European Journal of Operational Research xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Lagrangean duals and exact solution to the capacitated *p*-center problem

Maria Albareda-Sambola [a], Juan A. Díaz [b], Elena Fernández [a,*]

[a] Dpt. d'Estadística i Investigació Operativa, Universitat Politècnica de Catalunya, Barcelona, Spain
[b] Dpt. Ingeniería Industrial y Mecánica, Universidad de las Américas, Puebla, Mexico

ARTICLE INFO

Article history:
Received 27 July 2007
Accepted 5 February 2009
Available online xxxx

Keywords:
Discrete location
Capacitated *p*-center
Lagrangean relaxation

ABSTRACT

In this work, we address the capacitated *p*-center problem (*CpCP*). We study two auxiliary problems, discuss their relation to *CpCP*, and analyze the lower bounds obtained with two different Lagrangean duals based on each of these auxiliary problems. We also compare two different strategies for solving exactly *CpCP*, based on binary search and sequential search, respectively. Various data sets from the literature have been used for evaluating the performance of the proposed algorithms.

1. Introduction

In *p*-center problems we have to partition a set of customers into exactly *p* clusters. A cluster is defined both by the location of its facility and by the set of its customers. There is a given cost for assigning each customer to each facility, and we want to minimize the maximum assignment cost among all the customers. These are discrete location problems, since facilities must be located within a given set of potential locations. In the capacitated *p*-center problem (*CpCP*) each customer has also a known demand and each potential location a known capacity. Each cluster must be such that the total demand of all its customers does not exceed the capacity of its facility. Thus, *CpCP* is the problem of finding the set of *p* locations and the assignment pattern that satisfies the capacity constraints where the maximum assignment cost is as small as possible.

Typical applications of the *p*-center problem include the location of public facilities when it is required that no customer be too far away from his closest service point. This is the case of emergency services like, for instance, ambulances, hospitals or fire stations. However, most of these services have, in practice, limited capacities. For example, in a very dense small area, a location might exist that is close enough to all the inhabitants, but we cannot assume that one single facility can absorb all the demand in the area. Therefore, capacitated versions of the problem need to be investigated.

The uncapacitated version of the problem has been extensively studied, and both exact and approximate algorithms have been proposed. The paper by Elloumi et al. [5] is a recent work with a comprehensive up-to-date state of the art of the research on *p*-center problems. On the contrary, the capacitated *p*-center problem has received much less attention in the literature, as opposed to the capacitated version of the *p*-median problem for which there is a quite extensive related literature (see, for instance, [4,15,6]). A local search heuristic is presented in [14]. Approximation algorithms for the particular case in which all the demands are the same have been proposed in [2,9], and a polynomial exact algorithm for tree networks is developed in [8]. To the best of our knowledge, there is only one general exact algorithm for *CpCP* in the literature, recently proposed by Özsoy and Pinar [13]. For the particular case when all customer demands are one, and facility capacities are equal, it is possible to extend the subexponential exact algorithm proposed in [1] for the continuous version of the problem.

The exact algorithm of Özsoy and Pinar succeeds to provide optimal solutions in small times for small instances of the problem. However, the algorithm requirements increase fast with the instance dimensions, so that large instances cannot yet be solved efficiently. Clearly, the efficiency of any exact algorithm for *CpCP* strongly depends on the quality of the bounds that can be derived for it, and on the strategy that is used to enumerate the solution space once the bounds are at hand. In this paper we address these two issues. We are not aware of any published work studying lower bounds for *CpCP*. One of the contributions of this paper is the analysis of two auxiliary subproblems and their relation to *CpCP*. More specifically, we propose two different Lagrangean duals based on each of these subproblems and we evaluate how tight their associated bounds are. The two auxiliary problems that we study are the maximum demand coverage with fixed radius problem, denoted PD_δ, and the minimum required centers with fixed

* Corresponding author.
E-mail addresses: maria.albareda@upc.edu (M. Albareda-Sambola), juana.diaz@udlap.mx (J.A. Díaz), e.fernandez@upc.edu (E. Fernández).

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.02.022

Please cite this article in press as: M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated *p*-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022

radius problem, denoted PC_δ. Problem PD_δ consists of finding the maximum demand that can be satisfied with at most p centers within a given coverage radius δ, whereas problem PC_δ consists of finding the minimum number of centers that are needed in order to satisfy customers demands within a given coverage radius δ. For fixed radii δ, we study two different Lagrangean duals based on PD_δ and PC_δ, respectively. Each Lagrangean dual provides both a lower and an upper bound for *CpCP*. We are not aware of any work where problem PD_δ is considered in relation to *CpCP*. Problem PC_δ was used by Özsoy and Pinar in [13] to obtain lower bounds from its linear programming (LP) and mixed integer linear programming (MILP) relaxations. To the best of our knowledge, the Lagrangean duals studied in this work have not been previously considered in the literature neither as a tool for solving the *CpCP* nor in other contexts. This has led us to analyze their structure and to study their solutions. As we will see, the structure of the Lagrangean subproblems allows us to solve them very efficiently. One additional benefit of solving Lagrangean duals is that feasible solutions (and, thus upper bounds) can be easily obtained from the solutions to the Lagrangean subproblems. Our computational experiments assess the quality of the obtained bounds. However, we have observed that in the case of the *CpCP*, the difficulty for solving exactly different instances of the same size with similar optimal values and similar lower and upper bounds, might vary considerably, depending on the number of different radii (values for feasible solutions) within the lower and upper bounds. Indeed, the difficulty for solving exactly a given instance with lower and upper bounds *LB* and *UB*, respectively, increases with the number of radii in the interval $[LB, UB]$. As will be seen, the bounds obtained when solving our Lagrangean duals allow an important reduction on number of considered radii.

In the second part of the paper we propose an exact algorithm for *CpCP* that identifies the optimal solution by iteratively exploring the set of non-eliminated radii. It is in this phase when the beneficial effect of the bounding phase on the overall algorithm can be better appreciated, since the reduction on the number of candidate radii to be considered due to its application, implies a big reduction on the overall computing time, which largely compensates the computation time of the bounding phase. According to our experimental results and taking into account that, roughly speaking, the bounding phase reduces the number of candidate radii by over 85%, if the exact algorithm were used without the bounding phase, a time increase of at least 30%, raising over 100% would result in most of the instances.

For proposing an exact algorithm for the *CpCP*, we analyze two different strategies for selecting the candidate radius at each iteration. In the first one radii are increased sequentially, starting from the lower bound. This strategy was already used by Özsoy and Pinar in [13], starting from their lower bounds. The second strategy uses binary search on the range of candidate radii. As will be seen, the later strategy is more efficient, not only from a worst case analysis point of view but also, in general, empirically, specially for instances with very similar distances, where the set of candidate radii is often large, even when the current percent gap is very small.

This paper is structured as follows. Section 2 gives some notation, defines de problem formally and presents some elimination tests. In Sections 3 and 4, we discuss the two auxiliary problems PD_δ and PC_δ, their relation to *CpCP*, and their corresponding Lagrangean duals and heuristics. Section 5 presents the algorithmic framework to obtain the lower and upper bounds, describes the computational experiments and discusses the obtained results. Next, in Section 6 we present the exact algorithm, we describe the two strategies that have been compared and we analyze the computational results obtained with the best strategy. Finally, in Section 7 we draw some conclusions and give some final remarks.

2. The capacitated *p*-center problem

Let $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, m\}$ be the sets of indices for customers and centers, respectively. Let h_i denote the demand of customer $i \in I$, b_j denote the capacity of a center located at site $j \in J$, and p be the number of centers to locate. For each pair (i,j), $i \in I, j \in J$, let d_{ij} be the distance from customer i to center j.

Then, the goal of *CpCP* is to find a subset of plants of cardinality p to open, $JO \subseteq J$, and an assignment of customers to open plants $i \mapsto j(i) \in JO$, such that each open plant $j^* \in JO$, has assigned a total demand that does not exceed its capacity, $\sum_{i:j(i)=j^*} h_i \leqslant b_{j^*}$, and the maximum assignment cost, $\max_{i \in I} d_{ij(i)}$, is minimized.

