of the localization set. Safer methods, as for example bundle methods [21] or Proximal-ACCPM [2, 13, 17, 18], 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 of the localization set plus a quadratic proximal term which ensures the existence of a unique minimizer. 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.

2.2 A dual ascent method

We state the algorithm first and then prove finite convergence. This algorithm, at each iteration, increases some or all of the coordinates of the dual iterate u^k, by at least $\Delta > 0$.

Algorithm 1 Dual ascent algorithm (basic iteration)

1. Solve the oracle: Compute
$$x^k = \arg\min_x \{c^T x + (u^k)^T (b - Ax) \mid Ax \leq b,\ x \in S\},$$
where u^k is the current dual iterate.
2. Stopping criterion: If $s^k := b - Ax^k = 0$, stop. (x^k, u^k) is an optimal primal-dual point.
3. Update the dual iterate: For $j = 1, \ldots, n$, set
$$u_j^{k+1} = \begin{cases} u_j^k + \delta_j^k & \text{if } s_j^k > 0, \\ u_j^k & \text{otherwise,} \end{cases}$$
where $\delta_j^k \geq \Delta$.

We now prove finite convergence of the above algorithm.

Theorem 2 *The following statements hold.*

1. *Algorithm 1 is a dual ascent method when applied to solve the SLR dual problem: for any two consecutive iterates u^k and u^{k+1} we have $\mathcal{L}(u^{k+1}) > \mathcal{L}(u^k)$.*
2. *Let us suppose that $u^0 \geq 0$ and that $U^* \neq \emptyset$. Algorithm 1 converges to an optimal dual 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 3). Thus, $u^{k+1} > u^k$ and by Theorem 1.2 we have that $\mathcal{L}(u^{k+1}) > \mathcal{L}(u^k)$.

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:

$$u_1^{k_i} \geq u_1^{k_0} + i\,\Delta. \tag{7}$$

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

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

$$\begin{aligned}
\mathcal{L}(u^{k_i}) &= c^T x(u^{k_i}) + (b - Ax(u^{k_i}))^T u^{k_i} \\
&= c^T x(u^{k_i}) + (s^{k_i})^T u^{k_i} \\
&\geq (s^{k_i})^T u^{k_i} \\
&= \sum_{j \in J^{k_i}} s_j^{k_i} u_j^{k_i} \\
&\geq u_1^{k_i} \eta \\
&\geq (u_1^{k_0} + i\,\Delta)\eta.
\end{aligned}$$

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

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 'clients' indexed by $J = \{1, \ldots, n\}$. The purpose is to open facilities relative to the set of clients, and to assign each client to a single facility. The cost of an assignment is the sum of the shortest distances c_{ij} from a client to a facility plus the fixed costs of opened facilities f_i. The distance is sometimes weighed by an appropriate factor, e.g., the demand at a client node. The objective is to minimize this sum. The UFL problem is NP-hard [29] 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

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

$x_{ij} = 1$ if facility i serves the client 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 (note that, now, we keep the equality constraint (8b) as an inequality)

$$\mathcal{L}(u) = \min_{x,y} \quad f(u, x, y) \tag{11a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} \leq 1, \quad j \in J, \tag{11b}$$

$$x_{ij} \leq y_i, \quad i \in I, j \in J, \tag{11c}$$

$$x_{ij}, \ y_i \in \{0, 1\}, \tag{11d}$$

where

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

As in the previous section, we denote $(x(u), y(u))$ an optimal point for the oracle (11). The associated Lagrangian relaxation that we use in this paper, corresponds to formulation (11) without constraints (11b).

