European Journal of Operational Research 252 (2016) 204–212

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Decision Support

A branch-and-bound algorithm for the maximum capture problem with random utilities

Alexandre S. Freire[b], Eduardo Moreno[a,*], Wilfredo F. Yushimito[a]

[a] Faculty of Engineering and Sciences, Universidad Adolfo Ibañez, Santiago, Chile
[b] Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, Brazil

ARTICLE INFO

Article history:
Received 3 November 2014
Accepted 15 December 2015
Available online 22 December 2015

Keywords:
Facility location
Branch and bound
Maximum capture
Random utility model

ABSTRACT

The MAXIMUM CAPTURE PROBLEM WITH RANDOM UTILITIES seeks to locate new facilities in a competitive market such that the captured demand of users is maximized, assuming that each individual chooses among all available facilities according to the well-know a random utility model namely the multinomial logit. The problem is complex mostly due to its integer nonlinear objective function. Currently, the most efficient approaches deal with this complexity by either using a nonlinear programing solver or reformulating the problem into a Mixed-Integer Linear Programing (MILP) model. In this paper, we show how the best MILP reformulation available in the literature can be strengthened by using tighter coefficients in some inequalities. We also introduce a new branch-and-bound algorithm based on a greedy approach for solving a relaxation of the original problem. Extensive computational experiments are presented, benchmarking the proposed approach with other linear and non-linear relaxations of the problem. The computational experiments show that our proposed algorithm is competitive with all other methods as there is no method which outperforms the others in all instances. We also show a large-scale real instance of the problem, which comes from an application in park-and-ride facility location, where our proposed branch-and-bound algorithm was the most effective method for solving this type of problem.

1. Introduction

In recent years, competitive facility location models have received considerable attention both due to their interesting theoretical aspects and their practical applications. These models extend conventional facility location models to a more complex scenario, in which (a) companies compete for their market share and (b) the choices of independent decision makers, such as customers, are considered. As an example, we can think of a company that wants to locate r new supermarkets in a geographical zone where some supermarkets are already located (the competitors). The competitive facility location problem consists of choosing, from a given set of available locations, the locations for these r new facilities such that the demand captured by them (i.e. market share) is maximized.

This problem can be traced back to Hotelling's (1929) optimal location of two competing facilities on a line segment, and it was later embedded within the location theory, initially by Slater (1975) and further developed by Hakimi (1983).

In general, the literature considers that customers choose among different alternatives based on a given utility function that depends on a set of facility attributes (e.g., distance, transportation costs and waiting times, among others). The first deterministic model was proposed by ReVelle (1986), in which customers choose the closest facility among different competitors. However, these models imply an "all or nothing" assignment, in which the demand of a given point is assigned entirely to one facility. An alternative approach is proposed in the gravity-based model (Huff, 1964; Reilly, 1931), in which the demand captured by a facility is proportional to the "attractiveness" of the facility and inversely proportional to a power of the distance. Drezner and Eiselt (2002) and Berman, Drezner, Drezner, and Krass (2009) provide a comprehensive review of these different models.

Another alternative approach to the "all or nothing" assignment is to estimate the market share obtained by each facility through a random utility model. In random utility models (e.g., logit or probit models; see McFadden, 1973 or Ben-Akiva and Lerman, 1985), the utilities of economic agents are essentially derived from their preferences among a set of discrete options. In this case, the

* Corresponding author. Tel.: +56 223311351; fax: +56 223311351.
E-mail addresses: afreire@ime.usp.br (A.S. Freire), eduardo.moreno@uai.cl (E. Moreno), wilfredo.yushimito@uai.cl (W.F. Yushimito).

http://dx.doi.org/10.1016/j.ejor.2015.12.026
0377-2217/

problem can be stated as follows: given a set of customers and their respective demands, a set of open facilities (the competitors), and a set of available locations, the problem is to locate r new facilities such that the expected market share captured by the new facilities is maximized, where the market share captured by each selected facility is estimated through a random utility model (e.g. logit). This problem is referred to as the MAXIMUM CAPTURE PROBLEM WITH RANDOM UTILITIES (or MCRU problem, for short) and it was first introduced by Benati and Hansen (2002) with the multinomial logit model (MNL) as the underlying random utility model. Recent applications for this model includes locating schools (Haase & Müller, 2012), preventive healthcare facilities (Haase & Müller, 2015; Zhang, Berman, & Verter, 2012), and siting park-and-ride facilities (Aros-Vera, Marianov, & Mitchell, 2013).

Since the multinomial logit model is nonlinear by nature, modeling the MCRU problem usually results in nonlinear integer programing models, which in general are difficult to solve. Benati and Hansen (2002) have proposed different approaches to address the problem, namely concave programing, integer fractional programing and submodular maximization. The computational analysis presented in the cited paper shows that the concave programing, which is basically a branch-and-bound algorithm with a concave relaxation of the problem as dual bound, behaves better than the other two approaches.

Alternatively, equivalent Mixed-Integer Linear Programing (MILP) formulations have been proposed by Benati and Hansen (2002), Haase (2009), Aros-Vera et al. (2013), and Zhang et al. (2012). These formulations were recently evaluated by Haase and Müller (2014) to provide a computational comparison of them, being the model by Haase (2009) the most efficient in practice.

In this paper, we show how the MILP model introduced by Haase (2009) can be strengthened by using tighter coefficients in a class of inequalities. We also introduce a greedy algorithm for solving a relaxation of the MCRU problem, which is embedded into a branch-and-bound (B&B) algorithm to compute dual bounds. The success of a B&B algorithm relies basically on finding a good threshold between the quality of the bounds and the computational effort needed to calculate them. In fact, the obtained dual bounds are not necessarily sharper than the ones given by the known linear and nonlinear formulations of the problem, but they can be calculated much faster than the others. This allows to explore more nodes of the B&B tree, which in many cases is more effective than spending too much time computing better bounds. Moreover, the proposed B&B algorithm can be easily implemented and does not make use of any external solver, while all the other methods considered here do.

To evaluate the algorithm, extensive computational results are obtained for instances from three different datasets, namely the UflLib repository, the randomly generated instances introduced by Haase and Müller (2014), and a relatively large size instance (82341 customers and 59 available locations) that comes from a real application in location of park-and-ride facilities in New York City (Holguín-Veras, Reilly, Aros-Vera, Yushimito, & Isa, 2012). The methods considered here for comparison are the concave programing approach introduced by Benati and Hansen (2002), the MILP formulation introduced by Haase (2009) (using the tighter coefficients proposed in this work) and the proposed B&B algorithm. Results show that the proposed B&B algorithm is competitive with other available methods on all instances, and the most efficient method for solving the large real instance mentioned above.

The remainder of this paper is organized as follows. In Section 2, we present the notation and definitions used throughout this paper. In Section 3, we present some mathematical formulations for the MCRU problem found in the literature and show how the MILP model introduced by Haase (2009) can be strengthened by using tighter coefficients in a class of inequalities. In Section 4, we introduce a new B&B algorithm for the MCRU problem. The computational results are presented in Section 5. Finally, in Section 6 we draw some conclusions and present opportunities for future work.

2. Problem description

In this section, we give a formal description of the MCRU problem. Before describing the problem itself, we first explain the behavioral rationale underlying the customers' decisions, in which the market share captured by a particular facility is based on the preferences of the customers, which results in a choice probability of selecting a particular facility.