Along the paper we use some additional notation. For a given radius δ, for each $j \in J$, I_j^δ will denote the set of customers whose distance to center j does not exceed the radius δ. That is, $I_j^\delta = \{i \in I : d_{ij} \leqslant \delta\}$. Analogously, for a given customer $i \in I$, J_i^δ will denote the set of plants that are within a distance δ from customer i; $J_i^\delta = \{j \in J : d_{ij} \leqslant \delta\}$. Also, we denote by $D^0 < D^1 < \cdots < D^{k_{max}}$ the sorted distinct entries in the distance matrix (d_{ij}), and by $K = \{0, \ldots, k_{max}\}$ the set of their indices.

Obviously, the value of the optimal solution is one of $\{D^0, D^1, \ldots, D^{k_{max}}\}$. However, exploring the full list to find this value would be extremely inefficient, unless a large number of these radii are discarded beforehand. To this end, we obtain lower and upper bounds on the optimal value of *CpCP* and we apply the following elimination tests:

Elimination test 1

If *LB* is a lower bound for *CpCP*, then the set of radii $\{D^0, \ldots, D^{k_l}\}$, where k_l is the largest index such that $D^{k_l} < LB$, can be discarded.

Elimination test 2

Similarly, if *UB* is an upper bound, the optimal solution will take a value that is not in $\{D^{k_u}, \ldots, D^{k_{max}}\}$, where k_u is the smallest index such that $D^{k_u} > UB$.

The above elimination tests allow to reduce the set of candidate radii to $\{D^{k_l+1}, \ldots, D^{k_u-1}\}$. In order to obtain tight lower and upper bounds for *CpCP* that reduce the set of candidate radii as much as possible, in the next two sections, we address two different auxiliary problems.

3. The maximum demand coverage within fixed radius problem

When solving *CpCP*, it can be useful at some point to know the maximum demand that can be satisfied with at most p plants, within a given radius δ. This is the goal in the maximum demand coverage within fixed radius problem (PD_δ).

Defining the two sets of decision variables

$$y_j = \begin{cases} 1 & \text{if a center is located at site } j \in J; \\ 0 & \text{otherwise} \end{cases}$$

and

$$x_{ij} = \begin{cases} 1 & \text{if customer } i \in I \text{ is assigned to a center located at site } j \in J; \\ 0 & \text{otherwise,} \end{cases}$$

we can model this problem as:

$$(PD_\delta) \quad H(\delta) = \max \sum_{j \in J} \sum_{i \in I_j^\delta} h_i x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in J_i^\delta} x_{ij} \leqslant 1 \quad \forall\, i \in I, \tag{2}$$

$$\sum_{i \in I_j^\delta} h_i x_{ij} \leqslant b_j y_j \quad \forall\, j \in J, \tag{3}$$

Please cite this article in press as: M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated *p*-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022

$$\sum_{j\in J} y_j \leqslant p, \tag{4}$$

$$x_{ij} \in \{0,1\}, y_j \in \{0,1\} \quad \forall j \in J,\ i \in I_j^\delta. \tag{5}$$

By constraints (2) each customer is assigned to at most one center. Constraints (3) guarantee that no customer is assigned to a center that is not opened and also that the capacity of the open facilities is not violated. The last constraint (4) limits to at most p the number of open facilities.

For obtaining a stronger model with a tighter LP bound, in constraints (3) we substitute the facility capacities b_j by $\hat{b}_j^\delta$, $\forall j \in J$, where $\hat{b}_j^\delta$ bounds the maximum capacity that can be effectively used given a coverage radius δ: $\hat{b}_j^\delta = \min\{\sum_{i\in I_j^\delta} h_i, b_j\}$. Throughout we assume that capacity constraints (3) are expressed in this stronger form. Also, $H_{ag} = \sum_{i\in I} h_i$ will denote the aggregated demand.

The following observations help us to appreciate the close relationship between the above problem and *CpCP*.

Remarks

1. The optimal solution to *CpCP* can be obtained by finding the smallest $k \in K$ such that $H(D^k) \geqslant H_{ag}$. Note that the solutions to PD_δ correspond to binary assignments of customers to open facilities. Thus, if the total demand that can be satisfied within the radius D^k is at least the aggregated demand, all customers are assigned and D^k is a valid upper bound on *CpCP*.
2. If, for a given δ, $H(\delta) < H_{ag}$, then δ gives a valid lower bound on the value of *CpCP*. Moreover, when $\delta = D^k$ for some $k \in K$ such that $H(D^{k-1}) < H_{ag}$, then D^k is also a valid lower bound on the optimal value of *CpCP*, even if $H(D^k) \geqslant H_{ag}$.

The above remarks can be summarized in the following result.

Proposition 1. *The optimal solution to CpCP is given by D^{k^*}, where $k^* \in K$ is such that $H(D^{k^*-1}) < H_{ag} \leqslant H(D^{k^*})$.*

Unfortunately, for a given value of δ, problem PD_δ is not easy to solve since it is an NP-hard problem (notice that it has the generalized assignment problem as a special case). For this reason, we will not try to solve PD_δ exactly. Instead, we will use a relaxation of PD_δ for finding valid lower bounds for *CpCP*. For a given relaxation of PD_δ, let $\overline{H}(\delta)$ denote its value. Note that Remark 1 no longer holds for relaxations of PD_δ, since δ need not be an upper bound on the optimal value of *CpCP* when $\overline{H}(\delta) \geqslant H_{ag}$. The reason is that satisfying the aggregated demand constraint for a relaxation of PD_δ, no longer guarantees that there exists a feasible allocation of all the customers. On the contrary, Remark 2 also holds for the relaxations of PD_δ. For this reason we will obtain a valid lower bound for *CpCP* by solving a relaxation of PD_δ.

Proposition 2. *For $k \in K$, if $\overline{H}(D^{k-1}) < H_{ag}$ then D^k is a valid lower bound on the optimal value to CpCP. The best such bound is given by D^{k^*}, where $k^* \in K$ is such that $\overline{H}(D^{k^*-1}) < H_{ag} \leqslant \overline{H}(D^{k^*})$.*

We next present the relaxation of PD_δ that we have used and we show how to solve it in order to obtain the value $\overline{H}(\delta)$, for a given value of δ.

3.1. Lagrangean relaxation of PD_δ

In this section we consider the relaxation of PD_δ that results when relaxing constraints (2) in a Lagrangean fashion. The subproblem that we obtain for a given multipliers vector $u \in \mathbb{R}_+^{|I|}$ is:

$$\begin{aligned} L1^\delta(u) &= \max \sum_{j\in J}\sum_{i\in I_j^\delta} h_i x_{ij} + \sum_{i\in I} u_i\left(1 - \sum_{j\in J_i^\delta} x_{ij}\right) \\ &= \sum_{i\in I} u_i + \max\left\{\sum_{j\in J}\sum_{i\in I_j^\delta}(h_i - u_i)x_{ij}\right\} \\ &\text{s.t.} \sum_{i\in I_j^\delta} h_i x_{ij} \leqslant \hat{b}_j^\delta y_j \quad \forall j \in J, \\ &\sum_{j\in J} y_j \leqslant p, \\ &x_{ij} \in \{0,1\}, y_j \in \{0,1\} \quad \forall j \in J, i \in I_j^\delta. \end{aligned}$$

For solving $L1^\delta(u)$, we first solve for each $j \in J$ the following knapsack problem:

$$\begin{aligned} KP_j(u) = \max & \sum_{i\in I_j^\delta}(h_i - u_i)x_{ij}, \\ \text{s.t.} & \sum_{i\in I_j^\delta} h_i x_{ij} \leqslant \hat{b}_j^\delta, \\ & x_{ij} \in \{0,1\} \qquad \forall i \in I_j^\delta, \end{aligned}$$

and then we order the indices in J by non increasing values of $KP_j(u)$, so that $KP_{j_1}(u) \geqslant KP_{j_2}(u) \geqslant \cdots \geqslant KP_{j_{|J|}}(u)$. Let $p^* = \min\{p, \max\{r : KP_{j_r}(u) > 0\}\}$. Then, the solution to $L1^\delta(u)$ consists of opening the plants $j_1, j_2, \ldots, j_{p^*}$, and assigning to each of them the customers given by the optimal solution to its corresponding knapsack problem. Note that, although both PD_δ and $KP_j(u)$ are NP-hard problems, pseudo-polynomial algorithms exist that allow to solve knapsack problems very quickly so that $L1^\delta(u)$ can be efficiently computed.