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 for each client j, its *best combined costs* as $\tilde{c}_j := \min_i\{c_{ij} + f_i\}$. The vector of best combined costs is thus $\tilde{c} := (\tilde{c}_1, \ldots, \tilde{c}_n)$. Furthermore, for each client j, we sort its costs c_{ij}, and get the *sorted costs*

$$c_j^1 \leq c_j^2 \leq \cdots \leq c_j^m.$$

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

Proof Consider the oracle

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

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 $u \in \text{int}(U^*)$, then $u \geq c^1$.*

Proof Let us assume that $u_{j_0} < c_{j_0}^1$ for some $j_0 \in J$ and see that this contradicts $u \in \text{int}(U^*)$. If $u_{j_0} < c_{j_0}^1$ then $c_{j_0}^k - u_{j_0} > 0$ 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 $\text{int}(U^*)$. □

Corollary 1 *Let us consider the scalar $\epsilon > 0$, the vector $\bar{\epsilon}$, where each component is equal to ϵ, and the box*

$$\mathcal{B} := \{u \in \mathbb{R}^n \mid c^1 < u \leq \tilde{c} + \bar{\epsilon}\}.$$

Then, for the UFL problem one has

$$\text{int}(U^*) \cap \mathcal{B} \neq \emptyset.$$

This corollary implies that the search for an optimal dual point can be confined to a box. In particular, taking $\overline{u} = \tilde{c} + \bar{\epsilon}$ and solving the oracle for $\overline{u}$, 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 $\overline{u}$. It is also, likely that 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 dual methods that increase the dual iterates in small steps.

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 small 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, one fixes some variables of the oracle and obtains a reduced-size oracle called the *core subproblem*. (See, for instance, [1, 7].)

We now define the core UFL 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 (client) and the edge e_{ij} exists if facility i can serve client 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 point 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*.

It is easy to see that 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 $x(u)$. The advantage of solving oracle (11) with core graph $G(u)$ is that in $G(u)$ we may have much less edges (variables x_{ij}) and facility nodes (variables y_j), as we will see in Sect. 4.6. 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 decompose the core subproblem into smaller (easier) subproblems.

We now study the inverse image of the mapping $G(u)$. In other words, we wish to describe the set of $\overline{u}$'s that have the graph $\overline{G}$ as a common image through the mapping $G(u)$. It is easy to see that it is a simple box.

To define an elementary box, we use the sorted costs to partition the domain of each coordinate u_j into intervals of the form $]c_j^k,\ c_j^{k+1}]$, with the convention that $c_j^{m+1} = +\infty$. Note that some boxes may be empty (if $c_j^k = c_j^{k+1}$). These coordinate partitions induce a partition of the box $\mathcal{B}$ into elementary boxes. It is not difficult to show that, there is a bijection between core graphs $G(u)$ and elementary boxes. Therefore, it is enough to restrict the dual search to one representative point per elementary box. 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 dual search can be confined to the box $\mathcal{B}$. The discussion of the previous section shows that $\mathcal{B}$ can be partitioned into elementary boxes and that it is enough to restrict the dual search to one representative point per elementary box. Our choice is to take this representative point, as small as possible within each elementary box. Since an elementary box is determined by the intervals $]c_j^k,\ c_j^{k+1}]$, we choose the representative point to be $u_j = c_j^{l(j)} + \epsilon$ $(j = 1, \ldots, n)$, for some fixed $\epsilon > 0$.

Our algorithmic scheme is as follows: Take a small $\epsilon > 0$ and for each client j, take some $l(j) \in I$ (see Sect. 4.1 for details). Start the dual search at $u_j^0 := c_j^{l(j)} + \epsilon$ (each client node will have exactly $l(j)$ 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 client, that is, we update u_j from $c_j^{l(j)} + \epsilon$ to $c_j^{l(j)+1} + \epsilon$, for some $l(j) \in I$. We only update the coordinates of the dual iterate u^k whose corresponding subgradient coordinate 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 client $j \in J = \{1, \ldots, n\}$, set
   (a) $u_j^0 = c_j^{l(j)} + \epsilon$ for some $l(j) \in I = \{1, \ldots, m\}$,
   (b) $\tilde{c}_j = \min_i \{c_{ij} + f_i\}$,
   (c) $c_j^{m+1} = +\infty$.
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 point.
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-ascent.

4 Computational experiments

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. The CPU time limit is set to 7200 seconds. First, we compare our combined SLR-CPLEX approach to the plain CPLEX. Second, we compare our results to the results reported in [32] and in the *Uncapacitated Facility Location Library* (UflLib) [23].