2.1. Behavioral rationale and choice probability

Let S be a set of customers and H be a set of open facilities. Each customer $s \in S$ receives a utility $\tilde{u}_{sl}$ for choosing the facility $l \in H$. Assuming that customers behave rationally, each customer selects the facility that provides the highest utility value. That is, a customer $s \in S$ chooses a facility $l \in H$ if $\tilde{u}_{sl} \geq \tilde{u}_{sh}, \forall h \in H$.

In random utility theory, the utility $\tilde{u}_{sl}$ obtained by customer $s \in S$ choosing a facility $l \in H$ has two components: a deterministic part v_{sl} and a random term ϵ_{sl}, such that $\tilde{u}_{sl} = v_{sl} + \epsilon_{sl}$. The deterministic part is typically referred to as the systematic component, because it is composed of a set of observable attributes (e.g., distance and time), whereas the random components represent the non-observable attributes. The joint density of the random vector $\epsilon_s = \{\epsilon_{s1}, \ldots, \epsilon_{sl}\}$, denoted by $f(\epsilon_s)$, allows us to state the probability of choosing an alternative. According to McFadden (1973), whenever the elements in ϵ_s are identically and independently distributed, they have equal variability among cases, and $f(\epsilon_s)$ follows a Generalized Extreme Value (GEV) distribution (i.e., Gumbel distribution), the model is referred to as the multinomial logit model, and the probability that a customer s selects a facility l from the given set H of open facilities is given by the following equation:

$$p_{sl} = \frac{e^{v_{sl}}}{\sum_{h \in H} e^{v_{sh}}} \tag{1}$$

2.2. Problem description

In the MCRU problem, it is given a set L of available locations, a set A of open facilities (the competitors) and a set S of customers. For simplicity, we sometimes refer to an available location l, where a new facility can be located, as a facility itself. For each customer $s \in S$, it is given a positive demand d_s and a utility v_{sl} for choosing the facility located in $l \in L \cup A$. The objective is to choose a subset $L^* \subset L$ of r locations where new facilities can be located, such that the expected demand captured by the new facilities is maximized, which is given by

$$\sum_{s \in S} \sum_{l \in L^*} d_s p_{sl} \tag{2}$$

According to (1), the probability that a user $s \in S$ chooses a facility $l \in L^*$ is given by

$$p_{sl} = \frac{e^{v_{sl}}}{\sum_{h \in L^* \cup A} e^{v_{sh}}} \tag{3}$$

Note that, w.l.o.g., we can assume that there is a single open facility (i.e., $|A| = 1$). If there is more than one open facility (i.e., $|A| > 1$), we can represent them as a single facility a such that $v_{sa} = \log(\sum_{i \in A} e^{v_{si}})$. Hence, for simplicity, we assume that there is a single open facility a. Note that higher values of v_{sa} represent a problem with stronger incumbent competitors.

Given a subset $H \subset L$ of open facilities, let $\phi(H, s, l)$ be the expected demand of customer s captured by the facility located in l, which is given by

$$\phi(H,s,l) = d_s \frac{e^{v_{sl}}}{\sum_{h\in H\cup\{a\}} e^{v_{sh}}} = \frac{d_s}{\sum_{h\in H\cup\{a\}} e^{(v_{sh}-v_{sl})}} \tag{4}$$

Because the total demand is split between the open facilities, we have

$$\sum_{s\in S}\sum_{l\in H\cup\{a\}} \phi(H,s,l) = \sum_{s\in S} d_s \tag{5}$$

Therefore, the MCRU problem seeks for a subset $L^* \subset L$ of r locations that maximizes $\sum_{s\in S}\sum_{l\in L^*}\phi(L^*,s,l)$ or, equivalently, minimizes $\Sigma_{s\in S}\phi(L^*, s, a)$. In the second case, we are minimizing the expected demand captured by the competitor a.

3. Mathematical formulations for the MCRU problem

In this section, we present some approaches based on mathematical formulations to solve the MCRU problem, namely the concave programing approach introduced by Benati and Hansen (2002) and the MILP formulation introduced by Haase (2009). Finally, we show how this last formulation can be strengthened by using tighter coefficients in a class of inequalities.

3.1. A concave programing approach

Benati and Hansen (2002) have proposed different approaches to solve the MCRU problem, namely concave programing, integer fractional programing and submodular maximization, being the first one the most promising approach, according to the computational results presented in the cited paper. We now present the main idea behind the concave programing approach.

Consider a binary variable x_l, for each $l \in L$, with the interpretation that $x_l = 1$ if and only if a new facility is located in l (i.e., x is the characteristic vector of the solution). A natural formulation for the MCRU problem can be obtained by simply rewriting the statement of the problem using the x variables, which results in the following integer nonlinear programing model:

$$\max \sum_{s\in S}\sum_{l\in L} d_s \frac{e^{v_{sl}}x_l}{e^{v_{sa}} + \sum_{h\in L} e^{v_{sh}}x_h}$$

$$(\text{INLP}_{\text{MCRU}}) \quad \text{s.t.} \sum_{l\in L} x_l = r \tag{1.1}$$

$$x_l \in \{0,1\}, \forall l \in L \tag{1.2}$$

In Benati and Hansen (2002) it is shown that the objective function of the continuous relaxation of $(\text{INLP}_{\text{MCRU}})$ is concave. The authors use this fact to calculate an upper bound by relaxing the integrality constraints of $(\text{INLP}_{\text{MCRU}})$ and then solving the problem by gradient optimization. To solve the original problem, this upper bound calculation is embedded into a branch-and-bound algorithm.

3.2. Haase's MILP formulation

As shown in the previous section, a natural formulation for the MCRU problem results in a nonlinear model. Alternatively, research efforts have been made on finding equivalent MILP formulations. Recently, Haase and Müller (2014) provided a computational comparison of the different MILP formulations in the literature, concluding that the formulation introduced by Haase (2009) outperforms the other models studied. In this section, we present the MILP formulation introduced by Haase (2009) for the MCRU problem and we show how one of its constraints can be strengthened.

The variables x are used with the same interpretation as in $(\text{INLP}_{\text{MCRU}})$ and we introduce a variable p_{sl}, for each $s \in S$ and $l \in L\cup\{a\}$, to represent the probability that the customer s chooses the facility located in l. To simplify the presentation of the formulation, we define the constant $\gamma_{sl} = e^{(v_{sl}-v_{sa})}$, for each $l \in L$ and $s \in S$. We now present the MILP formulation introduced by Haase (2009).

$$\max \sum_{s\in S}\sum_{l\in L} d_s p_{sl}$$

$$(\text{MILP}_{\text{MCRU}}) \text{ s.t.} \quad \gamma_{sl} p_{sa} \geq p_{sl}, \quad \forall s \in S, \forall l \in L \tag{2.1}$$

$$p_{sl} \leq \frac{\gamma_{sl}}{1+\gamma_{sl}} x_l, \quad \forall s \in S, \forall l \in L \tag{2.2}$$

$$\sum_{l\in L\cup\{a\}} p_{sl} = 1, \quad \forall s \in S \tag{2.3}$$

$$\sum_{l\in L} x_l = r, \tag{2.4}$$

$$p_{sl} \geq 0, \forall s \in S, \quad \forall l \in L\cup\{a\} \tag{2.5}$$

$$x_l \in \{0,1\}, \quad \forall l \in L \tag{2.6}$$

Considering the objective function, we see that constraints (2.1) are satisfied with equality when $x_l = 1$. Thus,

$$p_{sl} = \gamma_{sl} p_{sa} = \frac{e^{v_{sl}}}{e^{v_{sa}}} \cdot \frac{e^{v_{sa}}}{\sum_{h\in L} e^{v_{sh}}x_h + e^{v_{sa}}} = \frac{e^{v_{sl}}}{\sum_{h\in L} e^{v_{sh}}x_h + e^{v_{sa}}},$$

which is precisely the probability that a customer chooses the facility located in l, according to Eq. (3).

