A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method

Jordi Castro
Universitat Politècnica de Catalunya
jordi.castro@upc.edu

Stefano Nasini
IESE Business School
University of Navarra
snasini@iese.edu

Francisco Saldanha-da-Gama
Faculdade de Ciências
Universidade de Lisboa
faconceicao@fc.ul.pt

Research Report UPC-DEIO DR 2015-01
June 2015

Report available from http://www-eio.upc.es/~jcastro

A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method

Jordi Castro · Stefano Nasini · Francisco Saldanha-da-Gama

Abstract We propose a cutting-plane approach (namely, Benders decomposition) for a class of capacitated multi-period facility location problems. The novelty of this approach lies on the use of a specialized interior-point method for solving the Benders subproblems. The primal block-angular structure of the resulting linear optimization problems is exploited by the interior-point method, allowing the (either exact or inexact) efficient solution of large instances. The effect of different modeling conditions and problem specifications on the computational performance are also investigated both theoretically and empirically, providing a deeper understanding of the significant factors influencing the overall efficiency of the cutting-plane method. This approach allowed the solution of instances of up to 200 potential locations, one million customers and three periods, resulting in mixed integer linear optimization problems of up to 600 binary and 600 millions of continuous variables. Those problems were solved by the specialized approach in less than one hour, outperforming other state-of-the-art methods, which exhausted the (144 Gigabytes of) available memory in the largest instances.

Keywords mixed integer linear optimization · interior-point methods · multi-period facility location · cutting planes · Benders decomposition · large-scale optimization

Mathematics Subject Classification (2000) 90C06 · 90C11 · 90C51 · 90B80

Jordi Castro*
Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Jordi Girona 1–3, 08034 Barcelona, Catalonia, Spain. E-mail: jordi.castro@upc.edu

Stefano Nasini
Dept. of Production, Technology and Operations Management, IESE Business School, University of Navarra, Av. Pearson 21, 08034 Barcelona, Catalonia, Spain. E-mail: snasini@iese.edu

Francisco Saldanha-da-Gama
Dept. of Statistics and Operations Research/Operations Research Center, Faculdade de Ciências, Universidade de Lisboa, Campo Grande 1749-016 Lisboa, Portugal. E-mail: faconceicao@fc.ul.pt

1 Introduction

A dynamic facility location problem consists of defining a time-dependent plan for locating a set of facilities in order to serve customers in some area or region. A finite planning horizon is usually considered representing the time for which the decision maker wishes plan. In a multi-period setting, the planning horizon is divided into several time periods each of which defining specific moments for making adjustments into the system. The most common goal is the minimization of the total cost—for the entire planning horizon—associated with the operation of the facilities and the satisfaction of the demand.

This class of problems extends their static counterparts and emerges as appropriate when changes in the parameters underlying a facility location problem can be predicted (e.g., demands or transportation costs). The reader can refer to the book chapter [21] for further details as well as for references in this topic.

The study of multi-period facility location problems is far from new. Nevertheless, the relevance of these problems is still quite notable which is explained by the fact that they are often at the core of more complex problems such as those arising in logistics (see, e.g., [18, 3]). Accordingly, their study is of great relevance. In particular, having efficient approaches for tackling them may render an important contribution to the resolution of more comprehensive problems.

The purpose of this paper is to introduce a method for the exact resolution of a class of large multi-period discrete facility location problems. In particular, we consider a pure phase-in setting in which a plan is to be devised for progressively locating a set of capacitated facilities over time. This is the “natural" extension to a multi-period context of the classical capacitated facility location problem. In addition, we specify a maximum number of facilities that can be operating in each time period. This is a means to control the “speed" at which the system changes in case the decision maker finds this necessary. A set of customers whose demand is known for every period is to be supplied from the operating facilities in every period. Nevertheless, we assume that service level is not necessarily 100%; instead, this will be an outcome of the problem. A cost is assumed for shortages at the customers. This cost may represent an opportunity loss cost or simply a penalty paid for the shortage. In addition to this cost, we consider operating costs for the facilities and transportation costs from the facilities to the customers. All costs are assumed to be time-dependent. The goal of the problem is to decide where and when to locate facilities in order to minimize the total cost over the planning horizon.

In the next section we show that the above problem is quite general in the sense that it allows capturing in a single modeling framework several particular cases with practical relevance and thus it actually represents a class of problems. Moreover, we show that the problem can be formulated as a mixed integer linear optimization problem with a set of binary variables (associated with the location decisions) and a set of continuous variables (associated with transportation for demand satisfaction and shortage at the customers). Such type of problems are well-known to be particularly suited for decomposition approaches based on cutting planes, namely Benders decomposition [15, 25]. In fact once the binary variables are fixed, the remaining prob-

lem is a linear optimization problem which can be dualized for deriving optimality cuts.

In this paper, we propose an efficient Benders decomposition approach for the problem above described (and thus for the class of problems it represents). The novelty in this Benders decomposition has to do with the resolution of the Benders subproblem, for which the specialized interior-point method for primal block-angular structures of [5, 6, 8] will be customized. In short, this is a primal-dual path-following method [26], whose efficiency relies on the sensible combination of Cholesky factorization and preconditioned conjugate gradient for the solution of the linear system of equations to be solved at each interior-point iteration. It is worth noting that interior-points methods have already been used in the past for the solution of integer optimization problem using cutting-plane approaches, such as in [19] for linear ordering problems. More recently, primal-dual interior-point methods have shown to be very efficient in the stabilization of column-generation procedures for the solution of vehicle routing with time windows, cutting stock, and capacitated lot sizing problems with setup times [11, 20].

Using such a specialized interior-point method for the Benders subproblems has two main benefits. The first is the ability to efficiently solve very large linear subproblems. As it will be shown, both theoretically and empirically, the specialized interior-point method becomes very efficient when the number of facilities and customers is large. This allows the effective solution of fine-grained situations, for instance, those with a few facilities and millions of customers (e.g., warehouses or post offices delivering items to households). Second, Benders decomposition does not require an optimal solution to the subproblem, and a primal-dual feasible solution (i.e., a point of the primal-dual space which is feasible for both the primal and dual pair of the subproblem) is enough for generating an additional cut. The interior-point method is thus an excellent choice, since it can quickly obtain such a primal-dual feasible point during the earlier iterations, skipping the last ones which focus on reducing the complementarity gap. In particular, avoiding the last interior-point iterations is instrumental for the specialized algorithm considered, since the performance of the embedded preconditioned conjugate gradient solver degrades close to the optimal solution. Using this approach we were able to solve multi-period capacitated facility location problems of up to 200 potential locations, one million customers and three periods, resulting in mixed integer linear problems of up to 600 binary and 600 millions of continuous variables. To the best of the authors' knowledge, facility location instances of such sizes have never been solved before in the literature.

The remainder of this paper is organized as follows. In Section 2 the problem is described in detail and formulated. In Section 3 the new decomposition approach is introduced and in Section 4 it is tested computationally. The paper ends with some conclusions drawn from the research done highlighting the main findings in this work.

2 Problem description and formulation

We consider a set of potential locations where facilities can be set operating during a planning horizon divided into several time periods. Additionally, there is a set of

customers whose demand in each period is known and that are to be supplied from the operating facilities. Facilities are capacitated and once installed they should stay opened until the end of the planning horizon. We specify the maximum number of facilities that can be operating in each time period. Finally, demands are not required to be fully satisfied; instead, we consider a service level that will be an outcome of the decision making process. We consider costs associated with (i) the operation of the facilities, (ii) the satisfaction of the demand and (iii) the shortages at the customers. The goal is to decide where facilities should be set operating and how to supply the customers in each time period from the operating facilities in order to minimize the cost for the entire planning horizon.

Before presenting an optimization model for this problem we introduce some notation that will be considered hereafter.

Sets:

$\mathscr{T}$	Set of time periods in the planning horizon with $k = \|\mathscr{T}\|$.
$\mathscr{I}$	Set of candidate locations for the facilities with $n = \|\mathscr{I}\|$.
$\mathscr{J}$	Set of customers with $m = \|\mathscr{J}\|$.

Costs:

f_i^t	Cost for operating a facility at $i \in \mathscr{I}$ in period $t \in \mathscr{T}$.
c_{ij}^t	Unitary transportation cost from facility $i \in \mathscr{I}$ to customer $j \in \mathscr{J}$ in period $t \in \mathscr{T}$.
h_j^t	Unitary shortage cost at customer $j \in \mathscr{J}$ in period $t \in \mathscr{T}$.

Other parameters:

d_j^t	Demand of customer $j \in \mathscr{J}$ in period $t \in \mathscr{T}$.
q_i	Capacity of a facility operating at $i \in \mathscr{I}$.
p^t	Maximum number of facilities that can be operating in period $t \in \mathscr{T}$.

The decisions to be made can be represented by the following sets of decision variables:

$$y_i^t = \begin{cases} 1 & \text{if a facility is operating at } i \in \mathscr{I} \text{ in period } t \in \mathscr{T}, \\ 0 & \text{otherwise.} \end{cases}$$

$x_{ij}^t =$ Amount shipped from facility $i \in \mathscr{I}$ to customer $j \in \mathscr{J}$ in period $t \in \mathscr{T}$.

$z_j^t =$ Shortage at customer $j \in \mathscr{J}$ in period $t \in \mathscr{T}$.