4.1 Parameter setting for the algorithms

Programs have been written in MATLAB 7.0 and run on a PC (Pentium-IV, 3.0 GHz, with 3 GB of RAM memory) under the Windows XP operating system. The reader should bear in mind that results reported in [32] were found on a different machine (SGI Challenge with 28 196-MHz MIPS R10000 processors, although 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 [2], 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 [3].

The dual ascent algorithm is implemented as stated in the previous section without parameter tuning ($\epsilon = 10^{-3}$). Proximal-ACCPM is used in its generic form as described in [2] with some tuning. We use Proximal-ACCPM in a two phase scheme: in the first phase we solve the Lagrangian relaxation (LR) dual problem. In the second phase we solve the semi-Lagrangian relaxation (SLR) dual problem. The Proximal-ACCPM parameter setting is as follows: In 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, we choose to fix the reference proximal point at the initial point u^0. As pointed out in [5], an optimal LR dual point u^*_{LR} is a good starting point u^0 for the SLR method. To be more precise, in our UFL computational experiments, for each client j, we take the $c_j^{l(j)}$ which is the closest to $u^*_{LR\,j}$ and define $u^0_j := c_j^{l(j)} + \epsilon$, for some small $\epsilon > 0$ (as we saw in Algorithm 2). The proximal weight ρ has been fixed, by tuning, to 10^{-4} for the Barahona-Chudak instances and to 10^{-3} for the Koerkel-Ghosh instances.

4.2 Instance description

For our test we use 36 *unsolved* UFL instances that can be obtained in the UFL library UflLib (see Table 1). The first set, with 18 instances, is called Barahona-Chudak [4]. In the Euclidian plane n points are randomly generated in the unit square $[0, 1] \times [0, 1]$. Each point simultaneously represents a facility and a client ($m = n$), with $n = 500, 1000, \ldots, 3000$. The connection costs c_{ij} are determined by the Euclidian distance. In each instance all the fixed costs f_i are equal and calculated by $\sqrt{n}/l$ with $l = 10, 100$ or 1000. All values are rounded up to 4 significant digits and made integer [23]. We use the label $n - l$ to name these instances.

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

Barahona-Chudak instances	Nb. of clients	Fix cost	Koerkel-Ghosh instances	Nb. of clients	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]

The second set of UFL instances is called Koerkel-Ghosh. In these instances, the connection costs c_{ij} are drawn uniformly at random from [1000, 2000]. The fixed costs f_i 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 [16] 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 CPLEX performance

In Table 2 we report the results obtained with CPLEX 9.1 (default settings and 7200 seconds as CPU time limit). Barahona-Chudak instances: We observe that CPLEX has solved the first three instances. However, the remaining 15 instances have become too large, even to solve their LP relaxation. Koerkel-Gosh instances: CPLEX has given an integer solution and a lower bound in most of the cases. Cases gs00750a-1 and ga00750a-1 have become too large at the beginning of the B&B process (CPLEX has been able to solve their LP relaxation and compute an integer solution). For the other 4 instances with 750 points, CPLEX has not even solved the LP relaxation within 7200 seconds.

Table 2 CPLEX performance: 'UB' stands for upper bound, 'LB' for lower bound. Larger instances have produced an 'Out of memory' either before CPLEX solved the LP relaxation or after ('Before LP' and 'After LP')

Instance	UB	LB	B&B gap (%)	CPLEX time (sec.)	Out of memory
500-1000	99169	99169	0.00	12	
500-100	326790	326790	0.00	4	
500-10	798577	798577	0.00	35	
1000-1000	–	–	–	–	Before LP
1000-100	–	–	–	–	Before LP
1000-10	–	–	–	–	Before LP
1500-1000	–	–	–	–	Before LP
1500-100	–	–	–	–	Before LP
1500-10	–	–	–	–	Before LP
2000-1000	–	–	–	–	Before LP
2000-100	–	–	–	–	Before LP
2000-10	–	–	–	–	Before LP
2500-1000	–	–	–	–	Before LP
2500-100	–	–	–	–	Before LP
2500-10	–	–	–	–	Before LP
3000-1000	–	–	–	–	Before LP
3000-100	–	–	–	–	Before LP
3000-10	–	–	–	–	Before LP
gs00250a-1	257999	257803	0.08	7200	
gs00250b-1	276892	274324	0.94	7200	
gs00250c-1	332935	324142	2.71	7200	
gs00500a-1	512237	510446	0.35	7200	
gs00500b-1	654860	533027	22.86	7200	
gs00500c-1	746506	601986	24.01	7200	
gs00750a-1	785232	762565	2.97	5023	After LP
gs00750b-1	–	–	–	7200	
gs00750c-1	–	–	–	7200	
ga00250a-1	257991	257778	0.08	7200	
ga00250b-1	276556	274112	0.89	7200	
ga00250c-1	334735	324892	3.03	7200	
ga00500a-1	512071	510614	0.29	7200	
ga00500b-1	649226	533339	21.73	7200	
ga00500c-1	747626	602861	24.01	7200	
ga00750a-1	783973	762464	2.82	5707	After LP
ga00750b-1	–	–	–	7200	
ga00750c-1	–	–	–	7200	