Constraints (2.2) state that the probability of a customer choosing a facility located in l can be positive only if the respective location is chosen (i.e., if $x_l = 0$, then $p_{sl} = 0$ for each $s \in S$).

3.3. Strengthening the Haase's MILP formulation

In constraints (2.2), we can replace the coefficient of x_l by any constant greater than or equal to the maximum value that p_{sl} can achieve. Aros-Vera et al. (2013) proposed an analogous formulation, in which constraints (2.2) are presented with the constant 1 rather than $\frac{\gamma_{sl}}{1+\gamma_{sl}}$, which is clearly a weaker formulation. We now show that constraints (2.2) can be strengthened even further.

Lemma 1. *Given a customer $s \in S$ and a location $l \in L$, let $L(s, l) \subset L$ be a subset of r locations such that $l \in L(s, l)$ and $v_{sh} \leq v_{st}$ for each $h \in L(s, l)\backslash\{l\}$ and $t \in L\backslash L(s, l)$. Then, $L(s, l)$ is a subset of r locations that maximizes the probability that the customer s chooses the facility located in l.*

Proof. In contrast, suppose that $H \subset L$ is a subset of r locations containing l such that $\frac{e^{v_{sl}}}{\sum_{h\in H} e^{v_{sh}}} > \frac{e^{v_{sl}}}{\sum_{t\in L(s,l)} e^{v_{st}}}$. Thus, there are locations $h \in H$ and $t \in L(s, l)$ such that $v_{st} > v_{sh}$, $t \in L(s, l)\backslash\{l\}$ and $h \in L\backslash L(s, l)$, which is in contradiction to the Lemma's hypothesis. □

As a consequence of Lemma 1, $\frac{e^{v_{sl}}}{e^{v_{sa}} + \sum_{h\in L(s,l)} e^{v_{sh}}}$ is the sharpest upper bound on the value that variable p_{sl} can achieve. Thus, replacing $\frac{\gamma_{sl}}{1+\gamma_{sl}}$ by $\frac{\gamma_{sl}}{1+\sum_{h\in L(s,l)}\gamma_{sh}}$ in constraints (2.2) leads to a formulation for the MCRU problem which is stronger than $(\text{MILP}_{\text{MCRU}})$.

Below, we give an example in which replacing $\frac{\gamma_{sl}}{1+\gamma_{sl}}$ by $\frac{\gamma_{sl}}{1+\sum_{h\in L(s,l)}\gamma_{sh}}$ in constraints (2.2) produces a strictly smaller value in the objective function of the linear relaxation of $(\text{MILP}_{\text{MCRU}})$.

Table 1
Row s column l contains v_{sl}.

	l_1	l_2	l_3	l_4	a
s_1	2	1	2	1	2
s_2	2	2	1	1	2
s_3	2	1	1	2	2
s_4	1	2	2	1	2

Table 2
Row s column $L_{i,j}$ contains $\sum_{l\in L_{i,j}} \phi(L_{i,j}, s, l)$. The last row contains the total expected demand captured by the facilities in $L_{i,j}$, which is given by $\sum_{s\in S}\sum_{l\in L_{i,j}} \phi(L_{i,j}, s, l)$. The values are rounded.

	$L_{1,2}$	$L_{1,3}$	$L_{1,4}$	$L_{2,3}$	$L_{2,4}$	$L_{3,4}$
s_1	0.58	**0.67**	0.58	0.58	0.42	0.58
s_2	**0.67**	0.58	0.58	0.58	0.58	0.42
s_3	0.58	0.58	**0.67**	0.42	0.58	0.58
s_4	0.58	0.58	0.42	**0.67**	0.58	0.58
Total	**2.40**	**2.40**	2.25	2.25	2.16	2.16

Example 2. Assume that $v_{sa} = v_{sl}$, for each $s \in S$ and $l \in L$. In this case, any subset of r locations is an optimal solution to the MCRU problem, in which $p_{sa} = \frac{1}{r+1}$ and $p_{sl} = \frac{1}{r+1}$, for each location l in the chosen subset, obtaining an optimal objective value of $\frac{r}{r+1}\sum_{s\in S} d_s$. For the linear relaxation of ($\text{MILP}_{\text{MCRU}}$), since $\gamma_{sl} = 1$ it has an optimal solution equal to $x_s = \frac{r}{|L|}$ and $p_{sl} = \frac{1}{|L|+1}$ for each $s \in S$ and $l \in L$, obtaining an objective value of $\frac{|L|}{|L|+1}\sum_{s\in S} d_s$. However, replacing $\frac{\gamma_{sl}}{1+\gamma_{sl}}$ by $\frac{\gamma_{sl}}{1+\sum_{h\in L(s,l)}\gamma_{sh}}$ in constraints (2.2), we obtain the constraint $p_{sl} \le \frac{1}{r+1}x_l$, for each $s \in S$ and $l \in L$. Hence, the previous solution is infeasible and now the optimal solution is $x_s = \frac{r}{|L|}$ and $p_{sl} = \frac{r}{(r+1)(|L|)}$, for each $s \in S$ and $l \in L$, obtaining an objective value equal to the optimal integer solution.

4. A branch-and-bound algorithm for the MCRU problem

In this section, we introduce a branch-and-bound (B&B) algorithm for the MCRU problem. We first present a greedy algorithm for solving a relaxation of the original problem, which is later embedded into a B&B algorithm to compute upper bounds in each node of the B&B tree.

4.1. A greedy algorithm for computing upper bounds

As in the previous formulations, our algorithm also represents a solution by a vector $x \in [0, 1]^{|L|}$, with the interpretation that $x_l = 1$ if and only if a new facility is open in location l. To illustrate the concept behind our algorithm, we first present an example of how it works.

Consider an instance of the MCRU problem in which $L = \{l_1, l_2, l_3, l_4\}$ and $S = \{s_1, s_2, s_3, s_4\}$, we want to select $r = 2$ locations and we have unitary demands (i.e., $d_s = 1$ for each $s \in S$). Table 1 presents the deterministic utility v_{sl} for each $s \in S$ and $l \in L \cup \{a\}$.

