An exact algorithm for a class of mixed-integer programs with equilibrium constraints

Teodora Dan [a]
Andrea Lodi [b]
Patrice Marcotte [c]

[a] Canada Excellence Research Chair in Data Science for Real-Time Decision-Making, Université de Montréal

[b] Canada Excellence Research Chair in Data Science for Real-Time Decision-Making, Polytechnique Montréal

[c] CIRRELT, Université de Montréal

`teodora.dan@umontreal.ca`
`andrea.lodi@polymtl.ca`
`marcotte@iro.umontreal.ca`

Canada Excellence Research Chair in Data Science for Real-Time Decision-Making

Abstract: In this study, we consider a rich class of mathematical programs with equilibrium constraints (MPECs) involving both integer and continuous variables. Such a class, which subsumes mathematical programs with complementarity constraints, as well as bilevel programs involving lower level convex programs is, in general, extremely hard to solve due to complementarity constraints and integrality requirements. For its solution, we design an (exact) branch-and-bound (B&B) algorithm that treats each node of the B&B tree as a separate optimization problem and potentially changes its formulation and solution approach by designing, for example, a separate B&B tree. The algorithm is implemented and computationally evaluated on a specific instance of MPEC, namely a competitive facility location problem that takes into account the queueing process that determines the equilibrium assignment of users to open facilities, and for which, to date, no exact method has been proposed.

Keywords: bilevel location, mixed integer programming, global optimization

1 Introduction

Mathematical programs with equilibrium constraints (MPECs) are NP-hard optimization problems that arise in engineering design, transportation, economics and multilevel games, to name a few areas of application. They embed constraints that are typically expressed as variational inequalities, which makes them highly nonconvex, even in the simplest cases. The aim of this work if twofold. First, we design an exact algorithm for an important class of mixed-integer MPECs, i.e., MPECs that involve both continuous and discrete variables. Next, we computationally evaluate the algorithm on a complex location-queueing model.

Formally, the model that we consider takes the mathematical form

$$\begin{aligned} \min_{x\in\mathbb{R}^n, y\in\mathbb{R}^m} \quad & f(x,y) \\ \text{s.t.} \quad & g_k(x,y) \le 0 \qquad k=1,\dots,r \\ & x_i \text{ integer} \qquad i=1,\dots,n\ , \end{aligned} \tag{1}$$

where the vector $y \in Y(x)$ satisfies the lower level variational inequality

$$\langle F(x,y), y-y' \rangle \le 0 \quad \forall y' \in Y(x)\ , \tag{2}$$

with $Y(x) = \{y \ge 0 : h_j(x,y) = 0, j = 1,\dots,p\}$.

Throughout the paper, we make the assumptions that all functions involved are continuously differentiable. Furthermore, f and $g_k : \mathbb{R}^n \to \mathbb{R}$ are convex, $h_j : \mathbb{R}^{n+m} \to \mathbb{R}$ are affine, and the mapping $F : \mathbb{R}^{n+m} \to \mathbb{R}^m$ is monotone in its second argument y. Whenever the Jacobian of F with respect to y is symmetric, F is a gradient mapping, i.e., $F = \nabla\phi$ for some convex function ϕ. In that case, a vector y satisfies the variational inequality (2) if and only if it is a solution of the lower level convex program

$$\begin{aligned} \text{LL:} \qquad \min_{y} \quad & \phi(y) \\ \text{s.t.} \quad & h_j(x,y) = 0 \quad j=1,\dots,p \\ & y \ge 0\ . \end{aligned} \tag{3}$$

Under our monotonicity and differentiability assumptions, the variational inequality (2) can be replaced by a Karush-Kuhn-Tucker (KKT) system. This allows to reformulate the global model as the mathematical program with complementarity constraints (MPCC)

$$\begin{aligned} \text{P:} \qquad \min_{\sigma,x,y} \quad & f(x,y) \\ \text{s.t.} \quad & g_k(x,y) \ge 0 && k=1,...,r \\ & x_i \text{ integer} && i=1,\dots,n \\ & h_j(x,y) = 0 && j=1,...,p && (4) \\ & 0 \le y \perp F(x,y) + \sum_{j=1}^{p} \sigma_j \nabla_y h_j(x,y) \ge 0\ , && && (5) \end{aligned}$$

where the vector of multipliers $\sigma \in \mathbb{R}^p$ is associated with the set of linear constraints.