4.4 Dual solution quality

In Table 3 we report the dual solution quality. We observe an important result: the semi-Lagrangian relaxation closes the duality gap (if convergence is reached). Column (3.a) reports the best known upper bound (UB) to the optimal primal cost. In the remaining 6 columns we report the results concerning the Lagrangian relaxation (LR) and the semi-Lagrangian relaxation (SLR) bounds.

In column (3.b) we have the LR lower bound (obtained with Proximal-ACCPM). In column (3.c) we give the LR duality gap (in fact we give an upper bound to the true duality gap since in some cases we do not know the true primal optimum but an upper bound). In the remaining 4 columns (3.d–g), we report the SLR lower bound information, obtained either with the dual ascent method o with Proximal-ACCPM. We also give the optimality gap of these dual values. Optimal SLR dual bounds have been written in boldface. In the remaining cases, the SLR procedure was stopped before reaching optimality because of the CPU time limit (7200 seconds).

4.5 Primal solution quality

In Table 4 we report the primal solution quality. Column (4.a) reports the best known lower bound. In column (4.b), we have the upper bounds (UB) to the UFL optima, reported in literature. In the first half of this column, we have the results for the Barahona-Chudak instances reported in [32], (they were obtained as the average of several runs). In the second half of this column, we have the results for the Koerkel-Ghosh instances as reported in UflLib (the results reported in [32] for these instances do not correspond to single instances but for groups of 5 instances). In columns (4.d) and (4.f) we have the UB obtained in the framework of the dual ascent method and Proximal-ACCPM, respectively.

In this section, the important result is that the SLR approach has been able to solve 18 instances of the 36 previously *unsolved* ones. These 18 instances have been solved by Proximal-ACCPM. On the other hand, the dual ascent method has been able to solve 16 UFL instances.

For the unsolved instances we have reported the upper bound obtained by the following simple heuristic which computes a primal feasible solution. At each iteration of the SLR method the oracle solution $y(u^k)$ proposes to open a set of facilities. To complete the solution, each client is assigned to its closest open facility. If the SLR method fails to converge, we take the best heuristic solution as the primal SLR solution. In general, the sophisticated heuristic used in [32] performs better than our simple primal heuristic. At the same time, the SLR procedure combined with this simple heuristic, has given better primal solutions than CPLEX.

4.6 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, after SLR, one may have from several to only one

Table 3 Dual solution quality: 'UB' stands for upper bound, 'LB' for lower bound, 'PACCPM' for Proximal-ACCPM, 'DA' for dual ascent. Optimal lower bounds are in boldface