We denote the subset containing the locations l_i and l_j by $L_{i,j} = \{l_i, l_j\}$. Given a feasible solution $L_{i,j}$, the expected demand of a customer s captured by the facilities located in l_i and l_j is given by

$$\sum_{l\in L_{i,j}} \phi(L_{i,j}, s, l) = \frac{e^{v_{sl_i}} + e^{v_{sl_j}}}{e^{v_{sl_i}} + e^{v_{sl_j}} + e^{v_{sa}}} \tag{6}$$

In Table 2, we show the objective value of each feasible solution according to Eq. (6). As shown in this table, considering each customer independently, the only local optimal solutions for customers s_1, s_2, s_3 and s_4 are $L_{1,3}$, $L_{1,2}$, $L_{1,4}$ and $L_{2,3}$, respectively. The primary objective of our algorithm is to solve a relaxed problem in which we consider a local optimal solution for each customer and from this solution derive an upper bound for the global optimal value. In the given example, $\sum_{l\in L_{1,3}} \phi(L_{1,3}, s_1, l) + \sum_{l\in L_{1,2}} \phi(L_{1,2}, s_2, l) + \sum_{l\in L_{1,4}} \phi(L_{1,4}, s_3, l) + \sum_{l\in L_{2,3}} \phi(L_{2,3}, s_4, l) = 4 \times 0.\bar{6} = 2.\bar{6}$ is an upper bound on the global optimal value, which is ≈ 2.40 (the global optimal solutions are $L_{1,2}$ and $L_{1,3}$). Intuitively, considering all available locations (i.e., potential facilities) in decreasing order of deterministic utility provided to a customer s, the corresponding local optimal solution is obtained by selecting the first r potential facilities. Note that locations l_1, l_2, l_3 and l_4 appear in a local optimal solution 3, 2, 2, and 1 times, respectively, which leads to the fractional solution $x = (\frac{3}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4})$.

In our B&B algorithm, we first solve the relaxation of the original problem as described above, obtaining a vector $x \in [0, 1]^{|L|}$ and an upper bound on the optimal value. If x is integral, it corresponds to an optimal solution for the original problem, and the upper bound calculated is precisely the optimal value. Otherwise, we choose a fractional variable x_l and create a subproblem in which the constraint $x_l = 0$ must be satisfied and another subproblem in which $x_l = 1$ must be satisfied. Thus, at any node of the B&B tree, we have a subset $L_0 \subset L$ of locations that are not allowed to be chosen and a subset $L_1 \subset L$ of locations that must be chosen. At the root node of the B&B tree, we have that $L_0 = L_1 = \emptyset$.

Algorithm MAX_UTILITY_GREEDY, presented below, maximizes the total expected utility received by each customer individually (i.e., it finds a local optimal solution for each customer). It considers two subsets of locations, L_0 and L_1, where locations in L_0 are forbidden and locations in L_1 are enforced. Given the subsets L_0 and L_1 as parameters, the algorithm returns an upper bound UB on the optimal value and a vector $x \in [0, 1]^{|L|}$, possibly fractional, representing all the customer's choices.

Clearly, the algorithm MAX_UTILITY_GREEDYterminates in polynomial running time. Moreover, Lemma 3 can be easily shown, which implies that the algorithm MAX_UTILITY_GREEDYis correct.

Lemma 3. *Let* (UB, x) *be the solution returned by the* MAX_UTILITY_GREEDY*algorithm, and let* OPT *be the value of an optimal solution for the corresponding MCRU subproblem, in which the locations in* L_0 *are forbidden and the locations in* L_1 *are enforced. The following properties are satisfied:*

(a) $\sum_{l\in L} x_l = r$;
(b) $x_l = 0$ *for each* $l \in L_0$;
(c) $x_l = 1$ *for each* $l \in L_1$;
(d) UB $\ge$ OPT;
(e) *If* x *is integral, then* UB = OPT.

4.2. Computing lower bounds

We first present a greedy heuristic for finding a feasible solution for the MCRU problem that leads to a lower bound on the optimal value. This heuristic has been studied in Benati and Hansen (2002), proving that it provides an ρ-approximation, with $\rho = \frac{e}{e-1}$.

Given subsets $L_0 \subseteq L$ and $L_1 \subseteq L$, the algorithm GREEDY_HEURISTIC, presented below, returns a feasible solution for the corresponding instance of the MCRU problem (i.e., a subset $H \subseteq L$ of r locations such that $L_1 \subseteq H$ and $L_0 \cap H = \emptyset$).

The choice of the next element to be included in H is made (in line 3) by selecting a location that provides the highest increase in the objective function.

4.3. The branch-and-bound procedure

The method that we propose here to solve the MCRU problem is a standard B&B procedure which invokes MAX_UTILITY_GREEDY and

GREEDY_HEURISTIC algorithms as subroutines for calculating the upper and lower bounds, respectively. We skip the unnecessary implementation details and focus only on the main aspects of the algorithm.

At each step, we have a set of live nodes, which correspond to the nodes that were not explored yet and are children of some explored node. Initially, the set of live nodes contains only the root node, which has no fixed variables. The choice of the next node to be explored is made by taking one live node with the maximum upper bound, aiming to decrease the global upper bound as soon as possible. After removing a node η from the set of live nodes, two new nodes γ_0 and γ_1 are created as children of η. A variable $x(\eta)_l$ with the most fractional value is selected (ties are broken by taking the first one in lexicographical order) and then all nodes in the subtrees rooted in γ_0 or γ_1 will consider location l as forbidden or enforced, respectively. Naturally, if a node provides an integer solution or an upper bound which is not larger than the value of a known feasible solution, we do not add his children to the set of live nodes. The algorithm stops when either global upper and lower bounds are equal or the given time limit is exceeded.

Before adding a node to the set of live nodes, we could call GREEDY_HEURISTIC to find a feasible solution, with the respective fixed variables, to attempt to increase the global lower bound. We decided not to implement this approach, because empirical tests revealed that this approach slows the procedure rather than speeding it up.

5. Computational experiments

We now present computational results for instances obtained from two different datasets, namely the UflLib repository and the randomly generated instances introduced by Haase and Müller (2014), and also for a relatively large size instance (82341 customers and 59 available locations) that comes from a real application in location of park-and-ride facilities in New York City. The methods considered here for comparison are the concave programing approach introduced by Benati and Hansen (2002), the MILP formulation introduced by Haase (2009) (using the different coefficients, proposed in this work, for inequalities (2.2)) and the B&B algorithm introduced in Section 4.

We divide the presentation of the results into three parts. We first compare the presented relaxations, aiming to establish which one provides the best dual bound among them. We then compare the presented exact methods, aiming to establish which one is the most efficient in practice in terms of running time, regardless of the quality of the dual bounds. In these first two parts, we use the two datasets mentioned above. Finally, we apply the exact methods to solve the real problem instance which comes from an application in park-and-ride facility location in New York City.

Before presenting the results, we first describe some implementation details, as well as the machine configurations and the datasets.

5.1. Implementation details and machine configurations

We have implemented the MILP formulation introduced by Haase (2009), which is the most efficient MILP formulation for the MCRU problem, according to the computational study in Haase and Müller (2014). Besides this formulation as it was introduced, we consider here two other versions of it which differ only by the coefficient in constraints (2.2), as explained in Section 3.3. One of them is due to (Aros-Vera et al., 2013), while the other one was introduced in this work and, as shown in Lemma 1, it has the tightest coefficient among these three formulations. We denote the above three approaches by (H09), (AV13) and (FMY15), respectively. We solved these formulation using IBM-ILOG CPLEX 12.6 under default settings.

We have also implemented the concave programing approach introduced by Benati and Hansen (2002), as described in Section 3.1. More precisely, we embedded their nonlinear model in our B&B algorithm, replacing the calls of MAX_UTILITY_GREEDY by their concave relaxation. To solve this nonlinear relaxation, we used the method-of-moving-asymptotes (MMA) algorithm (Svanberg, 2002), implemented in NLopt 2.4.2 library (Johnson, 2014). We denote this approach by (CP).