Thus, for a given vector of multipliers u, we can apply the above procedure to obtain the value $L1^\delta(u)$, that is a valid upper bound on the value of PD_δ. As usual, the best upper bound is obtained by solving the Lagrangean dual. Therefore,

$$(D1) \quad \overline{H}(\delta) = \min_{u\in\mathbb{R}_+^{|I|}} L1^\delta(u).$$

For a given vector u, a subgradient of $L1^\delta(u)$ is given by $\gamma = \mathbf{1} - x(u)$, where $\mathbf{1}$ is a vector whose components are all one, and $x(u)$ is the solution to subproblem $L1^\delta(u)$. Hence, $D1$ can be solved by applying subgradient optimization.

3.2. Heuristic solutions to *CpCP* from $L1^\delta(u)$

At the inner iterations of subgradient optimization, we apply a very simple heuristic for obtaining feasible solutions to *CpCP*. For a given vector of multipliers u, let $x(u)$, and $y(u)$ denote the optimal solution to $L1^\delta(u)$. Throughout the heuristic, the set of open facilities is given by $J(u) = \{j \in J : y_j(u) = 1\}$ and the facility to which customer $i \in I$ is assigned is denoted $j(i)$. The heuristic has two steps, that are the following:

1. First, all customers $i \in I$, such that there exists a facility $j^* \in J(u)$ with $x_{ij^*}(u) = 1$, are assigned to that facility: i.e., $j(i) = j^*$ (ties are broken by the minimum assignment distance).
2. All customers that are not yet assigned are ordered by decreasing values of their demands. Then, each unassigned customer is assigned to the open facility with more available capacity, if any. That is, for each unassigned customer, let $j(i) \in \arg\max\{s_j : j \in J(u) \cap J_i^\delta, s_j \geqslant h_i\}$ where $s_j = b_j - \sum_{i\in A_j} h_i$ denotes the available capacity of facility j (A_j is the set of customers already assigned to facility j).

Obviously, the above heuristic may fail to obtain a feasible solution. At the iterations when the heuristic succeeds, we compare the

obtained solution with the best solution obtained so far, and we record it if it improves its objective function value.

4. The Minimum Required Centers Within Fixed Radius Problem

The problem that we present next is also closely related to *CpCP*. It consists of finding the minimum number of centers that are needed to satisfy the customers demands within a fixed radius δ. We refer to that problem as the Minimum Required Centers Within Fixed Radius Problem. Using the same decision variables as before, the problem can be modeled as:

$$(PC_\delta) \quad Y(\delta) = \min \quad \sum_{j\in J} y_j, \tag{6}$$

$$\text{s.t.} \quad \sum_{j\in J_i^\delta} x_{ij} = 1 \quad \forall i \in I, \tag{7}$$

$$\sum_{i\in I_j^\delta} h_i x_{ij} \leqslant \hat{b}_j^\delta \quad \forall j \in J, \tag{8}$$

$$x_{ij} \leqslant y_j \quad \forall j \in J, i \in I_j^\delta, \tag{9}$$

$$\sum_{j\in J} \hat{b}_j^\delta y_j \geqslant H_{ag}, \tag{10}$$

$$x_{ij} \in \{0,1\}, \quad y_j \in \{0,1\} \quad \forall j \in J,\ i \in I_j^\delta. \tag{11}$$

Here, the new constraints (9) ensure that customers are only assigned to open facilities. Using these constraints, we can rewrite the capacity constraints as (8). Additionally, we include the constraint (10) to ensure that the capacity of the open facilities is enough to satisfy the aggregated demand. This aggregated demand constraint is redundant in PC_δ, but we include it in its formulation in order to strengthen the relaxation that we will consider later in this section. A problem similar to PC_δ, but with the non-reinforced expression of the capacity constraints (8) and without constraint (10), has been considered in the work of Özsoy and Pinar [13] for solving *CpCP*. Note that when for a given δ, $Y(\delta) > p$, then δ is a lower bound on the optimal value of *CpCP*. Thus, similarly to Proposition 1 in Section 3, the optimal solution to *CpCP* is given by D^{k^*}, where $k^* \in K$ is such that $Y(D^{k^*-1}) > p \geqslant Y(D^{k^*})$. Again, problems PC_δ are NP-hard so we will resort to solving relaxations of PC_δ in order to obtain valid lower bounds. Let, $\overline{Y}(\delta)$ denote the optimal value of the relaxed problem for a fixed value of δ.

Proposition 3. *For $k \in K$, if $\overline{Y}(D^{k-1}) > p$ then D^k is a valid lower bound on the optimal value to CpCP. The best such bound is given by D^{k^*}, where $k^* \in K$ is such that $\overline{Y}(D^{k^*-1}) > p \geqslant \overline{Y}(D^{k^*})$.*

We next present the relaxation of PC_δ that we have used and we show how to solve it in order to obtain the value $\overline{Y}(\delta)$, for a given value of δ.

4.1. Lagrangean relaxation of PC_δ

When relaxing constraints (7) in a Lagrangean fashion the subproblem that we obtain for a given multipliers vector $u \in \mathbb{R}^{|I|}$ is:

$$L2^\delta(u) = \min \quad \sum_{j\in J} y_j + \sum_{i\in I} u_i \left(1 - \sum_{j\in J_i^\delta} x_{ij}\right)$$

$$= \sum_{i\in I} u_i + \min \left\{ \sum_{j\in J} \left(y_j - \sum_{i\in I_j^\delta} u_i x_{ij} \right) \right\}$$

$$\text{s.t.} \quad \sum_{i\in I_j^\delta} h_i x_{ij} \leqslant \hat{b}_j^\delta \quad \forall j \in J,$$

$$x_{ij} \leqslant y_j \quad \forall j \in J, i \in I_j^\delta,$$

$$\sum_{j\in J} \hat{b}_j^\delta y_j \geqslant H_{ag},$$

$$x_{ij} \in \{0,1\}, y_j \in \{0,1\} \quad \forall j \in J,\ i \in I_j^\delta.$$

For each center $j \in J$, consider the subproblem

$$KP_j(u) = \max \quad \sum_{i\in I_j^\delta} u_i x_{ij}$$

$$\text{s.t.} \quad \sum_{i\in I_j^\delta} h_i x_{ij} \leqslant \hat{b}_j^\delta,$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I_j^\delta.$$

The index set of centers to be open in the optimal solution to $L2^\delta(u)$, denoted $J(u) \subseteq J$, can be found by solving

$$AG(u) = \min \quad \sum_{j\in J} (1 - KP_j(u)) y_j$$

$$\text{s.t.} \quad \sum_{j\in J} \hat{b}_j^\delta y_j \geqslant H_{ag},$$

$$y_j \in \{0,1\} \quad \forall j \in J.$$

In particular, if $y(u)$ denotes the optimal solution to $AG(u)$, then $J(u) = \{j \in J : y_j(u) = 1\}$. Thus, the optimal solution to $L2^\delta(u)$ consists of opening the centers indexed in $J(u)$ and for these centers performing the assignment of customers given by the optimal solution to subproblem $KP_j(u)$. Note that the knapsack problems $KP_j(u)$ have only $|J|$ variables, and that they can be solved in pseudo-polynomial time. Let $x(u)$ denote the resulting assignment vector.

Again, we use subgradient optimization to solve the Lagrangean dual

$$(D2) \quad \overline{Y}(\delta) = \max_{u\in\mathbb{R}^{|I|}} L2^\delta(u).$$

For a given vector u, the components of a subgradient vector $\gamma \in \mathbb{R}^{|I|}$ of $L2^\delta(u)$ are given by $\gamma_i = 1 - \sum_{j\in J_i^\delta} x_{ij}(u)$.