Instance	UB	LB					
		Lagrangian (PACCPM)		Semi-Lagrangian (Dual ascent)		Semi-Lagrangian (PACCPM)	
		Bound	Dual gap (%)	Bound	Optimality gap (%)	Bound	Optimality gap (%)
	(3.a)	(3.b)	(3.c)	(3.d)	(3.e)	(3.f)	(3.g)
500-1000	99169	99160	0.0091	**99169**	0	**99169**	0
500-100	326790	**326790**	0	**326790**	0	**326790**	0
500-10	798577	**798577**	0	**798577**	0	**798577**	0
1000-1000	220560	220559	0.0003	**220560**	0	**220560**	0
1000-100	607878	607814	0.0105	**607878**	0	**607878**	0
1000-10	1434154	1433389	0.0534	**1434154**	0	**1434154**	0
1500-1000	334962	334944	0.0054	**334962**	0	**334962**	0
1500-100	866454	866453	0.0001	**866454**	0	**866454**	0
1500-10	2000801	1999284	0.0758	2000696	0.0052	**2000801**	0
2000-1000	437686	437678	0.0018	**437686**	0	**437686**	0
2000-100	1122748	1122746	0.0002	**1122748**	0	**1122748**	0
2000-10	2558118	2557753	0.0143	**2558118**	0	**2558118**	0
2500-1000	534405	534395	0.0019	**534405**	0	**534405**	0
2500-100	1347516	1347494	0.0016	**1347516**	0	**1347516**	0
2500-10	3100225	3096856	0.1087	3097647	0.0831	3097647	0.0831
3000-1000	643463	643432	0.0048	**643463**	0	**643463**	0
3000-100	1602335	1601652	0.0426	1602063	0.0170	1602120	0.0134
3000-10	3570766	3570752	0.0004	**3570766**	0	**3570766**	0
Partial average	1200367.0	1199985	0.0184	1200203	0.0059	1200212	0.0054
gs00250a-1	257964	257639	0.1259	257899	0.0252	**257964**	0
gs00250b-1	276761	273693	1.1085	275363	0.5051	275574	0.4289
gs00250c-1	332935	322696	3.0753	329135	1.1414	330559	0.7137
gs00500a-1	511229	510408	0.1606	510408	0.1606	510408	0.1606
gs00500b-1	537931	533020	0.913	534029	0.7254	533477	0.8280
gs00500c-1	620041	601962	2.9158	607260	2.0613	609333	1.7270
gs00750a-1	763671	762562	0.1453	762562	0.1452	762562	0.1452
gs00750b-1	797026	790334	0.8396	790917	0.7665	790917	0.7665
gs00750c-1	900454	875340	2.789	879430	2.3348	879343	2.3445
ga00250a-1	257957	257618	0.1315	257882	0.0291	**257957**	0
ga00250b-1	276339	273296	1.1012	275080	0.4556	275200	0.4122
ga00250c-1	334135	322958	3.3451	329839	1.2857	331171	0.8871
ga00500a-1	511422	510587	0.1632	510587	0.1633	510587	0.1633
ga00500b-1	538060	533334	0.8784	534416	0.6772	533837	0.7849
ga00500c-1	621360	602852	2.9787	608518	2.0668	609975	1.8323
ga00750a-1	763576	762462	0.1459	762462	0.1459	762462	0.1459
ga00750b-1	796480	790112	0.7995	790631	0.7344	790630	0.7345
ga00750c-1	902026	875593	2.9304	880301	2.4085	879601	2.4861
Partial average	555520.4	547581	1.3637	549818	0.8795	550087	0.8089
Global average	877943.7	873783	0.6910	875010	0.4427	875149	0.4071

Table 4 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. [32] obtains an upper bound by a heuristic method. Optimal costs for the UFL problem are in boldface

Instance	LB	UB					
		[32]	% Opt.	DA	% Opt.	PACCPM	% Opt.
	(4.a)	(4.b)	(4.c)	(4.d)	(4.e)	(4.f)	(4.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	99.21237
3000-1000	643463	643541.8	99.9878	**643463**	100	**643463**	100
3000-100	1602120	1602530.9	99.9744	1602335	99.9866	1602397	99.98271
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	534029	537931	99.2693	551716	96.6880	551716	96.6880
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	879430	900454	97.6093	971616	89.5175	971616	89.5175
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	534416	538060	99.3181	546253	97.7851	546253	97.7851
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	790631	796480	99.2602	808105	97.7899	808105	97.7899
ga00750c-1	880301	902026	97.5321	1000972	86.2921	1008497	85.4373
Partial average	550193	555521	99.1952	573516	96.5397	571494	97.0326
Global average	875202	877970.5	99.5928	887547	98.2476	886538	98.4940