The multi-period facility location problem that we are working with can be formulated as follows:

$$\text{minimize} \quad \sum_{t\in\mathscr{T}}\left(\sum_{i\in\mathscr{I}} f_i^t y_i^t + \sum_{i\in\mathscr{I}}\sum_{j\in\mathscr{J}} c_{ij}^t x_{ij} + \sum_{j\in\mathscr{J}} h_j^t z_j^t\right), \tag{1}$$

$$\text{subject to} \quad \sum_{i\in\mathscr{I}} x_{ij}^t + z_j^t = d_j^t, \qquad t\in\mathscr{T},\ j\in\mathscr{J}, \tag{2}$$

$$\sum_{j\in\mathscr{J}} x_{ij}^t \le q_i y_i^t, \qquad t\in\mathscr{T},\ i\in\mathscr{I}, \tag{3}$$

$$\sum_{i\in\mathscr{I}} y_i^t \le p^t, \qquad t\in\mathscr{T}, \tag{4}$$

$$y_i^t \le y_i^{t+1}, \qquad t\in\mathscr{T}\setminus\{k\},\ i\in\mathscr{I}, \tag{5}$$

$$y_i^t \in \{0,1\}, \qquad t\in\mathscr{T},\ i\in\mathscr{I}, \tag{6}$$

$$x_{ij}^t \ge 0, \qquad t\in\mathscr{T},\ i\in\mathscr{I},\ j\in\mathscr{J} \tag{7}$$

$$z_j^t \ge 0, \qquad t\in\mathscr{T},\ j\in\mathscr{J}. \tag{8}$$

In the above model, the objective function (1) represents the total cost throughout the planning horizon, which includes the cost for operating the facilities, the transportation costs from facilities to customers and the costs for shortages at the customers. Constraints (2) ensure that the demand of each customer in each period is divided into two parts: the amount supplied from the operating facilities and the shortage. Inequalities (3) are the capacity constraints for the operating facilities. Constraints (4) define the maximum number of facilities that can be operating in each period. Relations (5) ensure that we are working under a pure phase-in setting, i.e., once installed, a facility should remain opened until the end of the planning horizon. Finally, constraints (6)–(8) define the domain of the decision variables.

The above model has several features which are worth emphasizing.

i) By considering constraints (5) we are capturing a feature of major relevance in many logistics network design problems which has to do with the need for progressively install a system since it is often the case that such systems cannot be setup in a single step (the reader can refer to [17] for a deeper discussion).
ii) Since the facilities involved in this problem are capacitated, the possibility of adjusting the set of operating facilities over time is a way for adjusting the overall capacity of the system as a response to changes in demands and costs. Some authors have explicitly considered capacity adjustments as part of the decision making process (e.g., [12, 13]) within a multi-period modeling framework for facility location problems.
iii) By specifying the values of p^t, $t\in\mathscr{T}$ we are setting a maximum "speed" for making adjustments in the system in terms of the operating facilities. When such a feature is not relevant, we can simply consider $p^t = n$, $t\in\mathscr{T}$ and the model is still valid. Since we are working with a pure phase-in problem we assume that $1 \le p^1 \le p^2 \le \cdots \le p^k \le n$.
iv) In the model proposed, the service level is not necessarily 100%; instead, it will be an outcome from the model and will result from a tradeoff between the different

costs considered. The relevance of considering a service level below 100% within the context of facility location has been discussed by several authors, such as [22], [2], and [1]. Since we are working with a multi-period problem, “service level" may be looked at as a vague term. In fact, we may consider a service level per time period or even a global service level for the planning horizon:

$$\mathrm{SL}(t) = \frac{\sum_{j\in\mathscr{J}}\sum_{i\in\mathscr{I}} x_{ij}^t}{\sum_{j\in\mathscr{J}} d_j^t} \quad , \quad \mathrm{GSL} = \frac{\sum_{t\in\mathscr{T}}\sum_{j\in\mathscr{J}}\sum_{i\in\mathscr{I}} x_{ij}^t}{\sum_{t\in\mathscr{T}}\sum_{j\in\mathscr{J}} d_j^t}.$$

In the first case, in order to obtain a “global" service level, we may simply average the service level attained in the different periods yielding

$$\mathrm{ASL} = \frac{1}{k}\sum_{t\in\mathscr{T}} \mathrm{SL}(t).$$

v) The above model is still valid if some facilities are already operating before the planning horizon and the goal is not to built a system from scratch but simply expanding a system that is already operating. In such a case we can use the same model if we fix to 1 the location variables associated with the existing facilities.

vi) In order to present a model that is as general as possible, we are assuming all parameters to be time-dependent. However, in practice this is not always the case. For instance, when the transportation costs are a function of the distance between the facilities and customers we may not observe a significant change from one period to the following and thus we may assume them to be time-invariant.

Considering the problem with $k = 1$ (one period), $p_1 = n$ and shortage costs arbitrarily large (thus ensuring that all z-variables are equal to 0), we obtain the well-known capacitated facility location problem which generalizes the uncapacitated facility location problem that is known to be NP-hard (see, e.g., [10]). Accordingly, the problem we are investigating is also NP-hard. Nevertheless, developing efficient exact approaches that can solve instances with a realistic size is always a possibility worth exploring. This is what we propose next.

3 The cutting-plane approach

The problem described in the previous section is a good candidate for the application of a Benders decomposition approach [4, 14, 23, 25]. In fact, once the binary y-variables are decided, the remaining problem is a linear optimization problem. Therefore, the problem can be projected onto the y-variables space yielding

$$\text{minimize} \quad \sum_{t\in\mathscr{T}}\sum_{i\in\mathscr{I}} f_i^t y_i^t + Q(y), \tag{9}$$

$$\text{subject to} \quad \sum_{i\in\mathscr{I}} y_i^t \le p^t, \qquad t\in\mathscr{T}, \tag{10}$$

$$y_i^t \le y_i^{t+1}, \qquad t\in\mathscr{T}\setminus\{k\}, i\in\mathscr{I}, \tag{11}$$

$$y_i^t \in \{0,1\}, \qquad t\in\mathscr{T}, i\in\mathscr{I}, \tag{12}$$

where $y = (y_i^t,\, i \in \mathscr{I},\, t \in \mathscr{T})$, and $Q(y)$ is defined as

$$Q(y) = \quad \min \quad \sum_{t\in\mathscr{T}} \left(\sum_{j\in\mathscr{J}} \sum_{i\in\mathscr{I}} c_{ij}^t x_{ij}^t + \sum_{j\in\mathscr{J}} h_j^t z_j^t \right), \tag{13}$$

$$\text{subject to} \quad \sum_{i\in\mathscr{I}} x_{ij}^t + z_j^t = d_j^t, \qquad t \in \mathscr{T},\, j \in \mathscr{J}, \tag{14}$$

$$\sum_{j\in\mathscr{J}} x_{ij}^t \le q_i y_i^t, \qquad t \in \mathscr{T},\, i \in \mathscr{I}, \tag{15}$$

$$x_{ij}^t \ge 0, \qquad t \in \mathscr{T},\, i \in \mathscr{I},\, j \in \mathscr{J}, \tag{16}$$

$$z_j^t \ge 0, \qquad t \in \mathscr{T},\, j \in \mathscr{J}. \tag{17}$$

$Q(y)$ is a convex piecewise linear function, so the overall problem can be solved by some nondifferentiable cutting-plane approach. Benders decomposition can be seen as a particular implementation of such an approach, where $Q(y)$ is approximated from below by cutting planes. These planes are obtained by evaluating $Q(y)$ at some particular y values, i.e., solving the (Benders) subproblem induced by those values. The new cuts replace $Q(y)$ and are sequentially added to (9)–(12) leading to an updated (Benders) master problem. Benders master and subproblem provide, respectively, lower and upper bounds to the optimal solution. Such a cutting-plane algorithm is iterated until the gap between the lower and upper bound is small enough or zero.

Fixing the location variables y_i^t ($i \in \mathscr{I}$, $t \in \mathscr{T}$), the linear optimization problem $Q(y)$ is separable for time periods. A resulting family of k independent linear optimization problems is obtained, which for a particular time period $t = 1,\ldots,k$ can be written as:

$$\text{SubLP}(y,t) = \quad \min \quad \sum_{j\in\mathscr{J}} \sum_{i\in\mathscr{I}} c_{ij}^t x_{ij}^t + \sum_{j\in\mathscr{J}} h_j^t z_j^t, \tag{18}$$

$$\text{subject to} \quad \sum_{i\in\mathscr{I}} x_{ij}^t + z_j^t = d_j^t, \qquad j \in \mathscr{J}, \tag{19}$$

$$\sum_{j\in\mathscr{J}} x_{ij}^t \le q_i y_i^t, \qquad i \in \mathscr{I}, \tag{20}$$

$$x_{ij}^t \ge 0, \qquad i \in \mathscr{I},\, j \in \mathscr{J}, \tag{21}$$

$$z_j^t \ge 0, \qquad j \in \mathscr{J}. \tag{22}$$

Therefore, the Benders subproblem can be written as $Q(y) = \sum_{t\in\mathscr{T}} \text{SubLP}(y,t)$. Its optimal solution provides the information about the goodness of the designed location decisions. That solution provides an upper bound to the original multi-period problem (1)–(8). It is worth to note that, in theory, a primal-dual feasible suboptimal solution to (18)–(22)—that is, an inexact solution to the subproblem, or an inexact Benders cut—is enough for the Benders decomposition algorithm, though the upper bound obtained may be higher, thus of worse quality. Inexact cuts have been studied and proven to guarantee convergence of the Benders method, for instance, in [27] for linear problems. Mixed integer linear problems can be seen as a simpler case, since the number of solutions of the Benders master problem is discrete and very often

finite. To the best of the authors' knowledge, the few references existing in the literature exploring the use of inexact cuts for mixed integer linear problems are very recent, namely [16, 24].