4.2. Heuristic solutions to CpCP from $L2^\delta(u)$

At the inner iterations of subgradient optimization we apply a heuristic for obtaining feasible solutions to *CpCP*. The idea of the heuristic is opening enough plants so as to assign all the customers within the coverage radius δ. Note that for a given set of plants $J(u)$ there might be customers that do not fall within the coverage radius δ of any of the plants in $J(u)$. We will denote $I_{nc} = \{i \in I : d_{ij} > \delta, \forall j \in J(u)\} = I \setminus (\bigcup_{j\in J(u)} I_j^\delta)$, the set of such "uncovered" customers. Let JO denote the set of plants opened by the heuristic (initially $JO = \emptyset$). The heuristic consists of three steps which are the following:

1. Let $J1 = \{j \in J \setminus J(u) : I_{nc} \cap I_j^\delta \neq \emptyset\}$ be the set of plants that cover at least one customer in I_{nc} within the coverage radius δ. When $I_{nc} \neq \emptyset$, in order to obtain a feasible solution to *CpCP*, some plants in $J1$ must open. For this reason, we successively open plants in $J1$, until all the customers in I_{nc} are assigned to some open plant within the coverage radius δ. The criterion that is used to select plants to open is to maximize the demand that is satisfied. Recall that if plant j is opened an estimation of the demand that it will satisfy is $s_j = \min\{\hat{b}_j^\delta, \sum_{i\in I_j^\delta \cap I_{nc}} h_i\}$, and corresponds to the total demand of the customers that will be assigned to it if opened. Let A_j denote the set of such customers. A summary of Step 1 is depicted in Algorithm 1.

Algorithm 1. Step 1.

$I_a = \emptyset$
while $I_{nc} \neq \emptyset$ **do**
  Select the unopened plant in $J1$ with the most available capacity
  $j^* \in \text{argmax}\{s_j : j \in J1\}$
  $JO := JO \cup \{j^*\}$
  $J1 := J1 \setminus \{j^*\}$
  $j(i) = j^*, \forall i \in A_{j^*}$
  $I_{nc} := I_{nc} \setminus A_{j^*}$
  $I_a := I_a \cup A_{j^*}$
end while

2. When the second step is entered, only the set of customers that are reachable from plants in $J(u)$ remains unassigned. Let $I_{na} = I \setminus I_a$ be this set. Now, we order customers in I_{na} by decreasing values of their demands and try to assign them to some open plant within its coverage radius δ, with enough available residual capacity. If no such plant exists, since all customers in I_{na} fall within the coverage radius δ of at least one plant in $J(u)$, we open one such plant in $J(u)$ and assign the customer to it. In both cases, the selected plant is the one (in the corresponding set of candidate plants) with the most residual capacity $s_j = b_j^\delta - \sum_{i:j(i)=j} h_i$. A summary of Step 2 is depicted in Algorithm 2.

Algorithm 2. Step 2.

$I_{na} = I \setminus I_a$
for $i \in I_{na}$ **do**
  Define set of candidate plants
  if $JO \cap \{j \in J_i^\delta : h_i \leqslant s_j\} \neq \emptyset$ **then**
    $J2 = JO \cap \{j \in J_i^\delta : h_i \leqslant s_j\}$
  else
    $J2 = \{j \in J(u) : h_i \leqslant s_j\}$
  end if
  if $J2 \neq \emptyset$ **then**
    Select the candidate plant with more available capacity
    $j^* \in \text{argmax}\{s_j : j \in J2\}$
    if $j^* \in J(u)$ **then**
      $J(u) := J(u) \setminus \{j^*\}$
      $JO := JO \cup \{j^*\}$
    end if
    $j(i) := j^*$
    $I_{na} := I_{na} \setminus \{i\}$
  end if
end for

3. If there are still customers that have not been assigned, new plants are opened until all unassigned customers are assigned to one open plant. Note that at this point the only unopened plants with enough capacity that could cover some unassigned customer within the radius δ are the ones in $J \setminus (JO \cup J(u))$. Again, we consider unassigned customers by decreasing values of their demand and the criterion that is used to select the plants is the available capacity.

5. Lower and upper bounds

We next obtain two lower bounds, $LB1 = D^{k_1^*}$ and $LB2 = D^{k_2^*}$, that satisfy the conditions of Propositions 2 and 3, respectively. Both radii, $D^{k_1^*}$ and $D^{k_2^*}$, are obtained by applying binary search on the value of the index k such that the radius D^k gives a valid lower bound. The procedure to obtain $D^{k_1^*}$ solves a series of Lagrangean duals of the type $\overline{H}(D^k)$, whereas the procedure to obtain $D^{k_2^*}$ resorts to the solution of Lagrangean duals of the type $\overline{Y}(D^k)$. When solving the corresponding Lagrangean duals, both procedures may also generate valid upper bounds, $ub1(D^k)$ and $ub2(D^k)$, by applying the heuristics as explained in Sections 3.2 and 4.2, respectively. More specifically, the values of the bounds are the objective function value in *CpCP* of the corresponding feasible solutions. Let $UB1$ and $UB2$ denote the best such bounds. The two procedures follow a very similar structure, which is summarized in Algorithm 3:

Algorithm 3. Identification of upper and lower bounds

Computation of $LB1$ and $UB1$.	Computation of $LB2$ and $UB2$.
$a = 1, b = k_{max}, UB1 = D^{k_{max}}$	$a = 1, b = k_{max}, UB2 = D^{k_{max}}$
while $a \neq b$ **do**	**while** $a \neq b$ **do**
$k := \lfloor \frac{a+b}{2} \rfloor$	$k := \lfloor \frac{a+b}{2} \rfloor$
Evaluate $\overline{H}(D^k)$	**Evaluate** $\overline{Y}(D^k)$
if $ub1(D^k) < UB1$ **then**	**if** $ub2(D^k) < UB2$ **then**
$UB1 := ub1(D^k)$	$UB2 := ub2(D^k)$
$b := k$	$b := k$
end if	**end if**
if $\overline{H}(D^k) < H_{ag}$ **then**	**if** $\overline{Y}(D^k) > p$ **then**
$a := k + 1$	$a := k + 1$
else	**else**
$b := k$	$b := k$
end if	**end if**
end while	**end while**
$k_1^* = a$	$k_2^* = a$
$LB1 := D^{k_1^*}$	$LB2 := D^{k_1^*}$

5.1. Empirical comparison of lower bounds

The two bounds presented in the last subsection correspond to Lagrangean relaxations of two different problems related with the *CpCP* that cannot be compared from a theoretical point of view. For this reason, we have carried out a series of computational experiences to empirically compare these bounds. We next describe these computational experiments and evaluate the quality of the obtained bounds. All the algorithms have been coded in C and run on a HP Pavilion ze5400 PC Intel Pentium 4, at 2.39 GHz and 512 MB of RAM. CPLEX 10.1 was used to solve subproblems $\overline{Y}(\delta)$ while the knapsack subproblems of the Lagrangean functions where solved using the algorithm presented in [11]. In all the experiments the cpu time limit has been set to one hour (3600 seconds). The 174 benchmark instances that we have used have $I = J$; they are the following:

1. The set with 160 instances with euclidean distances of [3]. The set contains 10 instances of each of the following dimensions: (i) $n = 50$ and $p = 5, 12, 16, 20$; (ii) $n = 100$ and $p = 10, 25, 33, 40$; (iii) $n = 150$ and $p = 15, 27, 50, 60$, and; (iv) $n = 200$ and $p = 20, 50, 66, 80$. These instances are referred to as set S1. set S1 contains the 20 instances generated by Osman and Christofides [12] for the capacitated p median problem (the instances with $n = 50$ and $p = 5$, and the instances with $n = 100$ and $p = 10$). These 20 instances are available in the OR-Library (http://mscmga.ms.ic.ac.uk/info.html). Customers demands in this set range in the interval $[1, 20]$. Facilities share the same capacity, whose value is chosen so that the total capacity of the open facilities is about 1.2 times the aggregated demand.

Please cite this article in press as: M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated p-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022

2. The set of eight instances derived in [14] for *CpCP* from the two instances with non-euclidean distances generated for the maximum covering location problem by Galvaõ and ReVelle [7]. There are two instances of each of the following dimensions: (100,5), (100,10), (150,10) and (150,15). These instances are referred to as set S2. Demands in the instances with 100 customers range in the interval $[20, 60]$, whereas in instances with 150 customers they range in $[40, 130]$. According to the facility capacities, there are two types of instances, one with homogeneous capacities, and one where capacities take three different values (small, average, and large). In all cases, the ratio of the aggregated demand over p is above 90% of the average capacity.
3. The set of six instances with euclidean distances of [10] that correspond to real data from the Brazilian city of São José dos Campos. These instances are available from http://www.la-c.inpe.br/lorena/instancias.html. Their dimensions (n, p) are (100,10), (200,15), (300,25), (300,30), (402,30) and (402,40), respectively. These instances are referred to as set S3. Here, demands range in the interval $[1, 600]$, although their average is between 30 and 60. All facilities share the same capacity b, and the ratio $H/(pb)$ ranges in $[45\%, 90\%]$.