subproblem. Furthermore, the number of subproblems usually is different for each SLR iteration. For this reason, we do not report the number of subproblems at each iteration, but its average. In Table 5, we have the average number of subproblems (ANS) per SLR iteration. More specifically, columns (5.a) and (5.b) report the ANS for the dual ascent method and for Proximal-ACCPM respectively. For example, in instance 500-1000 the ANS value is 108 and 84 for the dual ascent method and for Proximal.ACCPM, respectively. This means that the SLR method has managed to decomposed this instance. In other 30 instances (instance 500-100 for example) the SLR does not decompose the instance (ANS $= 1$). Note that the decomposition is slightly better with the dual ascent method 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 to 0 many variables). In column (5.c) we have the total number of x_{ij} variables. For example, instance 500-1000 has 250000 x_{ij} variables, but, as we can see in column (5.d) only 0.3% are relevant in the SLR combined with the dual ascent method (the remaining 99.7% are fixed to 0). The number of relevant x_{ij} variables in the case of Proximal-ACCPM is reported in (5.e). The analogous results for the y_i variables can be found in columns (5.f–h). Note that the number of variables is different for each SLR iteration and therefore we give average figures corresponding to all the SLR iterations.

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 (5.d) and (5.g)). In the Proximal-ACCPM case we use a slightly higher number of variables (columns (5.e) and (5.h)).

4.7 Performance

Finally, in Table 6 we report the performance of the dual ascent method and Proximal-ACCPM in terms of the number of SLR iterations and 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.

Proximal-ACCPM reduces by 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 (6.e) and (6.f), is that the average CPU time per SLR iteration 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 5. This produces harder (CPU time consuming) Oracle 2's. (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 (6.f)) versus the results reported in [32] (column (6.g)) we observe that: for the Barahona-Chudak instances, the 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 2 remaining instances we can evaluate its quality by the SLR lower bound. For the

Table 5 Semi-Lagrangian reduction and decomposition: 'ANS' stands for average number of subproblems per SLR iteration, 'DA' for dual ascent, 'PACCPM' for proximal-ACCPM, 'ANX' for average number of relevant x_{ij} variables per SLR iteration (in %) and 'ANY' for average number of relevant y_i variables per SLR iteration (in %)

Instance	ANS		Total	ANX (%)		Total	ANY (%)	
	DA (5.a)	PACCPM (5.b)	Nb. of x_{ij} (5.c)	DA (5.d)	PACCPM (5.e)	Nb. of y_i (5.f)	DA (5.g)	PACCPM (5.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

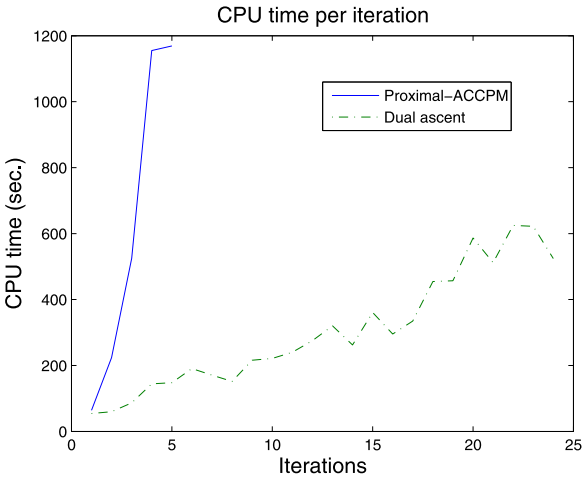

Fig. 1 CPU time per iteration for instance 1500-10

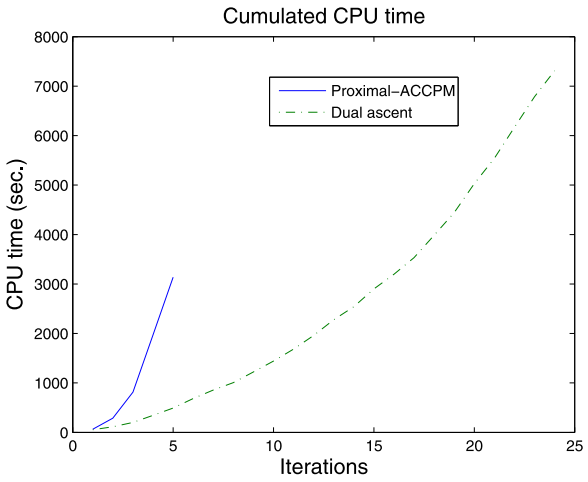

Fig. 2 Cumulated CPU time for the instance 1500-10

Koerkel-Ghosh instances SLR is less competitive (quality solution and CPU time) compared to [32]. However, for these instances and by using the SLR bounds we can determine for the first time that, on average, the [32] solutions are at least 99.18% optimal (Table 4).