For each $t \in \mathscr{T}$, let $\hat{x}_{ij}^t$ and $\hat{z}_j^t$ ($i \in \mathscr{I}$, $j \in \mathscr{J}$) be the optimal (or suboptimal feasible) values of the primal variables involved in SubLP(y,t) and let λ_j^t and μ_{ij}^t ($i \in \mathscr{I}$, $j \in \mathscr{J}$) be the corresponding (feasible) dual variables associated with (19) and (20), respectively. The idea underlying Benders decomposition is to use a cutting-plane method to transfer the information about the goodness of the location decisions specified by the y-variables from the subproblem to the master problem. This master problem involves the dual variables of the subproblem as well as the original binary variables. Considering an aggregation of all the cuts generated by SubLP(y,t) for $t \in \mathscr{T}$, we can write the master problem as follows (see, e.g., [4] for further details):

$$\text{minimize} \quad \sum_{t\in\mathscr{T}} \sum_{i\in\mathscr{I}} f_i^t y_i^t + \theta, \tag{23}$$

$$\text{subject to} \quad \theta \geq \sum_{t\in\mathscr{T}} \sum_{j\in\mathscr{J}} \lambda_j^t d_j^t + \sum_{t\in\mathscr{T}} \sum_{i\in\mathscr{I}} \mu_i^t q_i y_i^t, \tag{24}$$

$$\sum_{i\in\mathscr{I}} y_i^t \leq p^t, \qquad t \in \mathscr{T}, \tag{25}$$

$$y_i^t \leq y_i^{t+1}, \qquad t \in \mathscr{T}\setminus\{k\}, i \in \mathscr{I}, \tag{26}$$

$$y_i^t \in \{0,1\}, \qquad t \in \mathscr{T}, i \in \mathscr{I}. \tag{27}$$

The objective function of the master problem provides a lower bound to the original multi-period problem (1)–(8). Cuts (24) can also be disaggregated by considering one for each time period $t \in \mathscr{T}$, of the form $\theta^t \geq \sum_{j\in\mathscr{J}} \lambda_j^t d_j^t + \sum_{i\in\mathscr{I}} \mu_i^t q_i y_i^t$, and considering the objective function $\sum_{i\in\mathscr{I}} f_i^t y_i^t + \sum_{t\in\mathscr{T}} \theta^t$. In this work we considered aggregated cuts, since some preliminary computational experiments showed that this reduces significantly the size of the master problem, yet producing high quality cuts.

For the particular case of the capacitated multi-period facility location problem we are studying in this paper, the structure of the subproblem allows obtaining a deeper insight on the quality of the Benders cuts. In order to see this, consider an $\alpha-$parameterized version of the problem with $m = k = 1$ (one period and one customer), where the demand and the capacities are specified as $d = \alpha$ and $q_i = (1-\alpha)$, for $i \in \mathscr{I}$. The corresponding subproblem can be written as follows (we simplify some of the notation previously introduced since $m = k = 1$):

$$\text{SubLP}'(\alpha) = \quad \min \quad \sum_{i\in\mathscr{I}} c_i x_i + hz, \tag{28}$$

$$\text{subject to} \quad \sum_{i\in\mathscr{I}} x_i + z = \alpha, \tag{29}$$

$$x_i \leq (1-\alpha) y_i, \qquad i \in \mathscr{I}, \tag{30}$$

$$x_i \geq 0, \qquad i \in \mathscr{I}, \tag{31}$$

$$z \geq 0. \tag{32}$$

Denote by λ, μ_i ($i \in \mathscr{I}$), ν_i ($i \in \mathscr{I}$), and γ the dual variables associated with constraints (29), (30), (31), and (32), respectively.

Proposition 1 *In a Benders iteration, let $\mathscr{I}_A$ be the subset of $\mathscr{I}$ associated to the active constraints $x_i = (1-\alpha)y_i$ of SubLP$'(\alpha)$. The corresponding Benders cut is*

$$\theta \geq \alpha(h-\gamma) + (1-\alpha)\sum_{i\in\mathscr{I}_A}(h-c_i-\gamma)\,y_i. \tag{33}$$

Proof The dual feasibility of SubLP$'(\alpha)$ implies $\lambda - \mu_i + \nu_i = c_i$, for $i \in \mathscr{I}$, and $\lambda + \gamma = h$. Note that, $\mu_i = h - c_i - \gamma$, for all $i \in \mathscr{I}_A$, and $\mu_i = 0$, for all $i \in \mathscr{I} \setminus \mathscr{I}_A$. (In the special case when $y_i = 0$, either ν_i or μ_i can be arbitrarily fixed, and this relation still holds.) Based on (24), we have:

$$\begin{aligned}\theta &\geq \alpha\lambda + (1-\alpha)\sum_{i\in\mathscr{I}}\mu_i y_i \\ &= \alpha(h-\gamma) + (1-\alpha)\sum_{i\in\mathscr{I}_A}(h-c_i-\gamma)\,y_i\end{aligned}$$

□

Proposition 1 suggests two important elements which might substantially effect the goodness of a Benders cut: (i) the relationship between demand and total capacity, captured by α, (ii) the shortage cost h. When h is small enough, $z > 0$ and $\gamma = 0$, so that $\theta \geq \alpha h + (1-\alpha)\sum_{i\in\mathscr{I}_A}(h-c_i)\,y_i$. In particular, when $h < c_i$, for all $i = 1\ldots n$, the Benders cut is $\theta \geq \alpha h$, since $|\mathscr{I}_A| = 0$. Similarly, when α approaches either zero (the total capacity widely exceeds the demand) or one (the demand overcomes the total capacity), the two limit cases reduce to $\theta \geq \sum_{i\in\mathscr{I}_A}(h-c_i-\gamma)\,y_i$ or $\theta \geq \alpha(h-\gamma)$ respectively. It turns out that the information transmitted by the Benders cut reduces when the demand grows large with respect to the total capacity, as reflected by the smaller size of the term $(1-\alpha)\sum_{i\in\mathscr{I}_A}(h-c_i-\gamma)\,y_i$. Nonetheless, when the demand is too small $|\mathscr{I}_A| = 0$ and $(1-\alpha)\sum_{i\in\mathscr{I}_A}(h-c_i-\gamma)\,y_i = 0$. Thus, both cases give rise to conditions where the decisions of the subproblem poorly affect the decision to be made in the master problem.

3.1 Solving the subproblem by a specialized interior-point method

As mentioned above, the Benders subproblem can be decomposed into k independent linear optimization problems (18)–(22). For each $t \in \mathscr{T}$, (18)–(22) can be written as the following linear problem with primal block-angular constraints:

$$\text{SubLP}(y,t) = \quad \min \quad \sum_{j\in\mathscr{J}} \mathbf{c}_j^{t\top}\mathbf{x}_j^t \tag{34}$$

$$\text{subject to} \quad \begin{bmatrix} e^\top & & & \\ & e^\top & & \\ & & \ddots & \\ & & & e^\top \\ L & L & \ldots & L \;\; I \end{bmatrix} \begin{bmatrix} \mathbf{x}_1^t \\ \mathbf{x}_2^t \\ \vdots \\ \mathbf{x}_m^t \\ \mathbf{x}_0^t \end{bmatrix} = \begin{bmatrix} d_1^t \\ d_2^t \\ \vdots \\ d_m^t \\ \mathbf{q}^t \end{bmatrix} \tag{35}$$

$$\mathbf{x}_j^t \geq 0, \quad j = 0,1,\ldots,m, \tag{36}$$

where matrix $L = [I \mid \mathbf{0}] \in \mathbb{R}^{n\times(n+1)}$ is made up by an identity matrix with a zero column vector on the right; for each $j \in \mathscr{J}$, $\mathbf{c}_j^t = [c_{1j}^t, \ldots, c_{nj}^t, h_j^t]^\top \in \mathbb{R}^{n+1}$ and $\mathbf{x}_j^t = [x_{1j}^t, \ldots, x_{nj}^t, z_j^t]^\top \in \mathbb{R}^{n+1}$ represent, respectively, the shipping and shortage costs involving customer j and the amount of commodity shipped to and shortage of customer j; $e \in \mathbb{R}^{n+1}$ is a vector of ones; $\mathbf{x}_0^t \in \mathbb{R}^n$ are the slacks of the linking constraints; $\mathbf{q}^t = [q_1 y_1^t, \ldots, q_n y_n^t]^\top \in \mathbb{R}^n$ is the right-hand side vector for the linking constraints and contains the supply capacities of the designed locations. Note that the block constraints $e^\top \mathbf{x}_j^t = d_j^t$, $j \in \mathscr{J}$, correspond to (19), whereas the linking constraints $\sum_{j\in\mathscr{J}} L\mathbf{x}_j^t + \mathbf{x}_0^t = \mathbf{q}^t$ are (20).

Formulation (34)–(36) exhibits a primal block-angular structure, and thus it can solved by the interior-point method of [5, 8]. This method is a specialized primal-dual path-following algorithm tailored for primal block-angular problems. A thorough description of primal-dual path-following algorithms can be found in [26]. Shortly, these kind of methods follow the central path until they reach the optimal solution. The central path is derived as follows. Formulation (34)–(36) can be written in standard form as

$$\begin{aligned} \text{minimize} \quad & c^\top x, && (37)\\ \text{subject to} \quad & Ax = b, && (38)\\ & x \geq 0, && (39) \end{aligned}$$

where $c, x \in \mathbb{R}^{(n+1)m+n}$ contain, respectively, all the cost and decision variables vectors $\mathbf{c}_j^t$, $\mathbf{x}_j^t$, and $A \in \mathbb{R}^{(m+n)\times[(n+1)m+n]}$ and $b \in \mathbb{R}^{m+n}$ are, respectively, the constraints matrix and right-hand-side vector of (34)–(36). Denoting by λ and s the Lagrange multipliers of the equalities and inequalities, and considering a parameter $\mu > 0$, the perturbed Karush-Kuhn-Tucker optimality conditions of (37)–(39) are

$$\begin{aligned} Ax &= b && (40)\\ A^\top \lambda + z &= c && (41)\\ XS &= \mu e, \quad (x,z) \geq 0 && (42) \end{aligned}$$

where e is a vector of ones, and X and S diagonal matrices whose diagonal entries are those of x and s. The set of unique solutions of (40)–(42) for each μ is known as the central path, and these solutions converge to those of (37)–(39) when $\mu \to 0$ (see [26]).