The 20 instances generated by Osman and Christofides in S1, as well as all the instances in S2 and S3 have also been used in [14,13] for *CpCP*. To the best of our knowledge the remaining 140 instances in S1 have not been used for *CpCP*, although they have been used for the p-median problem in [3].

The numerical results with sets S1, S2, S3 are depicted in Tables 1–3 respectively. In these tables we evaluate our lower bounds in terms of quality, by comparing them with the value of the optimal/best-known solutions, and also in terms of the cpu times required to obtain them. Our lower bounds are also compared with those of Phases I and II of the algorithm of Özsoy and Pinar [13]. The bound of Phase I is obtained from the solution to the LP relaxation of problems PC_δ, whereas bound of Phase II, is obtained from the solution to the MILPs that result when relaxing the integrality constraints of the assignment variables, but maintaining integrality requirements on the location variables. Since these results were not available for most of the instances in S1 we report here the results that have been obtained with our implementation of the method of [13]. In addition, this allows a better comparison of the required cpu times. For the instances that were used in [13] the results that we have obtained do not fully coincide with the ones reported in [13]. We believe that this is due to technical differences in the updating of the search parameters.

In Table 1 we give the average results for each group of 10 instances with the same value of n and p, whereas in Tables 2 and 3 we give the results corresponding to each of the instances in Sets S2 and S3, respectively. In all the tables, columns under the headings n and p give, for each instance or group of instances, the number of customers and the value of the parameter p, respectively. Columns under the heading *% gap* give the percent deviation with respect to the optimal/best-known solution (defined as $100\frac{Opt-LB}{Opt}\%$), and columns under the heading *CPU time* give the required cpu-times in seconds. The results corresponding to the method of Özsoy and Pinar are depicted under the headings *OP*. In particular, the results corresponding to Phases 1 and 2 are given in columns under *Ph1* and *Ph2*, respectively. The results of our solution algorithms are given under the heading *ADF*. Now the results of the Lagrangean duals *D1* and *D2* are given in the columns under *D1* and *D2*, respectively. Since the optimal values of all the instances in set S1 are not known, in Table 1 column under *#opt.* depicts the number of instances of each group of 10 instances with the same parameter values for which the optimal solution is known. Also, in Table 1 Columns 4–7, under the heading *#LB = optimal*, give the number of instances of each group for which the value of the lower bound coincides with the optimal value, for each of the compared methods. In Tables 2 and 3 columns under *Opt* give the value of the optimal solution, whereas columns under *Lower Bound* give the values of the lower bounds obtained with each of the considered methods.

The results in Tables 1–3 indicate that the lower bounds obtained with the two Lagrangean duals are very good and the percent deviation with respect to the optimal/best-known solution is very small. This can be appreciated, in particular, for the instances in the set S1 where, out of the 104 instances for which the optimal solution is known, the bounds given by *D1* and *D2* were already optimal for 73 and 76 instances, respectively. These results are indeed better than those of *OP* where the number of instances for which the lower bound gives the optimal value are 15 and 57 for *Ph1* and *Ph2*, respectively. In terms of the percent average gaps, again *D2* gives the best results followed by *D1* and *Ph2*. If we consider again the subset of 104 instances for which the optimal solution is known, the distribution of the percent deviation

Table 1
Results with instances S1.

n	p	#Opt.	#LB = optimal				%gap				CPU time			
			OP		ADF		OP		ADF		OP		ADF	
			Ph1	Ph2	D1	D2	Ph1	Ph2	D1	D2	Ph1	Ph2	D1	D2
50	5	10	2	9	9	9	5.86	0.36	0.32	0.32	0.92	1.60	0.48	0.57
50	12	10	0	5	7	7	19.71	3.40	1.93	1.93	1.01	5.90	0.43	0.97
50	16	10	0	5	5	5	24.62	3.67	3.76	2.89	1.10	8.15	0.43	0.74
50	20	9	0	1	4	6	40.00	16.47	5.23	1.69	0.90	398.39	0.51	0.86
100	10	10	0	10	7	6	8.72	0.00	2.40	2.40	8.98	19.80	2.63	2.57
100	25	5	0	4	4	4	24.91	7.32	8.03	6.69	8.43	1578.04	2.12	2.81
100	33	2	0	1	1	1	26.16	14.08	9.44	7.96	11.01	2497.66	1.77	2.37
100	40	5	0	0	2	2	42.68	24.97	11.43	9.17	10.10	3256.56	2.10	2.80
150	15	9	2	9	7	7	5.50	0.59	1.73	1.73	40.73	74.46	6.55	39.64
150	37	5	0	3	6	7	21.05	10.83	6.83	5.83	54.08	2300.27	4.83	7.30
150	50	4	0	1	3	3	28.62	21.51	11.79	9.48	51.49	3600.00	6.17	8.34
150	60	5	0	0	3	4	38.55	34.66	10.73	8.08	50.45	3600.00	5.89	7.35
200	20	9	1	9	6	6	10.04	1.76	3.81	3.81	144.28	347.16	12.54	17.27
200	50	3	0	0	1	2	26.15	24.33	15.24	14.13	154.80	3600.00	11.79	17.63
200	66	4	0	0	4	4	30.77	30.77	7.26	7.26	160.40	3600.00	13.32	12.39
200	80	4	0	0	4	4	39.87	39.87	11.30	12.36	254.96	3600.00	13.22	15.94

Please cite this article in press as: M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated p-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022

Table 2
Results with instances S2.

	n	p	Opt	Lower bound				% ap				CPU time			
				OP		ADF		OP		ADF		OP		ADF	
				Ph1	Ph2	D1	D2	Ph1	Ph2	D1	D2	Ph1	Ph2	D1	D2
G1	100	5	94	93	94	93	93	1.06	0.00	1.06	1.06	18.70	31.10	7.10	90.04
G2	100	5	94	93	94	93	93	1.06	0.00	1.06	1.06	5.94	31.72	6.57	63.15
G3	100	10	83	58	83	59	59	30.12	0.00	28.92	28.92	3.69	190.47	3.21	3.56
G4	100	10	84	58	84	81	81	30.95	0.00	3.57	3.57	2.45	197.03	4.93	3.58
G5	150	10	95	93	95	93	93	2.11	0.00	2.11	2.11	14.23	335.28	19.00	42.76
G6	150	10	96	93	96	94	94	3.13	0.00	2.08	2.08	17.88	914.58	19.18	46.91
G7	150	15	89	86	88	86	86	3.37	1.12	3.37	3.37	10.62	396.92	9.22	8.96
G8	150	15	89	86	89	88	88	3.37	0.00	1.12	1.12	14.09	676.55	18.36	10.26

Table 3
Results with instances S3.

	n	p	Opt	Lower bound				% gap				CPU time			
				OP		ADF		OP		ADF		OP		ADF	
				Ph1	Ph2	D1	D2	Ph1	Ph2	D1	D2	Ph1	Ph2	D1	D2
1	100	10	364.73	316.12	349.67	350.04	350.04	13.33	4.13	5.28	4.03	14.44	523.05	12.02	92.08
2	200	15	304.14	301.01	303.98	302.83	302.83	1.03	0.05	0.43	0.43	122.32	538.58	60.98	35.24
3a	300	25	278.96	275.07	277.82	275.18	275.18	1.39	0.41	1.37	1.35	798.51	3600.00	114.37	85.21
3b	300	30	253.71	245.12	252.53	249.10	246.42	3.38	0.46	1.85	2.87	806.39	1781.19	113.67	88.43
4a	402	30	284.68	276.00	276.25	277.03	277.03	3.05	2.96	2.76	2.69	2348.22	3600.00	334.56	248.48
4b	402	40	239.38	237.00	237.03	237.32	237.32	0.99	0.98	0.87	0.86	2345.93	3600.00	242.97	284.18

from the optimal value for *D2* is the following: do not exceed 5% for 83 instances; in $(5, 10]$ for 18 instances; and over 10% for 3 instances. The maximum percent deviation is 11.11%. For the same subset of instances, the distribution of the percent deviation from the optimal value for *Ph2* is as follows: do not exceed 5% for 59 instances; in $(5, 10]$ for 12 instances and over 10% for 33 instances. The maximum deviation is 37.5%. As can be seen, the cpu times required to obtain these bounds are small both for *D1* and *D2*, taking into account the size and the difficulty of the problems. In general, *D1* required somewhat smaller times than *D2*, although this difference tends to disappear as the size of the instances increases. These results indicate that, in general, *D2* is more efficient than *D1*, since it gives slightly better lower bounds in similar cpu times. The quality of the bounds obtained with *Ph2* is rather good, although in general the bound *D2* (and also *D1*) is better. In addition, it is worth noting the big computational effort required to obtain the bounds *Ph2*, since they are obtained by solving exactly a series of mixed integer problems. This big computational burden is reflected in the cpu times required by *Ph2* and also in the quality of the bounds that deteriorates notably as the size of the instances increases, mainly due to the fact that the method is not able to terminate in the allowed cpu time.