The main difficulty associated with MPCCs is that the complementarity constraints involve both upper level (x), and lower level (y, σ) variables. Even in its simplest form $0 \le a \perp b \ge 0$ over scalars a and b, the feasible set is the union of two convex polyhedra, namely $\{(a,0) : a \ge 0\} \cup \{(b,0) : b \ge 0\}$. The linear independence constraint qualification (LICQ), requiring the gradients of the active constraints to be independent, is violated at all feasible points, which results in nonuniqueness of the constraint multipliers. The weaker Mangasarian-Fromovitz constraint qualification (MFCQ) is not satisfied either. We recall that MFCQ requires linear independence of the gradients of the equality constraints, and the existence of a direction d pointing into the interior of the region defined by the gradients of the active inequality constraints, such that $\langle \nabla h_j, d \rangle = 0$. Failure of LICQ and MFCQ results in unboundedness and nonuniqueness of the multipliers, therefore, many nonlinear programming (NLP) algorithms (and codes) could perform poorly because their performance strongly rests on these regularity conditions [Andreani and Martínez, 2001, Baumrucker et al., 2008].

Designing exact algorithms for this class of problems is a challenging task. Our novel algorithmic approach is based on a sophisticated combination of Mixed-Integer Linear Programming (MILP), linearization techniques and the iterative solution of convex subproblems. Roughly speaking, we start by embedding the solution space of P into an MILP relaxation – which is already nontrivial due to its nonconvexity – and, while performing a standard Branch and Bound (B&B), we iteratively solve either problem LL, to recover the true value of the objective function of a leaf (corresponding to a mixed-integer node), or some subproblems with strengthened relaxation (associated with a fractional node) to perform additional pruning in the tree and speed up convergence. In other words, we virtually treat each node of the B&B tree as a separate optimization problem to which we can adapt the formulation and, consequently, the solution technique, by building, for example, a separate B&B tree.

The algorithm is implemented and computationally tested on a specific instance of P, namely the *competitive congested user-choice discrete facility location problem* CC–FLP. CC–FLP is an extension of the classical (discrete) facility location problem, a fundamental structure in discrete optimization that is well suited to a variety of real-life applications. While it has been extensively considered in the literature, few studies have incorporated the specific features of a user-choice environment, which frequently involves congestion, either along the paths leading to a facility, or at the facility itself. The congested user-choice model belongs to the MPEC class and can be reformulated as an NP-hard bilevel program, thus falling into the category of mathematical programs with complementarity constraints.

Paper Contribution. The paper provides two strong contributions.

- On the one side, it proposes a novel exact algorithm for solving a fairly general class of mathematical programs with equilibrium constraints. To some extent, such an algorithm virtually treats every node as a separate, i.e., different, optimization problem, adapting either the relaxation or the solution method (or both) depending on some triggering conditions. This is done to achieve full flexibility and to exploit in the most effective way both the strength of the MILP solvers, e.g., their strong preprocessing at the root node, and the pieces of information acquired while exploring the enumeration tree, which could lead to alternative algorithmic decisions and/or problem formulations. Although this is not the only way to exploit this idea (see next section), we believe that it is a fundamental step in the direction of designing complex adaptive algorithms that have the capability, within the same B&B tree, to change solution strategies and formulations, whenever required.
- On the other side, and to the best of our knowledge, our algorithm is the first exact method for CC–FLP, which is a practically relevant and remarkably difficult generalization of the classical facility location problem. So far, in the literature, CC–FLP or variants thereof were only addressed by means of heuristic methods.

Paper Organization. The paper is structured as follows. In Section 2, we detail the novel solution method for P and discuss its connection with existing literature, together with its underlying assumptions and limitations. In Section 3, we describe CC–FLP, including a literature review, and we provide all necessary details for adapting the proposed algorithm to this application. In Section 5, we report on the extensive computational experiments for exactly solving CC–FLP. Finally, in Section 6, we draw some conclusions and open avenues for further research on this topic.

2 Algorithmic framework

Based on B&B, our solution approach consists of two main phases.