Each iteration of a primal-dual path-following method requires the solution of the normal equations system $A\Theta A^\top \Delta\lambda = g$, where $\Theta = XS^{-1}$ is diagonal and directly computed from the values of the primal and dual variables at each interior-point iteration, $\Delta\lambda \in \mathbb{R}^{m+n}$ is the direction of movement for the Lagrange multipliers λ, and $g \in \mathbb{R}^{m+n}$ is some right-hand side. Solving the normal equations is the most expensive computational step of the interior-point method. Exploiting the structure of A in (35), and appropriately partitioning Θ and $\Delta\lambda$ according to the $m+1$ blocks of variables and constraints, we have

$$A\Theta A^\top \Delta\lambda = \left[\begin{array}{ccc|c} e^\top\Theta_1 e & & & e^\top\Theta_1 L^\top \\ & \ddots & & \vdots \\ & & e^\top\Theta_m e & e^\top\Theta_m L^\top \\ \hline L\Theta_1 e & \dots & L\Theta_m e & \Theta_0 + \sum_{j\in\mathscr{J}} L\Theta_j L^\top \end{array}\right] \Delta\lambda$$

$$= \left[\begin{array}{ccc|c} \mathrm{Tr}(\Theta_1) & & & \theta_1^\top \\ & \ddots & & \vdots \\ & & \mathrm{Tr}(\Theta_m) & \theta_m^\top \\ \hline \theta_1 & \dots & \theta_m & D \end{array}\right] \left[\begin{array}{c} \Delta\lambda_{1_1} \\ \vdots \\ \hline \Delta\lambda_2 \end{array}\right] = \begin{bmatrix} B & C \\ C^\top & D \end{bmatrix} \begin{bmatrix} \Delta\lambda_1 \\ \Delta\lambda_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{43}$$

where $\mathrm{Tr}(M)$ denotes the trace of matrix M, $\theta_j = [\Theta_{j_{11}}, \dots, \Theta_{j_{nn}}]^\top$, for $j \in \mathscr{J}$, and

$$D = \begin{bmatrix} \Theta_{0_{11}} + \sum\limits_{j\in\mathscr{J}} \Theta_{j_{11}} & & \\ & \ddots & \\ & & \Theta_{0_{nn}} + \sum\limits_{j\in\mathscr{J}} \Theta_{j_{nn}} \end{bmatrix} \tag{44}$$

is diagonal.

By eliminating $\Delta\lambda_1$ from the first group of equations, the system (43) reduces to

$$\left(D - C^\top B^{-1} C\right) \Delta\lambda_2 = (g_2 - C^\top B^{-1} g_1) \tag{45}$$

$$B\Delta\lambda_1 = (g_1 - C\Delta\lambda_2). \tag{46}$$

Systems with matrix B of the form $Bu = v$ (for some u and v) in (45)–(46) are directly solved as

$$u_j = \frac{v_j}{\mathrm{Tr}(\Theta_j)} \quad j = 1, \dots, m.$$

The only computational effort is thus the solution of system (45)—the Schur complement of (43)—, whose dimension is n, the number of candidate locations.

System (45) is computationally expensive if solved by Cholesky factorization. As suggested in [5]—for multicommodity flow problems—and in [6]—for general block-angular problems, this system can be solved by a preconditioned conjugate gradient. A good preconditioner is instrumental for the performance of the conjugate gradient. As shown in [5, Prop. 4], the inverse of $C^\top B^{-1} C$ for this kind of block-angular problems can be computed as

$$(D - C^\top B^{-1} C)^{-1} = \left(\sum_{i=0}^{\infty} \left(D^{-1} (C^\top B^{-1} C) \right)^i \right) D^{-1}. \tag{47}$$

The preconditioner M^{-1} is an approximation of $(D - C^\top B^{-1} C)^{-1}$ obtained by truncating the infinite power series (47) at some term ϕ. In general the best results are obtained, for $\phi = 0$ such that the preconditioner is just $M^{-1} = D^{-1}$. In that case, the

solution of (45) by the conjugate gradient only requires matrix-vector products with matrix $(D - C^\top B^{-1} C)$—computationally cheap because of the structure of D, C and B—and the solution of systems with matrix D—which are straightforward since D is diagonal.

It has been shown [8] that the quality of the preconditioner depends on the spectral radius (i.e., the maximum absolute eigenvalue) of matrix $D^{-1}(C^\top B^{-1} C)$, denoted as ρ, which is real and always in $[0,1)$. The farther from 1, the better is the preconditioner. In practice it is observed that ρ comes closer to 1 as we approach the optimal solution, degrading the performance of the conjugate gradient. Therefore, since there is no need to optimally solve the Benders subproblem, the interior-point algorithm can be prematurely stopped for some not-too-small $\mu > 0$. The suboptimal primal-dual point will guarantee the primal and dual feasibility conditions (40) and (41), and its optimality gap can be controlled through μ. This way we can avoid the most expensive conjugate gradient iterations, providing at the same time a good primal-dual feasible point to generate a new cut for the master problem. We note that this cannot be (efficiently) achieved using the simplex algorithm for the Benders subproblem, since the points are either primal feasible (primal simplex) or dual feasible (dual simplex), and primal-dual feasibility is not reached until the optimal solution.

As stated above, the dimension of the Schur complement system (45) is n, the number of candidate locations. Therefore, we can expect a high performance of this approach when the number of potential facilities is small, even if the number of customers is very large. In fact, this "few locations and many customers" situation is the most usual in practice. This assertion is supported by the empirical evidence provided in the next section, where problems of a few hundreds of locations and up to one million of customers are efficiently solved.

In addition, theoretically, the method is also very efficient when the number of candidate locations n becomes large. In this case, as shown by next proposition, in the limit, the diagonal preconditioner $M^{-1} = D^{-1}$ provides the inverse of the matrix in the Schur complement system (45). We will assume the interior-point (x,s) of the current iteration is not too close to the optimal solution, such that it can be uniformly bounded away from 0 (in fact, at every iteration the current point is known to be greater than 0 [26]).

Proposition 2 *Let us assume there is a* $0 < \varepsilon \in \mathbb{R}$ *such that the current interior-point* (x,s) *satisfies* $x > \varepsilon$ *and* $s > \varepsilon$. *Therefore, when* $n \to \infty$ *(the number of candidate locations grows larger) we have* $D - C^\top B^{-1} C \to D$.

Proof This reduces to showing that matrix $C^\top B^{-1} C \to 0$ when $n \to \infty$. From the definition of C and B in (43), since B is diagonal, we have that entry (h,l) of $C^\top B^{-1} C$ is

$$C^\top B^{-1} C_{hl} = \sum_{j=1}^{m} \frac{\Theta_{j,hh}\Theta_{j,ll}}{\sum_{i=1}^{n} \Theta_{j,ii}} \leq \frac{1}{n} \sum_{j=1}^{m} \frac{\Theta_{j,hh}\Theta_{j,ll}}{\min_i \Theta_{j,ii}}.$$

Since $\Theta_j = X_j S_j^{-1}$ and $x_j > \varepsilon > 0$ and $s_j > \varepsilon > 0$, we have that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{m} \frac{\Theta_{j,hh}\Theta_{j,ll}}{\min_i \Theta_{j,ii}} = 0.$$

□

The major implication of Proposition 2 is that for large n the number of preconditioned conjugate gradient (PCG) iterations required for the solution of (45) is very small using $M^{-1} = D^{-1}$ as preconditioner. However, this was also empirically observed when the parameter that grows larger is the number of customers, m, as shown in the next section.

For the computational tests of next section we used the solver BlockIP, a C++ efficient implementation of the above specialized interior-point method, including many additional features [7].

4 Computational tests

In this section we describe a series of computational experiments designed to empirically support the efficiency of the proposed cutting-plane approach for capacitated multi-period facility location using the specialized interior-point method for block angular problems. All the runs were carried out on a Fujitsu Primergy RX300 server with 3.33 GHz Intel Xeon X5680 CPUs (24 cores) and 144 GB of RAM, under a GNU/Linux operating system (Suse 11.4), without exploitation of multithreading capabilities. CPLEX branch-and-cut (release 12.4) has been used for the solution of the Benders master problems; Benders subproblems were solved with both the barrier algorithm of CPLEX and BlockIP. The CPLEX barrier—which will be denoted as "BarOpt"—was used since it resulted to be more efficient than simplex variants for these large subproblems.

4.1 The effect of parameter specification

Consider a capacitated multi-period facility location problem of the form (1)–(8) and the demands, capacities and costs reported in Table 1. Geometrically, this parameter specification can be interpreted as a setting where facilities and customers are collinearly distributed along a one-dimensional line and the cost for operating a facility increases along that line. The transportation costs reflect some distance measure between the facilities and the customers; the demands follow a similar increasing pattern along the line, so that more expensive locations are geometrically closer to customers with higher demand. The capacities of the $\mathscr{I}$ locations grow linearly with respect to their costs and do not vary over time. The tuning parameters α and β allow balancing the relation between the total demand and the total capacity in the system.