Neither *D1* nor *D2* were able to give the optimal value of any of the eight instances in the set S2 although, excepting in one pathological instance, the percent deviation from the optimal solution was smaller than 4% both for *D1* and *D2*, which gave the same lower bounds for all the instances. Now the cpu time requirements of *D1* (which takes between 3 and 20 seconds) are smaller than those of *D2*, which takes 90 and 63 seconds, respectively, for the two smaller instances in the set, although it takes less than one minute for the remaining larger instances. For this set of instances *Ph2* gave the best results, although again the computational requirements in terms of time increase considerably with the sizes of the instances and are between three and fifteen minutes for all the instances excepting the two smaller ones.

For the instances in the set S3, *D1* and *D2* gave the same results, excepting for instance 3b, for which *D1* gave a slightly better bound. Both *D1* and *D2* beat *Ph2* in three of the six instances. In this data set it becomes again evident that the cpu time requirements of both *D1* and *D2* are very small as compared with *Ph2*, which reaches the one hour limit in three of the six instances. On the contrary, the cpu time requirements of *D1* and *D2* did not exceed 6 and 5 minutes, respectively, for any of the instances in this set. Thus, the results in Tables 1–3 allow us to conclude that both *D1* and *D2* give very good quality lower bounds in small computational times. These bounds are indeed much better than the ones of *Ph1* and require smaller computation times. Moreover, in general, the bounds obtained with *D1* and *D2* also beat those of *Ph2*, which is very costly in terms of cpu time and whose time requirements increase notably with the sizes of the instances. One additional advantage in solving *D1* and *D2* is that the heuristic that is applied also gives feasible solutions and thus valid upper bounds, which gives us a reference for evaluating the quality of the lower bounds. Note that both *Ph1* and *Ph2* terminate without a valid upper bound, unless optimality is proven.

In our opinion, the increasing difficulty of instances in S1, S2 and S3 is due not only to the increasing sizes of the instances but also to other characteristics of the instances such as, for instance, the range between the largest and the smallest radii. Thus, given that, for the considered sets of instances, the distribution of the number of radii in the interval of their range is not uniform, we believe that an indicator of the quality of the lower bounds better than the percent gap is the reduction on the number of radii that can be attained after applying Elimination Tests 1 and 2. Recall from Section 2 that Elimination Test 1 can be applied when a lower bound is at hand, but Elimination Test 2 requires an upper bound. For better appreciating the above comments, Fig. 1 depicts the percent reduction on the number of radii due to the elimination tests after computing the bounds using PC_δ for all the instances. Again the values in set S1 are average values over all the instances with

Please cite this article in press as: M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated *p*-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022

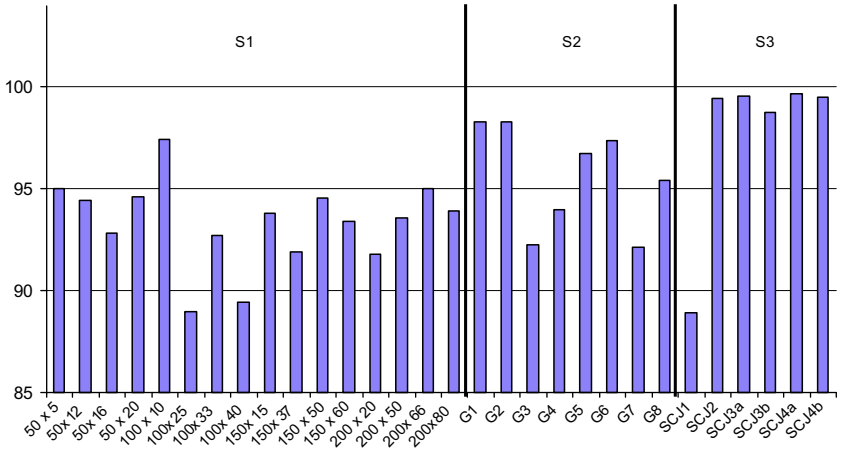

Fig. 1.

the same values of n and p. As can be seen, these reductions are indeed very significant, since the depicted values are always over 88%. In fact, if we do not take average values, all the percent reductions are over 80%, excepting for one small (50×5) instance in set S1 for which the percent reduction was 74.11. We believe that this information gives a richer picture of the effectiveness of the bounds than the resulting percent gaps, since, for a given *CpCP* instance, a constant increase of all the distances by the same term would affect the gap dramatically $\left(\frac{(Opt+K)-(LB+K)}{(Opt+K)} < \frac{Opt-LB}{Opt}\right)$ while the combinatorial structure of the problem would remain the same.

6. The exact solution to *CpCP*

As mentioned before, once good quality bounds have been obtained they can be used for solving exactly Problem *CpCP*. Given that, as we have seen, the best lower bounds are based on Problem PC_δ, our exact algorithm is also based on the solution to this problem. Let us recall that the optimal value to *CpCP* is given by the smallest radius δ^* such that the optimal value to PC_{δ^*} does not exceed p. Indeed, for given bounds LB and UB, such δ^* can be found by exploring the interval $[LB, UB]$. There are essentially two different strategies that can be used for exploring the set of candidate radii in $[LB, UB]$: the first one initially sets $\delta = LB$ and sequentially increases its value to the next radius, whereas the second one applies binary search. Although both strategies solve a sequence of problems PC_δ with different values of δ until optimality is proven, they differ in the way the sequence is defined. Note that by using different strategies, the sequence of problems that are solved will be different. Moreover, the number of problems solved will also vary from one strategy to another. Indeed, if n_{rad} denotes the number of radii in the initial interval $[LB, UB]$, with binary search the maximum number of problems to solve is $\mathcal{O}(\log_2(n_{rad}))$, whereas with sequential search this number may reach n_{rad}. In practice, the actual number of iterations with binary search will always be very close to its upper limit, whereas with sequential search this number will tend to be smaller, as the quality of the lower bound increases. In addition, it is worth to point out that the performance of binary search depends not only on the quality of the lower bound, as it happens with sequential search, but also on the quality of the upper bound. Thus, even if the lower bound is very tight, a high number of iterations will be needed when the upper bound is not good.

Taking into account the above considerations, in our computational experiments we have compared empirically both strategies. Given that the obtained results are slightly better for the binary search, these are the results that are fully reported here, although a brief summary of the comparison of both strategies is given later in this section. First, a summary of the binary search algorithm that we have used is given in Algorithm 4. As before, $Y(\delta)$ denotes the value of the optimal solution to PC_δ. Let k_l and k_u be such that $LB = D^{k_l}$ and $UB = D^{k_u}$.

Algorithm 4. Exact solution to *CpCP*.