5 Conclusions

The contributions of this paper are twofold: empirical and theoretical.

Empirical contribution: We have shown by example that the performance of a general MIP solver, as CPLEX, can be enhanced by combining it with the semi-Lagrangian relaxation (SLR) approach. In our computational experiments, CPLEX

Table 6 Performance: 'PACCPM' stands for proximal-ACCPM and 'DA' for dual ascent. [32] uses an heuristic method

Instance	Nb. of completed iterations			Total CPU time (sec.)			
	LR	SLR		LR	LR + SLR		[32]
	PACCPM (6.a)	DA (6.b)	PACCPM (6.c)	PACCPM (6.d)	DA (6.e)	PACCPM (6.f)	(6.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

solved 3 out of 36 Uncapacitated Facility Location (UFL) unsolved instances from the UflLib. In contrast, by using the SLR together with two standard optimization tools, CPLEX and Proximal-ACCPM, we solved 18 instances. For the remaining 18 still unsolved UFL instances, we have improved the best known lower bound and confirmed, for the first time, that the Hybrid Multistart heuristic of [32] provides near optimal solutions (over 99% optimal). The reason for this good result is that, the SLR drastically reduced the number of UFL relevant variables. Roughly speaking, on the average number of relevant x_{ij} variables was reduced to 3% and the average number of relevant y_i variables, was reduced to 50%.

Also from an empirical point of view, we have compared two dual optimization methods: Proximal-ACCPM and a dual ascent method. Proximal-ACCPM has shown a 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 of CPU time limit, Proximal-ACCPM and the dual ascent method have fully solved 18 and 16 UFL instances, respectively (from a pool of 36 unsolved instances). The advantage of the dual ascent method is its extreme simplicity compared to Proximal-ACCPM.

Theoretical contribution: From a theoretical point of view, this paper has proposed an extension of the Koerkel dual multi-ascent method to solve the SLR dual problem and we have proved (finite) convergence. Furthermore, We have studied the theoretical properties of the SLR dual problem in the UFL case. We have shown that for the UFL problem we can restrict our dual search to a box whose definition depends on the problem costs. This property has not shown to be very useful for the dual ascent method. In contrast, in the case of Proximal-ACCPM the explicit use of this box has slightly improved the dual search.

Acknowledgements We are thankful: To Mato Baotic (Swiss Federal Institute of Technology) for its Matlab/Cplex interface [3], to Martin Hoefer (Max Planck Institut Informatik) for the UFL data from the UflLib [23], and to the anonymous referees for their constructive comments and suggestions.

References

1. Avella, P., Sassano, A., Vasil'ev, I.: Computational study of large-scale p-median problems. Math. Program. **109**(1), 89–114 (2007)
2. Babonneau, F., Beltran, C., Haurie, A.B., Tadonki, C., Vial, J.-Ph.: Proximal-ACCPM: a versatile oracle based optimization method. In: Kontoghiorghes, E.J., Gatu, C. (eds.) Advances in Computational Economics, Finance and Management Science. Advances on Computational Management Science, pp. 200–224. Springer, Berlin (2006)
3. Baotic, M.: Matlab interface for CPLEX (2004). http://control.ee.ethz.ch/hybrid/cplexint.php
4. Barahona, F., Chudak, F.: Near-optimal solutions to large scale facility location problems technical report. Technical Report RC21606, IBM Watson Research Center (1999)
5. Beltran, C., Tadonki, C., Vial, J.-Ph.: Solving the p-median problem with a semi-Lagrangian relaxation. Comput. Optim. Appl. **35**(2), 239–260 (2006)
6. Beltran-Royo, C.: A conjugate Rosen's gradient projection method with global line search for piecewise linear concave optimization. Eur. J. Oper. Res. **182**(2), 536–551 (2007)
7. Briant, O., Naddef, D.: The optimal diversity management problem. Oper. Res. **52**(4), 515–526 (2004)
8. Canovas, L., Landete, L., Marin, A.: On the facets of the simple plant location packing polytope. Discrete Appl. Math. **124**, 27–53 (2002)
9. Cho, D.Ch., Johnson, E.L., Padberg, M.W., Rao, M.R.: On the uncapacitated facility location problem I: Valid inequalities and facets. Math. Oper. Res. **8**(4), 590–612 (1983)