The computational tests performed involve 150 instances of problem (1)–(8) divided into six groups of 25 instances all generated according to the parameter specification of Table 1. Each group of 25 instances is associated with a specific combination of m, n and k. The 25 instances in each group correspond to different combinations of α and β—which have been chosen to take the values $0.1, 0,3, 0.5, 0.7$, and 0.9, resulting in 25 possible combinations. The first two groups are associated with single-period time horizons (i.e., $k = 1$); the third and fourth groups include the instances

Table 1: Parameter specification for the instances to be used in the computational tests

$f_i^t = \frac{10+1000i}{n}$	for location $i = 1\dots n$ in period $t = 1\dots k$.	The building costs do not vary over time and only depend on the specific location i.
$c_{ij}^t = \frac{10+\lvert i-j\rvert}{n+m}$	for location $i = 1\dots n$, customer $j = 1\dots m$, in period $t = 1\dots k$.	The transportation costs do not vary over time and only depend on the distance between location i and destination j.
$h_j^t = n \times m$	for customer $j = 1\dots m$, in period $t = 1\dots k$.	The unitary shortage cost at customer j is chosen to overcome the maximum building cost.
$d_j^t = (1-\alpha)t\frac{10+j}{\lvert\mathscr{T}\rvert}$	for customer $j = 1\dots\lvert\mathscr{J}\rvert$, in period $t = 1\dots\lvert\mathscr{T}\rvert$	The demands increase over time and vary in size depending on the customer.
$q_i^t = \alpha\frac{100+2i}{n}$	for location $i = 1\dots n$, in period $t = 1\dots k$.	The capacities do not vary over time and only depend on the specific location i.
$p^t = \beta n$	for period $t = 1\dots k$.	The maximum number of facilities that can be operating in period t does not vary over time.

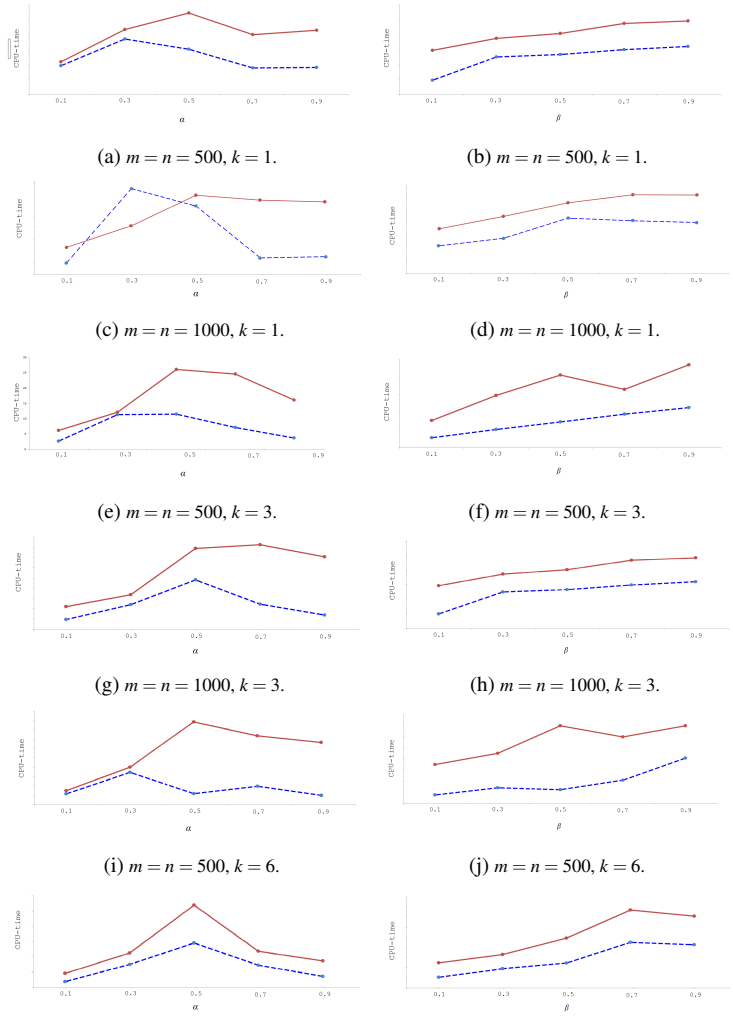
Table 2: Average CPU times for the three tables in Appendix A. The average number of Benders iterations is reported within parenthesis

						Benders decomposition		Branch-and-cut
n	m	k	const.	bin.var.	cont. var.	BarOpt	BlockIP	
500	500	1	1001	500	250500	8.1 (3.28)	4.9 (3.16)	27.3 (36950)
1000	1000	1	2001	1000	1001000	62.2 (3.52)	44.5 (3.48)	257.0 (82632)
500	500	3	3003	1500	751500	17.0 (4.92)	7.2 (4.36)	118.6 (152792)
1000	1000	3	6003	3000	3003000	115.7 (4.48)	48.0 (4.28)	1440.1 (384906)
500	500	6	6006	3000	1503000	56.1 (7.76)	19.1 (7.52)	433.8 (291345)
1000	1000	6	12006	6000	6006000	253.6 (6.40)	140.1 (6.44)	2936.0 (783983)

with a 3-period planning horizon ($k = 3$); the fifth and sixth groups corresponds to instances with a 6-period planning horizon ($k = 6$). The results are fully presented in Appendix A. All the instances have been solved by the cutting-plane algorithm above described, using both CPLEX BarOpt and BlockIP.

Figure 1 shows the CPU time (averaged over 25 instances) for each of the six groups, both for BarOpt and BlockIP. The vertical axis shows the CPU time (in seconds), whereas the horizontal axis reports the five different values of α (for the left plots) and β (for the right plots). The straightforward interpretation of these results is that, for almost all values of m, n, k, α and β, BlockIP significantly outperformed BarOpt when solving the Benders subproblems. Another interesting and relevant fact is the non-linear effect of α, which is consistent with what we claimed when discussing the implications of Proposition 1: extreme values of α are associated to poor effect of the second stage decision and transportation costs (subproblem solution) upon the goodness of the first stage decisions (master problem solution).

The aggregated results for the six groups of instances (averaged over 25 single problems) are reported in Table 2. In addition to the values of n, m and k, the table reports the number of constraints ("const."), binary variables ("bin. var") and continuous variables ("cont. var") of the resulting optimization problems. Columns "BarOPt" and "BlockIP" report the average CPU time and, within parentheses, the average number of Benders iterations for the subproblems. The column "Branch-and-cut" reports the average CPU time (seconds) and the average number of simplex iterations required by the CPLEX branch-and-cut solver for the solution of the



(a) $m = n = 500$, $k = 1$. (b) $m = n = 500$, $k = 1$.

(c) $m = n = 1000$, $k = 1$. (d) $m = n = 1000$, $k = 1$.

(e) $m = n = 500$, $k = 3$. (f) $m = n = 500$, $k = 3$.

(g) $m = n = 1000$, $k = 3$. (h) $m = n = 1000$, $k = 3$.

(i) $m = n = 500$, $k = 6$. (j) $m = n = 500$, $k = 6$.

(k) $m = n = 1000$, $k = 6$. (l) $m = n = 1000$, $k = 6$.

Fig. 1: Comparisons of the CPU times of Benders–with–BarOpt (red line) and Benders–with–BlockIP (blue dashed line) for different values of α and β, corresponding to the parameter specification of Table 1

monolithic formulation (1)–(8). It should be noted that the larger the problems, the more efficient the cutting-plane method—with either BarOpt or BlockIP—compared to branch-and-cut. BlockIP seems to be approximately twice faster than BarOpt for all the problem sizes.

Since the CPU times have been obtained for different combinations of α, β, n, m and k, a full factorial experiment was performed allowing the estimation of the effect of each parameter on the CPU time and on the number of Benders iterations. A linear regression has been applied to the collection of 150 numeric observations reported in

Table 3: Linear regression of Benders iterations and CPU time

	Iterations		CPU	
factor	effect	p-value	effect	p-value
intercept	3.19E-16	1.00000	-9.15E-17	1.00000
$\|\alpha - 0.5\|$	−0.20418	0.00428	-0.31078	3.26E-06
β	0.19115	0.00739	0.23008	0.00046
$m = n$	−0.04309	0.54113	0.39638	6.30E-09
k	0.44965	2.11E-09	0.30965	3.52E-06

Table 4: Linear regression of simplex iterations and CPU time

	Iterations		CPU	
factor	effect	p-value	effect	p-value
intercept	-1.05E-16	1.00000	1.65E-16	1.00000
$\|\alpha - 0.5\|$	-0.03690	0.34926	0.19710	0.00895
β	-0.188437	3.98E-06	-0.24829	0.00011
$m = n$	0.472138	1.63E-23	0.43848	7.98E-11
k	0.718540	1.77E-39	0.37580	1.39E-08

Appendix A. The two response variables are given by the CPU time and either the number of Benders iterations—for Table 3—or the number of simplex iterations—for Table 4. Based on the non-linear effect of α, observed in Figure 1, the regression model includes the linear effect $|\alpha - 0.5|$ (which is related to the excess of demand or excess of capacities), rather than α.

From Table 3 we conclude that the length of the planning horizon is the main responsible for the number of Benders iterations (0.44965), but its effect is comparatively reduced when the CPU time is taken into account. This is consistent with the fact that the size of the subproblems per each time period is exclusively determined by the number of potential locations and customers and this is the reason why the effect of m and n plays the strongest role (0.39638). Another interesting insight that can be deduced from the regression analysis performed is the fact that the excess of demand or capacities (captured by parameter $|\alpha - 0.5|$) gives rise to two different outcomes in the computational performance of the Benders decomposition and the branch-and-cut algorithm. In fact, reinterpreting Proposition 1, high values of $|\alpha - 0.5|$ should result in a poor dependency between the second stage and first stage decisions. Clearly, the same reasoning does not apply to the branch-and-cut algorithm, whose generation of valid inequalities follow a completely different logic.