$a = k_l, b = k_u$
while $(a \neq b)$ **do**
  $k := \lfloor \frac{a+b}{2} \rfloor$
  if $Y(D^k) > p$ **then**
    $a := k + 1$
  else
    $b := k$
  end if
end while
$\delta^* = D^a$

The results obtained with the binary search on the different sets of instances are given in Tables 4–6. As in the previous section, a limit of one hour of cpu time has been set for each instance. The

Table 4
Results of exact algorithm with instances S1.

n	P	% gap_i	$\#rad_i$	% gap_f	$\#rad_f$	#opt	#subp	cpu
50	5	15.47	5.60	0.00	1.00	10	1.20	3.00
50	12	26.50	6.70	2.11	1.40	9	2.00	393.19
50	16	40.05	7.90	0.00	1.00	10	2.30	156.49
50	20	25.17	6.00	2.00	1.40	9	1.80	397.54
100	10	10.85	3.10	0.50	1.10	9	1.20	662.03
100	25	93.97	13.40	13.09	2.70	4	2.10	2203.86
100	33	65.96	8.80	6.91	1.80	2	1.90	3096.17
100	40	95.97	12.80	10.07	2.20	4	2.40	2295.28
150	15	44.24	8.00	1.21	1.20	8	2.10	753.69
150	37	85.21	10.30	7.15	1.80	6	2.00	1584.79
150	50	59.84	6.90	9.84	2.00	4	1.50	2381.15
150	60	71.74	8.40	8.35	1.90	5	1.80	2126.33
200	20	68.85	10.60	2.86	1.40	8	2.30	926.69
200	50	80.44	8.20	16.44	2.50	3	1.50	2760.52
200	66	65.69	6.40	8.33	1.70	4	1.60	2365.37
200	80	75.24	7.40	17.93	2.60	4	1.70	2223.31

Please cite this article in press as: M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated p-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022

Table 5
Results of exact algorithm with instances S2.

	n	P	opt	$\%\ gap_i$	$\#rad_i$	$\%\ gap_f$	$\#rad_f$	$\#subp$	cpu
G1	100	5	94	1.08	2	0.00	1	1	289.18
G2	100	5	94	1.08	2	0.00	1	1	91.97
G3	100	10	83	47.46	9	0.00	1	3	240.73
G4	100	10	84	7.41	7	0.00	1	3	227.71
G5	150	10	95	4.30	5	0.00	1	2	556.06
G6	150	10	96	3.19	4	0.00	1	2	786.87
G7	150	15	89	12.79	12	0.00	1	3	464.94
G8	150	15	89	4.55	7	0.00	1	2	1324.37

Table 6
Results of exact algorithm with instances S3.

	n	P	opt	$\%\ gap_i$	$\#rad_i$	$\%\ gap_f$	$\#rad_f$	$\#subp$	cpu
1	100	10	364.73	41.45	544	11.55	273	1	3600.00
2	200	15	304.14	3.83	110	0.00	1	6	455.37
3a	300	25	278.96	5.39	186	0.00	1	8	837.79
3b	300	30	253.71	17.69	526	0.00	1	9	395.46
4a	402	30	284.68	5.04	262	2.74	6	6	3600.00
4b	402	40	239.38	9.06	358	0.00	1	9	1168.72

entries in Table 4 provide average values for groups of ten instances with the same values of n and p. Under heading $\%gap_i$ we give the percent gap between the initial lower and upper bounds ($100\frac{UB-LB}{LB}\%$). Next column, $\#rad_i$, provides the average number of radii in the interval $[LB, UB]$. Column under $\%gap_f$ depicts the percent gap between the final upper and lower bounds. Column under $\#rad_f$ provides the average number of radii between these final bounds, and column $\#opt$ gives the number of instances of each group that were optimally solved within the cpu time limit. The remaining columns provide information about the efficiency of the algorithm. Under $\#subp$, we report the average number of problems PC_δ that were solved either until optimality was proven or the time limit was reached, and under heading cpu we give the average cpu time in seconds, including the time used to obtain the upper and lower bounds. Tables 5 and 6 display the same information for each of the instances in sets S2 and S3, respectively. Now we have omitted columns $\#opt$ since they do not give any additional information, since each row corresponds to one single instance. Instead, the optimal value of each instance is given under opt.

In our opinion, the results are very satisfactory. In terms of the number of instances that were optimally solved within the cpu time limit of one hour, we were able to solve 99 instances in set S1, the eight instances in set S2 and four of the six instances in set S3, whereas with the algorithm proposed by Özsoy and Pinar, we could only solve 71 instances in set S1, the eight instances in set S2 and one single instance of set S3 within the same time limit. Moreover, except for the first instance in set S3, the binary search allowed to notably reduce the list of candidate solution values even for the instances that could not be solved exactly. It is also worth noting that for the instances that were not already solved within the bounding phase, the overall time is often one order of magnitude larger than the bounding time. Taking into account that, roughly speaking, the bounding phase reduces the number of candidate radii by over 85% if binary search were applied directly to the initial set of candidate radii, about three extra PC_δ subproblems would need to be solved. As compared to the time requirements displayed in Tables 4–6, this would imply a time increase of over 100% for instances in S1 and S2, and of approximately 30% for instances in S3.

In Fig. 2 we compare the efficiency of our algorithm with that of Özsoy and Pinar.

The instances that could be solved by at least one of the algorithms within the cpu time limit of one hour have been classified according to the times taken to solve them. We distinguish the instances that could be solved by only one of the algorithms within the cpu time limit, the ones where one algorithm was at least 10 times faster than the other, the ones where the time taken by one algorithm was between 10% and 80% of the time taken by the other, and the rest, where we consider that both algorithms are similar. In the figure labels, t_{OP} refers to the time taken by the algorithm in [13], whereas t_{ADF} refers to the time taken by our algorithm. This figure allows to appreciate the superiority of our algorithm. Actually, this superiority becomes more significant as the difficulty of the instances grows. Indeed, most of the larger instances that are represented in the figure are accounted in the bars corresponding with the instances where our algorithm was faster.

From the tables it can be appreciated that, although the size of the instances has a great impact on the computational effort required to solve them, there are other factors that are more relevant. For example, in all three sets there are instances with $n = 100$ customers and $p = 10$ centers to locate, but the computational burden required to solve them varies extremely among the sets; while the ones in set S2 were solved in less than five minutes, those in set S1

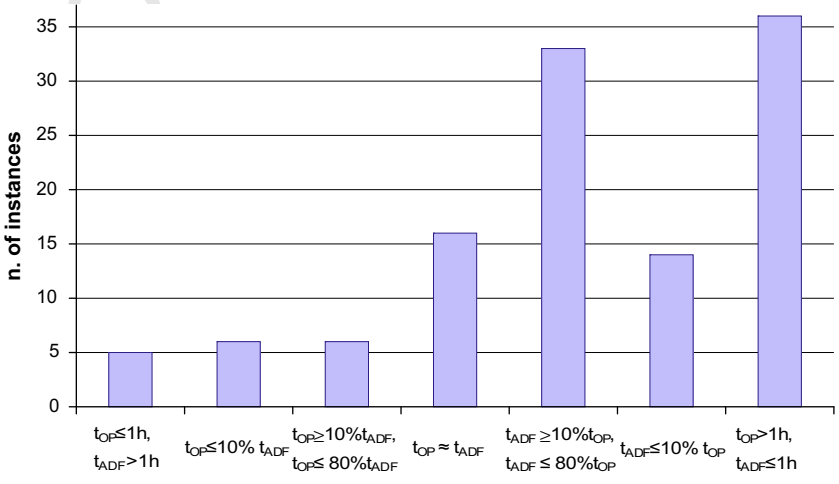

Fig. 2.

Please cite this article in press as: M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated p-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022

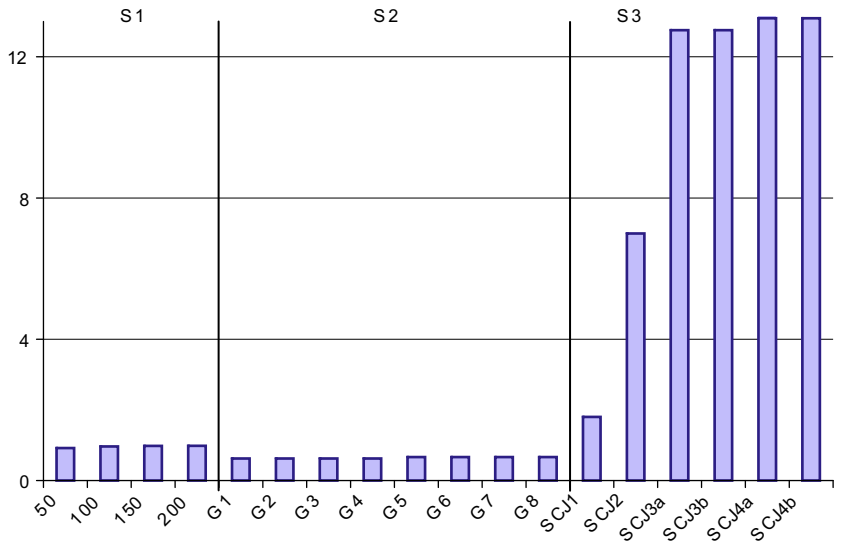

Fig. 3.

required 11 minutes on the average, and the one in set S3 could not be solved after one hour. Moreover, the percent gaps between the initial lower and upper bounds for those instances does not directly relate to these times, either. The factors that seem to better explain these large differences are two, the average cpu time required to solve problems PC_δ, and the number of radii in the interval defined by the initial bounds. Thus, as conjectured in the previous section, this last factor gives a better measure of the quality of the bounds than the percent gap between them.