Phase I In the first phase, we perform a linearization of nonlinear terms and constraints, in order to reduce the original program to an MILP. In our generic model P, there are two distinct sources of nonlinearity, namely F and the complementarity constraints. For linear F, the only linearization needed would involve the complementarity constraints, for which we could write an exact linear reformulation via binary variables[1]. For the sake of generality, we henceforth consider F to be nonlinear. Then, there are several options for linearization.

- Single level linearization:
  - Perform a piecewise linear approximation of F.
  - Introduce big-M constraints to linearize the complementarities.
- Bilevel linearization:
  - Replace the equilibrium constraints by a lower-level program, linearize it, then write the KKT optimality conditions.
  - Linearize the resulting program by replacing the bilinear terms with McCormick envelopes.

The main difficulty in both cases is to perform linear approximations that guarantee that the objective function of the obtained MILP is a bounding function for the objective of the original program. Let $\bar{F}$ be a piecewise linear approximation of F and let $(\tilde{\sigma}, \tilde{x}, \tilde{y})$ be a feasible solution of P. Similarly, given $\tilde{x}$, let $(\bar{\sigma}, \bar{y})$ satisfy (4)–(5) when F is replaced by $\bar{F}$. Due to the approximation of F, $\bar{y}$ may not correspond to the true solution of the lower level, therefore $\bar{y}$ may differ from $\tilde{y}$. It follows that $f(\bar{x}, \bar{y})$ may also differ from $f(\tilde{x}, \bar{y})$. This implies that, after linearization and relaxation of integrality constraints at integer nodes, the approximated objective value may not correspond to the true objective value. Our approach is to perform a linear approximation such that the objective function of the obtained MILP be a valid upper bound on the objective of the original program. This will ensure that

$$\textit{optimum of relaxed MILP} \leq \textit{optimum of MILP} \leq \textit{optimum of original problem.} \tag{6}$$

Inequality (6) must hold in order for B&B not to prune nodes that might contain the optimal solution. In other words, the constructed MILP must be a valid relaxation of P.

Phase II In the second phase, we implement a B&B algorithm to find the optimal solution of the original problem. At the leaves of a standard B&B tree, the bound equals the value of the corresponding solution. In our case, this is not necessarily true, due to the approximations performed in Phase I. We tackle this issue by interacting with the B&B throughout its execution, i.e., by computing the correct value of the objective function at any associated leaf. To achieve this, we fix the upper level (integer) variables to their corresponding values at the current node, we compute the optimal solution of the (convex) lower-level problem, and we recover the associated objective value of the original problem. In order to reduce the size of the B&B tree and to make the algorithm more efficient, in some of the fractional nodes we compute an on-the-fly lower-bound for the subtree rooted at the corresponding node, by taking into account the bounds of some of the variables. This algorithmic framework is summarized in Algorithm 1 (see pseudocode below).

Algorithm 1

```
1: perform linear approximations of nonlinear terms and constraints so as to satisfy (6)
2: solve the resulting MILP using B&B:
3: for all nodes in the tree do
4:     if integer node then
5:         compute the true corresponding objective function value
6:     else
7:         if some condition holds then
8:             compute a tighter lower bound (by improving the relaxation)
9: return best found solution
```

Algorithm 1 performs Phase I at Step 1. This can be achieved either under the single level or bilevel linearization. The quality of those linearizations will be discussed later on in the paper and especially

[1]The reader is referred to [Hijazi and Liberti, 2014, Jeroslow and Lowe, 1984] for a discussion on the theoretical representability of unbounded disjunctions.

evaluated in the computational investigation for the special case of the CC–FLP. The rest of Algorithm 1 concerns Phase II and implements the idea, outlined in the introduction, that every node of the B&B tree can potentially be treated differently. What is described in the algorithm above is the distinction between fractional and integer nodes, where we either (potentially) improve the relaxation or compute the real objective function value by solving an auxiliary continuous optimization problem, respectively. However, a third special case for the CC–FLP involves B&B nodes in which a certain set of variables, namely the facility location ones, is settled, i.e., all decisions on where locating the facilities are taken but some other integer decisions are not. Each of those nodes is reformulated and solved by the MILP solver IBM CPLEX as a separate MILP within the same B&B algorithm by taking advantage of `callback` functions, which allows the solver to exploit the full power of its preprocessing phase by heavily simplifying the formulation. Of course, the overall branching scheme is tailored so as the location variables are settled first, thus encountering those nodes as early as possible in the search tree.