4.2 Solution of very large-scale instances

In addition to the instances analysed in the previous section, we generated a set of 18 very large-scale instances to test the efficiency of the cutting-plane approach using the specialized interior-point algorithm for the subproblems. These additional instances were obtained by considering all the combinations for $n \in \{100, 200\}$, $m \in \{100000, 500000, 1000000\}$ and $k \in \{1, 2, 3\}$. The parameters α and β were set to 0.9999 and to 1, respectively, for all the instances, to avoid problems with large

Table 5: Dimensions and results with optimality tolerance 10^{-3} for the subproblems

						BlockIP			BarOpt			
n	m	k	const.	bin.var.	cont. var.	iter.	gap	CPU	iter.	gap	CPU	rel. diff.
100	100000	1	100101	100	10100000	3	0.0010	17.35	3	0.0002	40.14	0.0000
100	100000	2	200302	200	20200000	4	0.0007	32.04	4	0.0001	117.45	0.0000
100	100000	3	300503	300	30300000	5	0.0009	63.31	6	0.0035	373.97	-0.0017
100	500000	1	500101	100	50500000	2	-0.0040	69.17		*		—
100	500000	2	1000302	200	101000000	3	0.0006	272.39		*		—
100	500000	3	1500503	300	151500000	5	0.0007	760.93		*		—
100	1000000	1	1000101	100	101000000	2	-0.0002	123.53	2	0.0008	655.72	-0.0003
100	1000000	2	2000302	200	202000000	2	-0.0010	312.23		†		—
100	1000000	3	3000503	300	303000000	3	0.0009	891.81		†		—
200	100000	1	100201	200	20100000	3	0.0010	17.39	3	0.0002	100.54	0.0000
200	100000	2	200602	400	40200000	4	0.0007	32.09	4	0.0001	297.03	0.0000
200	100000	3	301003	600	60300000	5	0.0009	63.55	6	0.0035	912.05	-0.0017
200	500000	1	500201	200	100500000	3	0.0010	110.02	3	0.0000	1146.60	0.0001
200	500000	2	1000602	400	201000000	4	0.0010	309.68		‡		—
200	500000	3	1501003	600	301500000	6	0.0010	868.43		†		—
200	1000000	1	1000201	200	201000000	3	0.0009	729.79		†		—
200	1000000	2	2000602	400	402000000	4	0.0010	1109.64		†		—
200	1000000	3	3001003	600	603000000	6	0.0009	3254.21		†		—

* Repeated solution in master, Benders would not converge
† CPLEX ran out of memory (required more than 144 Gigabytes of RAM)
‡ CPLEX aborted

Table 6: Dimensions and results with optimality tolerance 10^{-5} for the subproblems

						BlockIP			BarOpt			
n	m	k	const.	bin.var.	cont. var.	iter.	gap	CPU	iter.	gap	CPU	rel. diff.
100	100000	1	100101	100	10100000	3	0.0000	25.39	3	0.0000	42.01	0.0000
100	100000	2	200302	200	20200000	4	0.0000	63.36	4	0.0000	129.91	0.0000
100	100000	3	300503	300	30300000	6	0.0000	166.86	6	0.0000	336.20	0.0000
100	500000	1	500101	100	50500000	2	0.0003	552.66	2	0.0011	474.67	-0.0004
100	500000	2	1000302	200	101000000	3	0.0000	2534.44	3	0.0004	1391.85	-0.0004
100	500000	3	1500503	300	151500000	4	0.0071	5524.19	4	0.0004	3932.30	0.0067
100	1000000	1	1000101	100	101000000	2	0.0001	1292.37	2	0.0002	1221.85	0.0000
100	1000000	2	2000302	200	202000000	2	0.0002	3124.20		†		—
100	1000000	3	3000503	300	303000000	3	0.0000	11218.75		†		—
200	100000	1	100201	200	20100000	3	0.0000	25.45	3	0.0000	102.69	0.0000
200	100000	2	200602	400	40200000	4	0.0000	63.59	4	0.0000	310.33	0.0000
200	100000	3	301003	600	60300000	6	0.0000	167.76	6	0.0000	787.70	0.0000
200	500000	1	500201	200	100500000	3	0.0000	1402.13	3	0.0000	1064.93	0.0001
200	500000	2	1000602	400	201000000	5	0.0000	5814.12		‡		—
200	500000	3	1501003	600	301500000	6	0.0000	8652.29		†		—
200	1000000	1	1000201	200	201000000		*			†		—
200	1000000	2	2000602	400	402000000	4	0.0001	14514.18		†		—
200	1000000	3	3001003	600	603000000	6	0.0001	40744.86		†		—

* Preconditioned conjugate gradient solver failed
† CPLEX ran out of memory (required more than 144 Gigabytes of RAM)
‡ Execution aborted

shortages due to lack of capacity. The dimensions of these instances are inspired by real-world location problems that may be faced, for instance, by internet-based retailer multinational companies. Such problems call for a few dozens or hundreds of locations spread around the world for the warehousing activities, and hundreds of thousands of "customers" related, for instance, to cities over some threshold population. To the best of the authors' knowledge, the resolution of facility location problems with such dimensions have never been reported in the literature.

We ran those instances twice with the cutting-plane approach, using optimality tolerances 10^{-3} and 10^{-5} for the interior-point solver in the subproblems. As stated in previous sections, inexact solutions to subproblems save the last and thus often

the most expensive interior-point iterations with BlockIP since the performance of PCG degrades near the optimal solution. For a fair comparison with the off-the-shelf solver in use, these same optimality tolerances were set for CPLEX BarOpt, although its performance should not be significantly affected by this tolerance since it does not rely on PCG. The master problems were also suboptimally solved with CPLEX branch-and-cut by using a positive optimality tolerance to avoid too expensive solutions; this tolerance was reduced at each Benders iteration multiplying it by a factor in the interval $(0,1)$ (in particular, we used 0.95). A positive small optimality gap was also used for the global Benders decomposition; Benders iterations stop when the relative difference between the best lower and upper bounds is below this tolerance.

Tables 5 and 6 report the results obtained for optimality tolerances 10^{-3} and 10^{-5} in the subproblems, respectively. The information contained in the columns "m", "n", "k", "const.", "bin. var" and "cont. var" is the same as in Table 2. Columns "iter.", "gap" and "CPU" show the number of interior-point iterations, the achieved Benders optimality gap, and the CPU time, respectively, for both BlockIP and BarOpt. In column "rel. diff" we report the relative difference in terms of the best solutions (i.e., best Benders upper bounds) obtained by BlockIP and BarOpt. A negative value indicates that the upper bound obtained with BlockIP was smaller (and thus better) than that obtained using BarOpt. Although these values are omitted in the tables, it is worth noting that, as it was advanced in Section 3.1, BlockIP required in average only two PCG iterations for the solution of system (45) in the largest instances. (Analogous results for primal block-angular problems with the form (34)–(36) have been also observed in the field of complex network problems [9].)

Some results presented in Tables in these tables 5 and 6 require an extra explanation. In Table 5 we can observe three cells marked with * which correspond to the resolution of the instances using BarOpt for solving the subproblems. This indicates that the master problem reported the same binary solution in two consecutive iterations. In such a case, the Benders subproblem would generate the same new constraint for the master and the algorithm would iterate forever. This happens due to the nonzero optimality tolerance considered, which makes the inequality generated by the (inexact) solution to Benders subproblem not really a *cut*, but only a valid inequality for the master problem. This situation only occurred in Table 5—where the subproblem optimality tolerance is 10^{-3}—, never in Table 6 when a tighter optimality tolerance was considered. It is also worth noting that only runs using BarOpt suffered from this effect.

It is also worth to note that three executions with BlockIP in Table 5 reported slightly negative gaps for the Benders method. In those cases only two Benders iterations were performed (only one master was needed, the first one is a dummy one). This is due both to the inexact solutions of both the subproblem and master. Even in that case, however, for the only instance with this effect that could also be solved by CPLEX, the upper bound obtained with Benders with BlockIP was slightly better than the one obtained with CPLEX.

Observing Table 5 we conclude that using the optimality tolerance 10^{-3} Benders decomposition with BlockIP outperformed Benders with CPLEX BarOpt in all the instances. When using the smaller tolerance 10^{-5}, BlockIP was not so competitive for the smaller instances, as it can be observed in Table 6; in fact, the last interior-point

iterations were very time consuming due to the use of PCG. However, the subproblem solutions reported were slightly better with this tighter tolerance, as we conclude from the better gaps reported in Table 6. In spite of this worse performance with the tighter tolerance, Benders with BlockIP was able to provide a good solution to the largest instances, while CPLEX with BarOpt run out of memory. Remarkably, Benders with BlockIP was able to solve the largest cases, namely those with a number of opened facilities equal to 181 in period 3. This means that the dimension of the subproblems solved by BlockIP was up to 181 million of continuous variables.

As for the memory requirements, from tables 5 and 6 we conclude that BlockIP is much more efficient than CPLEX with BarOpt. CPLEX exhausted the 144 Gigabytes of RAM of the computer in the largest instances (executions marked with † in those tables), while BlockIP just required a small fraction of the available memory. Indeed, Benders with BlockIP was able to solve all the instances but one among those reported in Table 6 (marked with *); in this instance the PCG solver failed and BlockIP was stopped since otherwise it would have switched to the solution of normal equations by Cholesky factorization, which happened to be computationally prohibitive (both in terms of CPU time and memory requirements). CPLEX also failed in one of the instances—$n = 200$, $m = 500000$ and $k = 2$—but for both optimality tolerances; it just aborted without any message error.

5 Conclusions

In this work we exploited the use of a specialized interior-point method for solving the Benders subproblems associated with the decomposition of large-scale capacitated multi-period discrete facility location problems. This was accomplished by taking advantage from the primal block-angular structures of the underlying constraints matrices. The computational tests performed and reported in the paper show that this led to a substantial decrease in the computational effort for the overall Benders procedure. The effect of different modeling conditions on the computational performance was also investigated, which provided a deeper understanding of the significant factors influencing the overall efficiency.