The branch-and-bound algorithm presented in Section 4, which calls the MAX_UTILITY_GREEDY algorithm to compute upper bounds, was implemented in C++ programing language and compiled using GCC 4.4.6. We denote this approach by (MUG).

The same notation is used to designate an exact method as well as its corresponding relaxation, being clear by the context which of them is referred to. All computations were made in machines running Linux 2.6.32 under x86_64 architecture, with two quad-core Intel Xeon E5-2650 processors and 146 Gb of RAM.

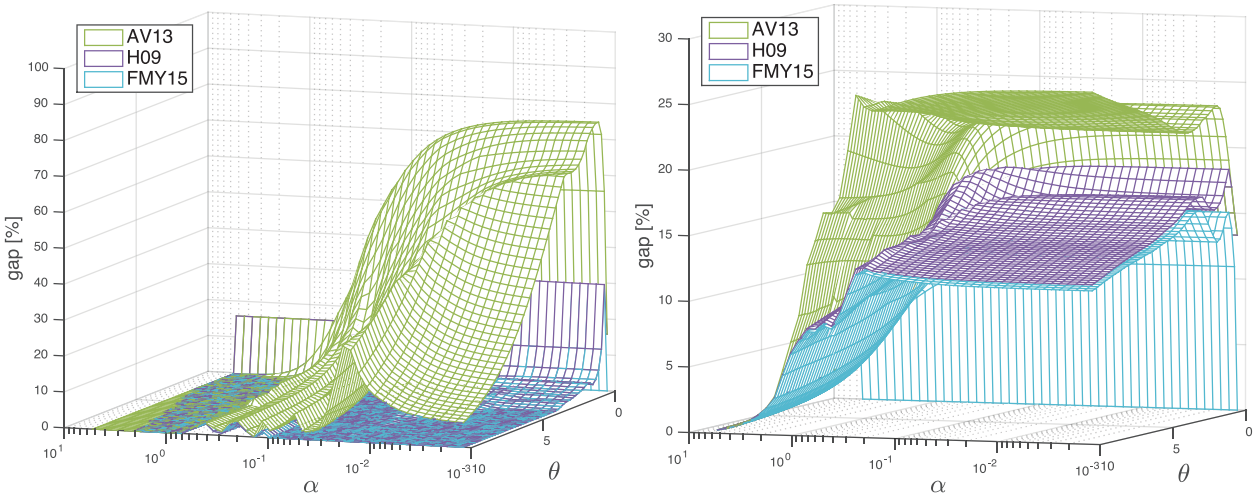
5.2. Description of the datasets

Our first dataset was generated as in Haase and Müller (2014). That is, we randomly locate $|S|$ customers and $|L|$ locations over a rectangular region of 30×30. We set a cost c_{sl} equal to the distance from each customer $s \in S$ to each location $l \in L$. To represent the competitors, we also randomly locate $\lceil |L|/10 \rceil$ points in the region, and we set the cost c_{sa} equal to the minimum distance between each customer $s \in S$ to these points. We generated instances with 50, 100, 200, and 400 customers; and 25, 50, and 100 locations. We denote this dataset by HM14.

We now describe the dataset which corresponds to the ORlib instances, taken from the UflLib repository (Hoefer, 2003), for the uncapacitated warehouse location problem. Since these instances were constructed for a slightly different problem, we only considered the demand and the cost from each customer for each facility c_{sa}. To introduce competition, for each customer s, we also randomly choose a subset of $\lceil |L|/10 \rceil$ facilities and set c_{sa} equal to the minimum cost among this subset. We omitted the instances cap71, cap72, cap73 and cap74 because they are too small and can be solved by all algorithms in a few seconds. We denote this dataset by ORlib.

Finally, we describe the dataset generated from a real instance of the MCRU problem. The instance was obtained from the NYMTC (2009) and it corresponds to an improved version of the location of park-and-ride facilities in New York City, presented in Aros-Vera et al. (2013). The data include 82341 trips from 3600 origins in New York, New Jersey and Connecticut to 317 destinations in Manhattan, representing the demand in the morning peak. The "facilities" available are park-and-ride locations which can be selected among 59 candidate locations (state-owned available parking lots). That is, each customer (a trip from one origin to another destination) chooses between the incumbent solution (a direct auto trip) or a given facility (an auto trip from origin to the park-and-ride, and a transit trip from the facility to its destination). Hence, $|S| = 82341$ and $|L| = 59$. The generalized costs used in the logit function consist of direct auto trip costs (including travel time, tolls, and auto costs) from an origin to the park-and-ride location, plus transit costs (travel times and waiting time) from that location to the destination. The incumbent cost only considers the direct auto trip from the origins to the destination. We denote this dataset by P&R-NYC.

For each instance, we have solved the problem for $r = 2, 3, \ldots, 10$. The deterministic part of the utility function is given by $v_{sl} = -\theta \cdot c_{sl}$, for each location, and by $v_{sa} = -\alpha \cdot \theta \cdot c_{sa}$, for the competitors. Parameter θ represents the sensitivity of customers



(a) $|S| = 100, |L| = 100, r = 5$ (b) cap101, $r = 5$

Fig. 1. Gaps between the linear relaxation of the MILP models and the best integer solution found on HM14 (on the left side) and ORlib (on the right side).

about the perceived utility and parameter α allows us to vary the competitiveness of the incumbent competitors. In each part of the experiments, we have considered a specific choice of parameters θ and α, which is described latter on.

5.3. Analysing the quality of the different relaxations

Although we proved that the linear relaxation (FMY15) dominates (H09), which in turn dominates (AV13), as explained in Section 3.3, we are still interested in comparing these three relaxations to establish how they differ from each other in practice. For this analysis, we have considered different values for θ and α, specifically $\theta = 0.001 \times (1.2)^i$, for $i = 1, 2, \ldots$ (up to $\theta = 10$) and $\alpha = 0.2 \times i$, for $i = 1, 2, \ldots$ (up to $\alpha = 10$).

In Fig. 1 we show the results for relaxations (FMY15), (H09) and (AV13), while in Fig. 2 we show the results for the relaxations (FMY15), (CP) and (MUG). We plot the gap between the value of each relaxation and the best-know feasible solution, for the different values of θ and α (in the horizontal axis). In these figures we present the numerical results for only one instance (a representative) from each dataset. For the other instances in each dataset basically the same pattern with no significant variations is obtained.

As Figs. 1 and 2 show, the quality of the relaxations strongly depends on the dataset and the values of the parameters α and θ. It can be seen in Fig. 1 that the relaxations (FMY15) and (H09) are very close to each other, while (AV13) is significantly weaker.

As Fig. 2 shows, for the HM14 dataset, the (FMY15) relaxation obtained gaps close to zero for all values of parameters α and θ, while (CP) and (MUG) relaxations obtained similar gaps in many cases, achieving a gap of ≈ 70 percent, depending on the values of these parameters. On the other hand, for ORlib dataset, the (CP) relaxation obtained gaps very close to zero for all instances, while the relaxations (FMY15) and (MUG) obtained similar gaps, achieving a gap of ≈ 20 percent, depending on the parameters θ and α.