10. Cho, D.Ch., Padberg, M.W., Rao, M.R.: On the uncapacitated facility location problem II: Facets and lifting theorems. Math. Oper. Res. **8**(4), 590–612 (1983)
11. Conn, A.R., Cornuéjols, G.: A projection method for the uncapacitated facility location problem. Math. Program. **46**, 373–398 (1990)
12. de Silva, A., Abramson, D.: A parallel interior point method and its application to facility location problems. Comput. Optim. Appl. **90**(3), 249–273 (1998)
13. du Merle, O., Vial, J.-Ph.: 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)
14. Erlenkotter, D.: A dual-based procedure for uncapacitated facility location. Oper. Res. **26**, 992–1009 (1978)
15. Gao, L.-L., Robinson, E.P.: Uncapacitated facility location: General solution procedure and computational experience. Eur. J. Oper. Res. **76**(3), 410–417 (1994)
16. Ghosh, D.: Neighborhood search heuristics for the uncapacitated facility location problem. Eur. J. Oper. Res. **150**, 150–162 (2003)
17. Goffin, J.-L., Haurie, A., Vial, J.-Ph.: Decomposition and nondifferentiable optimization with the projective algorithm. Manag. Sci. **37**, 284–302 (1992)
18. Goffin, J.-L., Vial, J.-Ph.: Convex nondifferentiable optimization: A survey focussed on the analytic center cutting plane method. Optim. Methods Softw. **179**, 805–867 (2002)
19. Guignard, M.: A Lagrangean dual ascent algorithm for simple plant location problems. Eur. J. Oper. Res. **35**, 193–200 (1988)
20. Guignard, M.: Lagrangian relaxation. TOP **11**(2), 151–228 (2003)
21. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms, vols. I, II. Springer, Berlin (1996)
22. Hiriart-Urruty, J.-B., Lemaréchal, C.: Fundamentals of Convex Analysis. Springer, Berlin (2000)
23. Hoefer, M.: Ufllib (2006). http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/
24. Janacek, J., Buzna, L.: An acceleration of Erlenkotter-Koerkel's algorithms for the uncapacitated facility location problem. Ann. Oper. Res. **164**, 97–109 (2008)
25. Kelley, J.E.: The cutting-plane method for solving convex programs. J. SIAM **8**, 703–712 (1960)
26. Kochetov, Y., Ivanenko, D.: Computationally difficult instances for the uncapacitated facility location problem. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Solvers. Springer, Berlin (2005)
27. Koerkel, M.: On the exact solution of large-scale simple plant location problems. Eur. J. Oper. Res. **39**(2), 157–173 (1989)
28. Landete, M., Marin, A.: New facets for the two-stage uncapacitated facility location polytope. Comput. Optim. Appl. **44**(3), 487–519 (2009)
29. Mirchandani, P., Francis, R.: Discrete Location Theory. Wiley, New York (1990)
30. Mladenovic, N., Brimberg, J., Hansen, P.: A note on duality gap in the simple plant location. Eur. J. Oper. Res. **174**(1), 11–22 (2006)
31. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
32. Resende, M.G.G., Werneck, R.F.: A hybrid multistart heuristic for the uncapacitated facility location problem. Eur. J. Oper. Res. **174**(1), 54–68 (2006)
33. Sun, M.: Solving uncapacitated facility location problems using tabu search. Comput. Oper. Res. **33**, 2563–2589 (2006)