Overall, the extensive computational results reported in Section 4 show that in all the instances tested, a Benders decomposition approach embedding BlockIP clearly outperformed other approaches, such as branch-and-cut or Benders using a generic interior-point method, even when the latter makes use of the full strength of an off-the-shelf solver such as CPLEX. Furthermore, the specialized interior-point method was able to solve the Benders subproblems of the largest instances, namely, those in which the number of opened facilities in the last period was 181 and thus with subproblems involving up to 181 million of continuous variables.

The research presented in this paper opens new possibilities for solving exactly large instances of more comprehensive multi-period facility location problems, A straightforward extension that can be considered is the mixed phase-in/phase-out problem that in addition to the features considered in this paper assumes that some

facilities are already operating before the beginning of the planning horizon and that can be closed in any period.

Another challenging area in which the developments proposed in this work may have a strong impact concerns stochastic single- and multi-period discrete facility location problems.

Summing up, the cutting-plane approach proposed in this work is promising and can be now easily adapted to discrete facility location problems whose underlying constraints matrices have a block-angular structure.

Acknowledgements This work has been supported by grant MTM2012-31440 of the Spanish Ministry of Economy and Competitiveness.

References

1. M. Albareda-Sambola, E. Fernández, Y. Hinojosa, and J. Puerto. The multi-period incremental service facility location problem. *Computers & Operations Research*, 36:1356–1375, 2009.
2. M. Albareda-Sambola, A. Alonso-Ayuso, L. Escudero, E. Fernández, Y. Hinojosa, and C. Pizarro-Romero. A computational comparison of several formulations for the multi-period incremental service facility location problem. *TOP*, 18:62–80, 2010.
3. S. Alumur, B.Y. Kara, and T. Melo. Location and logistics. In G. Laporte, S. Nickel, and F. Saldanha-da-Gama, editors, *Location Science*, Berlin and Heidelberg, 2015. Springer.
4. J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Computational Management Science*, 2:3–19, 2005. English translation of the original paper appeared in Numerische Mathematik, 4:238–252, 1962.
5. J. Castro. A specialized interior-point algorithm for multicommodity network flows. *SIAM Journal on Optimization*, 10:852–877, 2000.
6. J. Castro. An interior-point approach for primal block-angular problems. *Computational optimization and Applications*, 36:195–219, 2007.
7. J. Castro. Interior-point solver for convex separable block-angular problems. *Optimization Methods and Software*, 2015. doi: 10.1080/10556788.2015.1050014.
8. J. Castro and J. Cuesta. Quadratic regularizations in an interior-point method for primal block-angular problems. *Mathematical Programming A*, 130:415–445, 2011.
9. J. Castro and S. Nasini. Mathematical programming approaches for classes of random network problems. *European Journal of Operational Research*, 245: 402–414, 2015.
10. G.P. Cornuéjols, G. L. Nemhauser, and L.A. Wolsey. The uncapacitated facility location problem. In P.B. Mirchandani and R.L. Francis, editors, *Discrete Location Theory*, New York, 1990. Wiley-Interscience.
11. J. Gondzio, P. González-Brevis, and P. Munari. New developments in the primal-dual column generation technique. *European Journal of Operational Research*, 224:41–51, 2013.

12. S. Jena, J.-F. Cordeau, and B. Gendron. Modeling and solving a logging camp location problem. *Annals of Operations Research*, 2012. doi: 10.1007/s10479-012-1278-z.
13. S. Jena, J.-F. Cordeau, and B. Gendron. Dynamic facility location with generalized modular capacity. *Transportation Science*, 2014. doi: 10.1287/trsc.2014.0575.
14. T.L. Magnanti and R.T. Wong. Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. *Operations Research*, 29:464–484, 1981.
15. T.L. Magnanti and R.T. Wong. Decomposition methods for facility location problems. In P.B. Mirchandani and R.L. Francis, editors, *Discrete Location theory*, New York, 1990. Wiley.
16. J. Malick, W. de Oliveira, and S. Zaourar. Nonsmooth optimization using uncontrolled inexact information. Technical report, INRIA Grenoble, 2013. Available from `http://www.optimization-online.org/DB_HTML/2013/05/3892.html`, submitted.
17. M.T. Melo, S. Nickel, and F. Saldanha-da-Gama. Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning. *Computers & Operations Research*, 33:181–208, 2006.
18. M.T. Melo, S. Nickel, and F. Saldanha-da-Gama. Facility location and supply chain management - A review. *European Journal of Operational Research*, 196:401–412, 2009.
19. J.E. Mitchell and B. Borchers. Solving real-world linear ordering problems using a primal-dual interior point cutting plane method. *Annals of Operations Research*, 62:253–276, 1996.
20. P. Munari and J. Gondzio. Using the primal-dual interior point algorithm within the branch-price-and-cut method. *Computers & Operations Research*, 40:2026–2036, 2013.
21. S. Nickel and F. Saldanha-da-Gama. Multi-period facility location. In G. Laporte, S. Nickel, and F. Saldanha-da-Gama, editors, *Location Science*, Berlin and Heidelberg, 2015. Springer.
22. S. Nickel, F. Saldanha-da-Gama, and H. P. Ziegler. A multi-stage stochastic supply network design problem with financial decisions and risk management. *Omega*, 40:511–524, 2012.
23. W. Rei, J.-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing*, 21:333–345, 2009.
24. W. van Ackooij, A. Frangioni, and W. de Oliveira. Inexact stabilized Benders' decomposition approaches to chance-constrained problems with finite support. Technical Report 15-01, Università di Pisa, 2015. Available from `http://www.di.unipi.it/~frangio/`, submitted.
25. P. Wentges. Accelerating Benders decomposition for the capacitated facility location problem. *Mathematical Methods of Operations Research*, 44:267–290, 1996.
26. S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philapelphia, PA, 1996.

27. G. Zakeri, A.B. Philpott, and D.M. Ryan. Inexact cuts in benders decomposition. *SIAM Journal on Optimization*, 10:643–657, 2000.

A Tables of numerical experiments

One period

Tables 7, 8 and 9 show the CPU times required by the Benders decomposition and the Branch and Cut to solve instances of (1)–(8), with one, three and six time periods respectively. The parameter specification has been defined in Table 1, with different combinations of α and β and for two sizes $m = n = 500$ and $m = n = 1000$.

Table 7: CPU time of instances of two sizes $m = n = 500$ and $m = n = 1000$, with one time period ($k = 1$). The three tables report the CPU times of the three analyzed solution methods: Benders–with–BlockIP, Benders–with–BarOpt and Branch–and–cut. The values inside the parenthesis are either the number of Benders iterations (for the first two tables) or the number of MIP simplex iterations (for the last table)

500 facility locations – 500 destinations

β, α	Benders–with–BlockIP				
	0.1	0.3	0.5	0.7	0.9
0.1	3.04 (2)	7.62 (2)	3.95 (3)	2.59 (3)	3.84 (5)
0.3	3.25 (2)	8.22 (2)	5.14 (3)	4.59 (5)	3.78 (4)
0.5	3.68 (2)	8.45 (2)	7.00 (4)	3.63 (4)	3.79 (4)
0.7	4.25 (2)	9.03 (2)	9.12 (4)	3.72 (4)	3.68 (4)
0.9	5.02 (2)	11.52 (2)	9.14 (4)	3.27 (4)	3.11 (4)

β, α	Benders–with–Baropt				
	0.1	0.3	0.5	0.7	0.9
0.1	4.52 (2)	7.85 (2)	5.84 (3)	6.06 (3)	9.60 (5)
0.3	4.59 (2)	7.76 (2)	6.79 (3)	9.61 (5)	8.68 (4)
0.5	5.35 (2)	7.90 (2)	10.71 (5)	8.26 (4)	8.39 (4)
0.7	5.86 (2)	9.11 (2)	15.88 (5)	8.15 (4)	8.25 (4)
0.9	6.08 (2)	10.93 (2)	15.52 (5)	8.09 (4)	8.23 (4)

β, α	Branch–and–cut (CPLEX)				
	0.1	0.3	0.5	0.7	0.9
0.1	24.78 (23534)	24.39 (35422)	28.20 (29337)	30.16 (46270)	29.31 (44764)
0.3	26.64 (24741)	22.83 (36596)	25.90 (32865)	28.45 (46158)	29.99 (44584)
0.5	26.41 (27408)	22.18 (36994)	26.09 (35442)	27.92 (44459)	30.07 (44584)
0.7	26.24 (27394)	22.76 (39697)	23.10 (29404)	27.91 (44459)	30.18 (44584)
0.9	26.36 (27147)	22.14 (39457)	24.02 (29404)	27.70 (44459)	31.06 (44584)

1000 facility locations – 1000 destinations

β, α	Benders–with–BlockIP				
	0.1	0.3	0.5	0.7	0.9
0.1	3.04 (2)	7.62 (2)	3.95 (3)	2.59 (3)	3.84 (5)
0.3	3.25 (2)	8.22 (2)	5.14 (3)	4.59 (5)	3.78 (4)
0.5	3.68 (2)	8.45 (2)	7.00 (4)	3.63 (4)	3.79 (4)
0.7	4.25 (2)	9.03 (2)	9.12 (4)	3.72 (4)	3.68 (4)
0.9	5.02 (2)	11.52 (2)	9.14 (4)	3.27 (4)	3.11 (4)

β, α	Benders–with–Baropt				
	0.1	0.3	0.5	0.7	0.9
0.1	4.52 (2)	7.85 (2)	5.84 (3)	6.06 (3)	9.60 (5)
0.3	4.59 (2)	7.76 (2)	6.79 (3)	9.61 (5)	8.68 (4)
0.5	5.35 (2)	7.90 (2)	10.71 (5)	8.26 (4)	8.39 (4)
0.7	5.86 (2)	9.11 (2)	15.88 (5)	8.15 (4)	8.25 (4)
0.9	6.08 (2)	10.93 (2)	15.52 (5)	8.09 (4)	8.23 (4)