Considering the time spent by each method to solve the corresponding relaxation, the (CP) was the slowest one, with an average time of 32.4 seconds per instance, while (FMY15) spent 15.72 seconds per instance, in average, and the time spent by (MUG) was negligible (less than a second for all instances). In the first two cases, the dispersion of the time was considerable high, with 15–20 percent of the instances being solved in less than a second.

The success of a B&B algorithm relies basically on finding the best balance between the quality of the bounds and the computational effort needed to calculate them. As Figs. 1 and 2 show, (MUG) does not provide an upper bound sharper than the considered linear and nonlinear formulations. On the other hand, the upper bound given by (MUG) can be calculated much faster than the others, allowing to explore more nodes of the B&B tree, which in many cases is more effective than spending too much time computing better bounds, as the next section shows.

5.4. Analysing the running time of the exact methods

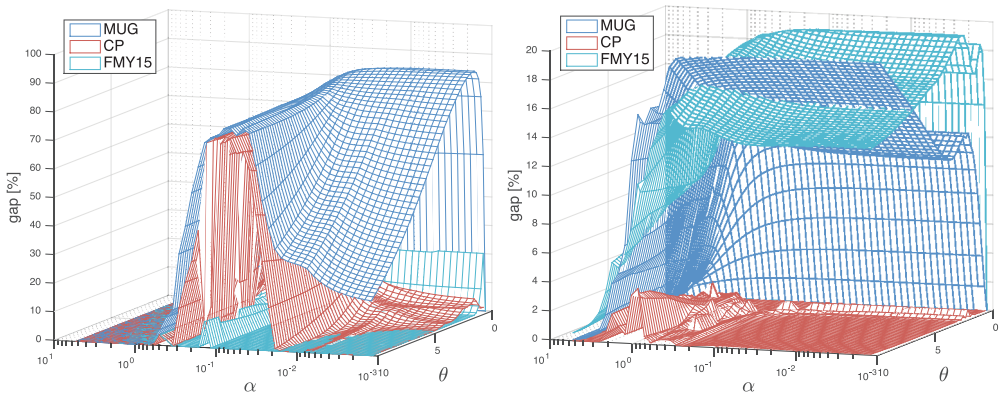
We now compare the performance of the exact methods considered in this paper. For this analysis, we have considered $\theta \in \{1, 5, 10\}$ and $\alpha \in \{0.01, 0.1, 1\}$, representative of the different behavior observed in Figs. 1 and 2. This results in a total of 972 instances from HM14 and 891 instances from ORlib. We have given a time limit of one hour per instance.

Note that all the methods considered here have in common that each one of them solves a relaxation embedded in a B&B algorithm. In the case of (CP) and (MUG), the B&B algorithm was implemented as explained in Section 4.3, while in the MILP approaches ((AV13), (H09) and (FMY15)) we have used CPLEX, which provides its own branch-and-cut implementation, as well as other advanced techniques (presolving, some different general purpose cutting planes generation, more sofisticated branching strategies, etc...). None of these techniques was used in our B&B algorithm.

In Fig. 3, we present the performance profiles (see Dolan & Moré, 2002) of the different methods on HM14 and ORlib. Each curve plots the fraction of instances solved by each method before a given time (in horizontal axis). Further details are provided in Tables 5 and 6.

As Fig. 3 shows, in both graphs clearly there are two different curve patterns, where (MUG) and (CP) follow the same pattern, while the MILP formulations follow another pattern.

For the HM14 dataset, although the MILP formulations have solved more instances than the other two methods, for almost 70 percent of the instances (MUG) was the fastest method. On the other hand, for the ORlib dataset, the MILP formulations were the slowest ones, being unable to solve any of the largest instances (capa, capb and capc). Comparing only the MILP formulations, (FMY15) and (H09) obtained similar performances for the HM14 dataset, while (FMY15) could solve $\approx$10 percent more instances



(a) $|S| = 50, |L| = 100, r = 3$ (b) cap134, $r = 4$

Fig. 2. Gaps between the relaxations ((FMY15), (H09) and (AV13)) and the best integer solution found for HM14 (on the left side) and ORlib (on the right side).

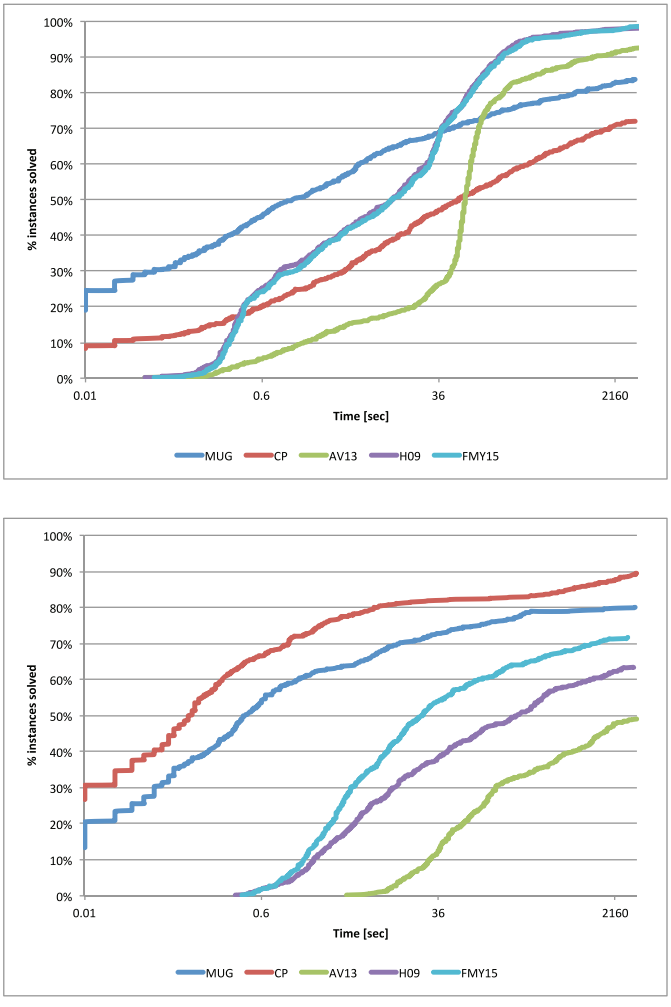


Fig. 3. Performance profile of each method for HM14 (on the top) and ORlib (on the bottom).

than (H09) for the ORlib dataset. For both datasets, (AV13) was slower than the other two MILP formulations.

Although the curves representing (MUG) and (CP) in Fig. 3 follow a similar pattern, further analysis shows that the performance of (CP) strongly relies on the quality of the relaxation in the root node of the B&B tree, while this behavior is not observed in the (MUG) method. This is also explained by the fact that the lower

Algorithm 1 MAX_UTILITY_GREEDY (S, L, L_0, L_1).

$x \leftarrow \mathbf{0}$, UB $\leftarrow 0$
for each $s \in S$ **do**
  $L_s \leftarrow L_1$
  while $|L_s| < r$ **do**
    $h \leftarrow \arg\max_{l \in L \setminus (L_s \cup L_0)} v_{sl}$
    $L_s \leftarrow L_s \cup \{h\}$,
  end while
  for each $l \in L_s$ **do**
    UB $\leftarrow$ UB $+ d_s \frac{e^{v_{sl}}}{\sum_{h \in L_s \cup \{a\}} e^{v_{sh}}}$
    $x_l \leftarrow x_l + \frac{1}{|S|}$
  end for
end for
return (UB, x)

Algorithm 2 GREEDY_HEURISTIC (S, L, L_0, L_1).