According to our experiments, a third factor that has influence on the difficulty of the instances is the distance between the different radii. To illustrate this fact, in Fig. 3 we display the densities $\frac{\#radii}{range}$ of the different instances in each set.

As before, for set S1 the values correspond to averages over all the instances in each size group. As can be seen, the smaller densities, ranging in $[0.624, 0.665]$ correspond to instances in set S2. Let us recall that all the instances in this set were optimally solved both with our method and by the one proposed in [13]. On the other end, instances in set S3 have the largest densities, ranging in $[1.8, 21.635]$. This is precisely the set of instances where we have observed more differences in the behavior of both the compared methods and the search strategies. Actually, in this set of instances the value of the radii are not integer and within two consecutive integer values there are possibly several radii. Finally, note that, as previously stated, our numerical results indicate that radii densities allow to make fair estimations of the meaning of the gap values for *CpCP*.

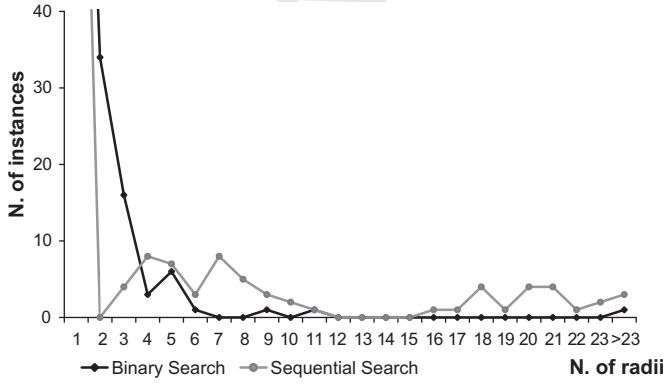

Fig. 4.

We next summarize the results of our comparison between sequential search and binary search for exploring the set of radii in the interval $[LB, UB]$. Since the number of instances that could be solved within the cpu time limit of one hour is almost the same with both strategies, in Fig. 4 we compare the performance of the two search strategies in terms of the number of candidate radii left after termination. Note that this final number of radii will be one only for the instances that could be exactly solved within this cpu time limit, whereas larger values will appear for the instances that needed larger times. In particular, Fig. 4 depicts, for each strategy, the number of instances for which, at termination, the interval $[LB, UB]$ contained exactly k radii, for each integer value k within the range of the corresponding strategy.

Fig. 4 allows to appreciate the superiority of the binary search over the sequential search in terms of the number of candidate solution values left after termination. In fact, for the instances in set S1, the number of radii in $[LB, UB]$ at termination ranges in $[1, 23]$ with sequential search and in $[1, 11]$ with binary search. For the instances from set S3 that could not be solved within the cpu time limit, the number of radii in the final interval $[LB, UB]$ is always much smaller with binary search than with sequential search. Moreover, for the instances in the set S2 (that were all optimally solved with both strategies) binary search also beats sequential search in terms of the required cpu time, since the average cpu time to optimality was 548.73 seconds with sequential search and 497.73 seconds with binary search.

Regarding the effort required to solve subproblems PC_δ, the computational results show that, for a given instance, not all subproblems PC_δ are equally difficult so solve. Our feeling is that, as it happens in other capacitated location problems, instances become harder when capacity constraints are tight in the optimal solution, which commonly happens for δ values near the optimal coverage radius. Therefore, as the value of the radius δ gets closer to the value of the optimal solution, in practice, it becomes really much harder to check how the optimal value of PC_δ compares with p. Thus, when the value δ is very close to the optimal value, for some instances it was impossible to solve completely or to prove infeasibility of just one PC_δ subproblem within one hour of cpu time. This gives an extra advantage of binary search over sequential search, since, typically, less values very close to the optimal value must be explored with the former than with the latter.

Please cite this article in press as: M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated p-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022

7. Conclusions

In this paper we have addressed the *CpCP*, which is an NP-hard problem. We have studied two auxiliary problems, the maximum demand coverage with fixed radius problem and the minimum required centers with fixed radius problem, their relation to *CpCP*, and the lower bounds derived from two associated Lagrangean duals. These bounds are very tight and allow a significant reduction on the number of radii that are candidate for the optimal value of the problem.

In the second part of the paper, we present an exact method for solving *CpCP*, once a lower and an upper bound are at hand, that is based on exploring the set of candidate radii between them. Two different strategies are compared and the results obtained with the best one, binary search, are reported. The obtained numerical results are very satisfactory. Additionally, they allow to identify characteristics of the instances that have a great impact on their difficulty.

Acknowledgements

The authors are grateful to A. Ceselli and G. Righini for providing them with their set of test instances. This work has been partially supported through Grant MTM2006-14961-C05-01 of the Inter-Ministerial Spanish Commission of Science and Technology. The research of the third author has been partially supported by the Departamento de Ingeniería Industrial y Textil, Universidad de las Américas, Puebla, México. These supports are gratefully acknowledged.

References

[1] P. Agarwal, C.M. Procopiuc, Exact and approximation algorithms for clustering, Algorithmica 33 (2002) 201–226.
[2] J. Bar-Ilan, G. Kortsarz, D. Peleg, How to allocate network centers, Journal of Algorithms 15 (1993) 385–415.
[3] A. Ceselli, G. Righini, A branch-and-price algorithm for the capacitated *p*-median problem, Networks 45 (2005) 125–142.
[4] J.A. Díaz, E. Fernández, Hybrid scatter search and path relinking for the capacitated *p*-median problem, European Journal of Operational Research 169 (2006) 570–585.
[5] S. Elloumi, M. Labbé, Y. Pochet, A new formulation and resolution method for the *p*-center problem, INFORMS Journal on Computing 16 (2004) 84–94.
[6] K. Fleszar, K.S. Hindi, An effective VNS for the capacitated *p*-median problem, European Journal of Operational Research 191 (2008) 612–622.
[7] R.D. Galvaõ, C. ReVelle, A Lagrangean heuristic for the maximum covering location problem, European Journal of Operational Research 88 (1996) 114–123.
[8] M. Jaeger, J. Goldberg, A polynomial algorithm for the equal capacity *p*-center problem on trees, Transportation Science 28 (1994) 167–175.
[9] S. Khuller, Y.J. Sussmann, The capacitated *k*-center problem, SIAM Journal on Discrete Mathematics 13 (2000) 403–418.
[10] L.A.N. Lorena, E.L.F. Senne, A column generation approach to capacitated *p*-median problem, Computers and Operations Research 31 (2004) 863–876.
[11] S. Martello, P. Toth, Knapsack Problems, Algorithms and Computer Implementations, Wiley, Chichester, 1990.
[12] I.H. Osman, N. Christofides, Capacitated clustering problems by hybrid simulated annealing and tabu search, International Transactions in Operational Research 1 (1994) 317–336.
[13] F.A. Özsoy, M.Ç. Pinar, An exact algorithm for the capacitated vertex *p*-center problem, Computers and Operations Research 33 (2006) 1420–1436.
[14] M.P. Scapparra, S. Pallotino, M.G. Scutellà, Large-scale local search heuristics for the capacitated vertex *p*-center problem, Networks 43 (2004) 241–255.
[15] S. Scheuerer, R. Wendolsky, A scatter search heuristic for the capacitated clustering problem, European Journal of Operational Research 169 (2006) 533–547.