β, α	Branch–and–cut (CPLEX)				
	0.1	0.3	0.5	0.7	0.9
0.1	24.78 (23534)	24.39 (35422)	28.20 (29337)	30.16 (46270)	29.31 (44764)
0.3	26.64 (24741)	22.83 (36596)	25.90 (32865)	28.45 (46158)	29.99 (44584)
0.5	26.41 (27408)	22.18 (36994)	26.09 (35442)	27.92 (44459)	30.07 (44584)
0.7	26.24 (27394)	22.76 (39697)	23.10 (29404)	27.91 (44459)	30.18 (44584)
0.9	26.36 (27147)	22.14 (39457)	24.02 (29404)	27.70 (44459)	31.06 (44584)

Three periods

Table 8: CPU time of instances of two sizes $m = n = 500$ and $m = n = 1000$, with three time periods ($k = 3$). The three tables report the CPU times of the three analyzed solution methods: Benders–with–BlockIP, Benders–with–BarOpt and Branch–and–cut. The values inside the parenthesis are either the number of Benders iterations (for the first two tables) or the number of MIP simplex iterations (for the last table)

500 facility locations – 500 destinations

β, α	Benders–with–BlockIP				
	0.1	0.3	0.5	0.7	0.9
0.1	2.41 (2)	2.40 (2)	2.45 (2)	2.67 (3)	4.22 (6)
0.3	2.49 (2)	2.72 (2)	7.19 (5)	9.88 (7)	3.58 (6)
0.5	2.57 (2)	5.81 (3)	17.65 (8)	7.05 (6)	3.57 (6)
0.7	2.71 (2)	18.32 (3)	15.10 (6)	7.97 (6)	3.55 (6)
0.9	2.90 (2)	27.39 (4)	15.12 (6)	7.89 (6)	3.59 (6)

β, α	Benders–with–Baropt				
	0.1	0.3	0.5	0.7	0.9
0.1	4.97 (2)	4.86 (2)	5.06 (2)	7.64 (3)	15.97 (6)
0.3	5.23 (2)	5.53 (2)	16.09 (5)	30.91 (9)	16.34 (6)
0.5	5.84 (2)	11.05 (3)	41.81 (10)	28.07 (8)	16.25 (6)
0.7	6.84 (2)	15.12 (3)	41.36 (8)	28.12 (8)	16.09 (6)
0.9	8.05 (2)	23.98 (4)	41.39 (8)	28.15 (8)	15.93 (6)

β, α	Branch–and–cut (CPLEX)				
	0.1	0.3	0.5	0.7	0.9
0.1	75.80 (82767)	91.62 (145763)	121.33 (162566)	141.96 (199826)	119.93 (161368)
0.1	77.37 (88267)	99.69 (156288)	117.77 (158843)	441.77 (236393)	121.26 (157888)
0.1	83.79 (86095)	95.19 (143148)	1027.8 (361530)	122.68 (166711)	120.32 (157888)
0.1	77.61 (84922)	93.10 (141979)	102.11 (129615)	122.58 (166711)	120.77 (157888)
0.1	79.98 (88707)	85.02 (130412)	102.23 (129615)	122.32 (166711)	122.2 (157888)

1000 facility locations – 1000 destinations

β, α	Benders–with–BlockIP				
	0.1	0.3	0.5	0.7	0.9
0.1	18.04 (2)	18.15 (2)	18.07 (2)	19.25 (3)	29.09 (7)
0.3	18.38 (2)	18.82 (2)	27.71 (3)	63.34 (8)	27.21 (6)
0.5	18.09 (2)	27.84 (2)	70.03 (5)	53.83 (6)	27.33 (6)
0.7	19.46 (2)	74.71 (3)	208.92 (8)	54.00 (6)	27.05 (6)
0.9	20.16 (2)	100.52 (3)	157.41 (6)	53.89 (6)	27.87 (6)

β, α	Benders–with–Baropt				
	0.1	0.3	0.5	0.7	0.9
0.1	35.09 (2)	36.20 (2)	36.72 (2)	59.44 (3)	144.28 (7)
0.3	40.30 (2)	41.18 (2)	72.47 (3)	216.97 (8)	140.21 (7)
0.5	42.69 (2)	48.20 (2)	164.24 (5)	193.69 (7)	140.30 (7)
0.7	48.32 (2)	96.80 (3)	267.14 (7)	192.33 (7)	140.55 (7)
0.9	53.19 (2)	115.36 (3)	249.55 (6)	192.58 (7)	141.01 (7)

β, α	Branch–and–cut (CPLEX)				
	0.1	0.3	0.5	0.7	0.9
0.1	1092.41 (190457)	1128.3 (497498)	1341.32 (433991)	1771.85 (521141)	3140.49 (364098)
0.3	1083.25 (183858)	1432.37 (503591)	1341.28 (/398657)	2371.77 (517030)	1040.00 (366078)
0.5	1048.64 (185015)	1344.06 (477182)	1328.27 (431055)	1384.44 (446753)	1218.97 (366078)
0.7	1113.06 (183404)	1215.51 (491545)	1505.74 (409296)	1753.35 (446753)	1036.53 (366078)
0.9	1066.31 (185925)	1199.02 (447635)	1256.25 (396701)	1361.38 (446753)	1118.97 (366078)

Six periods

Table 9: CPU time of instances of two sizes $m = n = 500$ and $m = n = 1000$, with six time periods ($k = 6$). The three tables report the CPU times of the three analyzed solution methods: Benders–with–BlockIP, Benders–with–BarOpt and Branch–and–cut. The values inside the parenthesis are either the number of Benders iterations (for the first two tables) or the number of MIP simplex iterations (for the last table)

500 facility locations – 500 destinations

β, α	Benders–with–BlockIP				
	0.1	0.3	0.5	0.7	0.9
0.1	4.86 (2)	4.85 (2)	5.29 (3)	6.21 (5)	15.62 (10)
0.3	5.01 (2)	6.81 (3)	18.32 (8)	30.85 (12)	8.59 (9)
0.5	5.32 (2)	14.53 (4)	75.02 (19)	20.54 (9)	8.59 (9)
0.7	15.30 (2)	39.33 (5)	95.68 (18)	20.53 (9)	8.34 (9)
0.9	27.09 (3)	108.11 (7)	95.76 (18)	20.50 (9)	8.16 (9)

β, α	Benders–with–Baropt				
	0.1	0.3	0.5	0.7	0.9
0.1	9.86 (2)	10.02 (2)	15.12 (3)	27.63 (5)	112.91 (20)
0.3	10.28 (2)	17.66 (3)	46.21 (7)	98.39 (14)	53.49 (10)
0.5	11.82 (2)	31.66 (4)	172.06 (20)	81.40 (11)	53.48 (10)
0.7	14.15 (2)	51.10 (5)	103.71 (10)	78.04 (11)	53.07 (10)
0.9	27.95 (3)	90.66 (7)	100.21 (10)	77.86 (11)	53.82 (10)

β, α	Branch–and–cut (CPLEX)				
	0.1	0.3	0.5	0.7	0.9
0.1	160.15 (174912)	237.84 (306138)	305.12 (320752)	358.98 (363900)	281.51 (311002)
0.3	162.65 (171758)	240.52 (312191)	314.58 (324738)	475.19 (307862)	498.59 (315966)
0.5	169.77 (174745)	272.58 (322766)	> 3600 (611082)	282.61 (299578)	490.38 (315966)
0.7	170.17 (170519)	233.78 (294001)	251.62 (266350)	313.44 (299578)	495.00 (315966)
0.9	166.23 (168989)	199.87 (252973)	277.84 (266350)	321.92 (299578)	499.23 (315966)

1000 facility locations – 1000 destinations

β, α	Benders–with–BlockIP				
	0.1	0.3	0.5	0.7	0.9
0.1	35.19 (2)	36.11 (2)	36.29 (2)	40.02 (4)	105.62 (10)
0.3	36.93 (2)	48.30 (3)	62.84 (4)	253.28 (18)	63.02 (9)
0.5	38.27 (2)	85.19 (4)	269.69 (7)	145.83 (9)	65.63 (9)
0.7	39.52 (2)	200.53 (4)	670.59 (14)	145.77 (9)	64.11 (9)
0.9	44.90 (2)	388.82 (5)	415.43 (11)	145.03 (9)	64.86 (9)

β, α	Benders–with–Baropt				
	0.1	0.3	0.5	0.7	0.9
0.1	76.39 (2)	73.05 (2)	74.45 (2)	153.22 (4)	233.15 (10)
0.3	81.84 (2)	134.18 (4)	195.97 (4)	298.28 (16)	103.91 (9)
0.5	87.89 (2)	225.99 (4)	458.89 (7)	275.83 (10)	174.63 (9)
0.7	100.74 (2)	274.54 (4)	1129.83 (14)	235.77 (9)	179.04 (9)
0.9	118.49 (2)	422.32 (5)	831.57 (10)	221.14 (9)	178.31 (9)

β, α	Branch–and–cut (CPLEX)				
	0.1	0.3	0.5	0.7	0.9
0.1	2806.43 (438667)	2615.41 (947346)	> 3600 (1037026)	> 3600 (936011)	3234.75 (703221)
0.1	2699.30 (458865)	3379.19 (946117)	3416.73 (1055603)	> 3600 (1030661)	2195.65 (709446)
0.1	2261.68 (447090)	3450.16 (931857)	> 3600 (970240)	> 3600 (765625)	2199.04 (709446)
0.1	2449.55 (432943)	3572.15 (980130)	3144.12 (912201)	> 3600 (792742)	2192.47 (709446)
0.1	2223.04 (405934)	2626.27 (871642)	3571.51 (918625)	> 3600 (779241)	2194.18 (709446)