$H \leftarrow L_1$
while $|H| < r$ **do**
  $h \leftarrow \arg \max_{h \in L \setminus (L_0 \cup H)} \sum_{s \in S} \sum_{l \in H \cup \{h\}} \phi(H \cup \{h\}, s, l)$
  $H \leftarrow H \cup \{h\}$
end while
LB $\leftarrow \sum_{s \in S} \sum_{l \in H} \phi(H, s, l)$
return LB

bound found by the GREEDY_HEURISTIC algorithm is optimum in almost all cases and it is very close to the optimal in the other cases. Hence, the quality of the relaxation plays an important role. For example, for instances in HM14 with $\alpha = 0.1$, in which the initial gaps reach the peak (see Fig. 2), the (CP) method could solve only 156 of the 324 instances, while (MUG) solved 229 instances (≈46 percent more than (CP)) among this same subset of instances. This can be explained by the fact that, in general, the (CP) method spends considerably much more time solving its corresponding relaxation than (MUG), allowing it to visit only a small number of nodes of the B&B tree.

We conclude from the presented experiments that there is no method which overcomes the others in all instances and, for each method considered here (apart from (AV13)), there is a subset of instances for which the respective method is the most efficient. For HM14 instances, (MUG) was more efficient than (CP), while for the ORlib it was the opposite. Considering only the MILP formulations, (FMY15) provided the tightest relaxation and was more efficient than the others.

Table 3
Results for P&R-NYC dataset, grouped by r (9 instances per row).

	Solved instances		Spent time[a]		Initial gap[a] percent		Visited nodes[a]	
r	(CP)	(MUG)	(CP)	(MUG)	(CP)	(MUG)	(CP)	(MUG)
2	6	9	3727	69	6.55	14.87	13	111
3	6	9	2485	170	2.65	9.00	11	271
4	5	9	2338	411	1.79	6.07	7	725
5	5	9	1813	1303	1.07	4.15	7	2204
6	7	9	4707	3187	0.42	2.82	7	6753
7	6	9	1169	6562	0.26	1.94	5	13826
8	6	9	2441	10157	0.18	1.49	8	32078
9	6	6	4025	2995	0.10	0.38	12	6015
10	5	6	1469	3843	0.06	0.28	7	7370

[a] Average among solved instances.

Table 4
Results for P&R-NYC dataset, grouped by α and θ (9 instances per row).

		Solved instances		Spent time[a]		Initial gap[a] percent		Visited nodes[a]	
α	θ	(CP)	(MUG)	(CP)	(MUG)	(CP)	(MUG)	(CP)	(MUG)
0.5	0.5	9	9	22	3	0.00	0.57	1	1
0.5	1.0	9	9	22	2319	0.00	8.52	1	4701
0.5	2.0	9	9	44	3	0.00	0.01	1	1
1.0	0.5	9	7	6715	9149	5.11	12.01	20	23791
1.0	1.0	9	7	2523	6558	3.32	15.23	11	20490
1.0	2.0	0	7	–	8754	–	11.12	–	21980
2.0	0.5	6	9	6044	1729	0.01	0.58	23	2996
2.0	1.0	1	9	23928	1485	0.04	0.70	13	2574
2.0	2.0	0	9	–	1856	–	0.56	–	3082

[a] Average among solved instances.

Table 5
Results for P&R-HM14 dataset, grouped by $|S|$ and $|L|$ (81 instances per row).

		Solved instances					Spent time[a]					Visited nodes[a]				
$\|S\|$	$\|L\|$	(AV13)	(H09)	(FMY15)	(CP)	(MUG)	(AV13)	(H09)	(FMY15)	(CP)	(MUG)	(AV13)	(H09)	(FMY15)	(CP)	(MUG)
50	25	81	81	81	69	81	36.5	27.9	28.1	13.8	0.2	31	2	1	451	11070
50	50	81	81	81	67	79	105.8	28.4	26.6	211.1	106.3	5188	7	7	5375	4141023
50	100	75	80	81	48	61	382.8	138.0	270.3	272.5	167.1	12185	51	33	461	4480988
100	25	81	81	81	67	81	45.7	18.4	19.8	55.3	1.7	144	1	0	2573	40582
100	50	81	81	81	58	72	198.3	20.9	22.9	162.6	207.9	7303	4	5	1696	4778935
100	100	73	79	81	49	58	706.0	119.5	162.7	289.1	200.3	7655	81	70	368	2959530
200	25	81	81	81	74	81	55.9	14.0	14.3	142.7	9.4	336	2	2	2922	110327
200	50	81	81	81	57	67	485.1	42.2	39.6	254.7	211.1	5873	3	2	1316	2400039
200	100	67	79	81	46	46	802.4	227.9	663.2	404.5	112.7	4135	138	154	228	686808
400	25	81	81	81	77	81	89.5	49.7	34.3	133.0	11.7	131	2	1	1367	49637
400	50	76	78	81	52	62	687.3	91.0	284.4	388.3	259.9	3065	11	11	970	1044952
400	100	63	74	76	36	45	766.8	237.3	552.2	355.7	299.2	456	104	114	172	758270
Total		921	957	967	700	814	344.1	83.0	174.6	203.4	117.9	3856	33	33	1666	1750200

[a] Average among solved instances.

5.5. Results for an application in park-and-ride facility location

We now compare the performance of the exact methods presented on a real instance of the problem, which comes from an application in park-and-ride facility location. Since utilities of each customer are linearly correlated with the travel time of each trip, for this analysis we have considered $\theta \in \{0.5, 1, 2\}$ and $\alpha \in \{0.5, 1, 2\}$, resulting in a total of 81 instances. We have given a time limit of 8 hours per instance. For all instances in this dataset, all the MILP formulations could not solve even the linear relaxation. Hence, we limit the analysis to the other two methods, namely (MUG) and (CP).

In Table 3 the results are grouped by the value of r, while in Table 4 the results are grouped by the values of α and θ.

As Table 3 shows, for the instances with r from 2 to 8, (MUG) solved all the 63 instances, while (CP) solved only 41 instances ($\approx$ 35 percent less). In all cases, the initial gap obtained by (CP) was tighter than (MUG). On the other hand, (MUG) could visit much more nodes of the B&B tree than (CP) and, in most cases, it was also much faster than (CP).

Considering the instances grouped by the values of α and θ (see Table 4), we observe that the choice of these parameters strongly influences the results, which was predictable if we consider the analysis presented in Section 5.3. For example, for $\alpha = 0.5$, the (CP) method solved all the instances in the root node of the B&B tree (i.e., the relaxation found an integer solution) in a few seconds, while for $\alpha = \{1, 2\}$ and $\theta = 2$ it could not solve (in 8 hours) the relaxation for any of the 18 instances. This shows that the performance of the (CP) method is very sensitive not only to the input size but also to the numbers which appear as input (the utility function). This discrepancy is not observed in the (MUG) method. Moreover, appart from 3 rows (2nd, 4th and 5th) of Table 4, (MUG) was considerably faster than (CP) and it could solve a greater number of instances as well.

From Tables 3 and 4, we conclude that the performance of the (MUG) method is more related to the value of r than to the values of the parameters θ and α, while the (CP) method behaves in the opposite way. As observed, (CP) was able to visit a very small number of B&B nodes and, as a consequence, it can solve the problem only in the cases in which the initial gap is very close to

Table 6
Results for P&R-ORlib dataset, grouped by problem name (81 instances per row).

	Solved instances					Spent time[a]					Visited nodes[a]				
Name	(AV13)	(H09)	(FMY15)	(CP)	(MUG)	(AV13)	(H09)	(FMY15)	(CP)	(MUG)	(AV13)	(H09)	(FMY15)	(CP)	(MUG)
cap101	81	79	81	81	81	365.1	147.6	13.8	0.4	0.2	99734	26087	4111	34	9057
cap102	80	79	81	81	81	323.7	143.6	14.1	0.9	0.2	76577	27504	4840	170	11596
cap103	81	81	81	81	81	255.3	117.0	6.6	0.7	0.1	51206	18644	1387	86	7559
cap104	81	81	81	81	81	301.7	135.6	8.3	0.1	0.2	61925	23396	1862	7	10026
cap131	28	60	78	81	81	660.3	291.5	253.1	1.6	7.5	15401	16071	39303	59	296281
cap132	29	63	79	81	81	686.3	312.6	213.2	0.5	5.9	16610	16783	37362	15	225039
cap133	29	61	78	81	81	705.2	334.6	199.6	0.3	14.2	15555	17834	34694	8	543304
cap134	28	62	79	81	81	762.2	337.0	218.3	0.9	13.9	17608	17375	39494	36	525487
capa				48	21				737.5	356.3				112	308778
capb				49	23				665.6	471.4				110	393240
capc				53	21				477.0	296.5				61	225427
Total	437	566	638	798	713	413.7	215.5	114.2	117.4	39.2	57724	20893	20112	60	213407

[a] Average among solved instances.

zero. Instead, the (MUG) method calculates the dual bounds very quickly, allowing it to solve the problem also when the initial gap is not so tight.

6. Concluding remarks and future work

In this work, we have introduced a strengthened version of the MILP formulation presented in Haase (2009), which was the most efficient MILP formulation for the MCRU problem, according to the computational experiments presented by Haase and Müller (2014). As our computational results showed, the proposed strengthened MILP formulation provided slightly better dual bounds and also could solve more instances to optimality than the other MILP formulations.

We have also introduced a greedy algorithm for solving a relaxation of the MCRU problem, which is embedded into a branch-and-bound (B&B) algorithm to compute dual bounds. Considering all the exact methods discussed in this work (MILP formulations, a concave programing approach and our B&B algorithm), we concluded that there is no method which overcomes the others in all instances and, for each method considered here, there is a subset of instances for which the respective method is the most efficient. The proposed B&B algorithm was the most efficient method for solving a dataset which comes from an application in location of park-and-ride facilities in New York City. Moreover, this algorithm can be easily implemented and does not make use of any external solver, while all the other methods considered here do.

Note that the MCRU problem still has some simplifications compared to other facility location problems. There are two natural extensions to this family of facility location problems. One extension is to replace the cardinality constraint by a budget constraint, assuming that an opening cost is associated to each location and the total cost of the chosen locations cannot exceed a given budget. For this problem, our B&B algorithm can be adapted in a natural fashion, solving a knapsack problem to calculate the dual bounds. We leave further details and computational experiments for this variant of the problem for a future work. Another extension is to consider capacity constraints, which requires that the total expected demand captured by each facility do not exceed its given capacity. Such constraints can easily be incorporated into the MILP models, although maybe these models would become very hard to solve in practice. For this variant of the problem, even finding a feasible solution seems to be NP-hard.

Acknowledgments

The work of Alexandre S. Freire was supported by FAPESP (Procs: 2012/17585-9 and 2013/03447-6). Eduardo Moreno was supported by FONDECYT grant 1130681. Wilfredo F. Yushimito was supported by FONDECYT Iniciación grant 11121439. The authors want to thank Professor José Holguín-Veras from Rensselaer Polytechnic Institute for providing us the NYC dataset. We would especially like to thank the anonymous referees for their helpful feedback that significantly improved the presentation of the article.

References

Aros-Vera, F., Marianov, V., & Mitchell, J. E. (2013). p-hub approach for the optimal park-and-ride facility location problem. *European Journal of Operational Research, 226*, 277–285.
Ben-Akiva, M. E., & Lerman, S. R. (1985). *Discrete choice analysis: Theory and application to predict travel demand volume 9 of Transportation Studies*. The MIT press.
Benati, S., & Hansen, P. (2002). The maximum capture problem with random utilities: Problem formulation and algorithms. *European Journal of Operational Research, 143*, 518–530.
Berman, O., Drezner, T., Drezner, Z., & Krass, D. (2009). *Modeling competitive facility location problems: New approaches and results* (pp. 156–181). San Diego, CA: INFORMS. TutORials in Operations Research.
Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming, 91*, 201–213.
Drezner, T., & Eiselt, H. (2002). Consumers in competitive location models. *Facility Location: Applications and Theory* (pp. 151–178). Springer Verlag, Berlin.
Haase, K. (2009). Discrete location planning. *Technical report wp-09-07*. Institute for Transport and Logistics Studies, University of Sydney.
Haase, K., & Müller, S. (2012). Management of school locations allowing for free school choice. *Omega, 41*, 847–855.
Haase, K., & Müller, S. (2014). A comparison of linear reformulations for multinomial logit choice probabilities in facility location models. *European Journal of Operational Research, 232*, 689–691.
Haase, K., & Müller, S. (2015). Insights into clients' choice in preventive health care facility location planning. *OR Spectrum, 37*, 273–291.
Hakimi, S. (1983). On locating new facilities in a competitive environment. *European Journal of Operational Research, 12*, 29–55.
Hoefer, M. (2003). UflLib: Benchmarks instances for the uncapacitated facility location problem.
Holguín-Veras, J., Reilly, J., Aros-Vera, F., Yushimito, W., & Isa, J. (2012). Park-and-ride facilities in New York City, economic analyses of alternative locations. *Transportation Research Records, 2276*, 123–130.
Hotelling, H. (1929). Stability in competition. *Economic Journal, 39*, 41–57.
Huff, D. L. (1964). Defining and estimating a trade area. *Journal of Marketing, 28*, 34–38.
Johnson, S. G. (2014). The NLopt nonlinear-optimization package. http://ab-initio.mit.edu/nlopt. Accessed 01.08.15.
McFadden, D. (1973). Conditional logit analysis of qualitative choice behavior. *Frontiers in Econometrics (Economic theory and mathematical economics)* (pp. 105–142). New York: Academic Press.
New York Metropolitan Transportation Council (2009). NYMTC best practice model. http://www.nymtc.org/project/bpm/bpmindex.html. Accessed 30.06.13.
Reilly, W. (1931). *The Law of Retail Gravitation*. New York: Knickerbocker Press.
ReVelle, C. (1986). The maximum capture or "sphere of influence" location problem: Hotelling revisited on a network. *Journal of Regional Science, 26*, 343–358.
Slater, P. (1975). Maximum facility location. *Journal of Research of the National Bureau of Standards: B, Mathematical Sciences, 79*.
Svanberg, K. (2002). A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal of Optimization, 12*, 555–573.
Zhang, Y., Berman, O., & Verter, V. (2012). The impact of client choice on preventive healthcare facility network design. *OR Spectrum, 34*, 349–370.