We close this discussion by outlining the relationship between our solution approach and the relevant literature with respect to the management of the B&B tree.

- First, our algorithm clearly borrows from the vast literature on Global Optimization (GO) the idea (and the need) of iteratively improving the relaxation within the enumeration scheme. In GO, this is achieved by Spatial Branching (i.e., by iteratively improving the convex relaxation) and by applying expensive bound tightening in virtually any node. Of course, the relaxation can be (and is actually) improved in MILP as well by cutting plane generation. However, our approach is a hybrid because, within an MILP scheme, what we do in the majority of cases in (selected) fractional nodes is to tighten the formulation not by exploiting integrality (like for cut generation in MILP) but by improving the linearization of the nonlinear component(s) of the original problem. This is in the spirit, for example, of the work of [Belotti et al., 2016], where big-M constraints are iteratively strengthened within the MILP B&B as a GO solver would do. With respect to [Belotti et al., 2016], Step 8 of Algorithm 1 goes slightly further by providing a different formulation of the node (by improving the piecewise linear approximation) instead of simply tightening some of its constraints individually. This is related to the work of [Furini and Traversi, 2013] where, at some nodes of the B&B tree of a mixed-integer nonlinear programming (MINLP) approach to binary quadratic programming, an alternative semidefinite programming (SDP) relaxation is defined and solved with the aim of improving the bound, thus possibly fathoming the node. The difference is that in Algorithm 1 the improved reformulation of a node is kept in the entire subtree rooted at the node itself, while the SDP node in [Furini and Traversi, 2013] is discarded if its improved bound does not allow the subtree to be fathomed. Ideally, of course, one can think of solving the associated mixed-integer SDP instead, which is not done in [Furini and Traversi, 2013].
- Second, the idea of reformulating some of the nodes of a B&B scheme so as to exploit some special structure has been used with different flavors, for example, in [Raghunathan, 2013]. More precisely, [Raghunathan, 2013] solves by B&B a convex MINLP relaxation of a water network design problem. However, at the nodes in which all binary variables are decided (namely, the diameters of the pipes in the designed network), instead of solving the associated continuous convex relaxation, it is observed that the original nonconvex counterpart has a unique solution and such a problem is efficiently solved directly. In other words, the convex MINLP relaxation is used to embed the solution approach within a rather efficient (although nonlinear) B&B framework, while the special structure of the continuous optimization problems obtained after fixing the diameters is exploited to more accurately solve those nodes. This is precisely what we achieve at Step 5 of Algorithm 1.

Overall, Algorithm 1 yields an effective framework that uses in a flexible way several algorithmic ingredients like preprocessing, reformulation, linearization, etc. through an extremely sophisticated tree management. We believe that managing the enumeration tree flexibly is key to solving extremely difficult nonconvex MINLPs like the one discussed in the next section.

3 CC–FLP

To illustrate our algorithmic framework, we consider the problem faced by a company making location and service level decisions in a market where competitors already operate. The aim of the company is to maximize the number of attracted customers, subject to a budget constraint. In the model, queueing at facilities, together with user's selfish behavior, are explicitly taken into account. Under these assumptions clients select and patronize the facility minimizing their disutility, expressed as the weighted sum of travel and waiting time. Assuming constant travel times, the underlying network reduces to a bipartite graph. Let $V = I \times J$ be a complete bipartite graph, where I denotes the set of demand nodes, and J the set of locations. Every node $i \in I$ represents a population zone of a city, and is endowed with an inelastic demand d_i. Let J_1 denote the set of candidate locations of the emerging company, while J_c is the set of competitors' locations. A client originating from node $i \in I$ and patronizing facility $j \in J$ incurs a travel time t_{ij}. Arriving at facility j, she enters an $M/M/c_j/\infty$ queue, equipped with c_j servers having identical mean service rate μ, and mean (queueing plus service) delay w_j . The disutility u_{ij} is defined as a linear combination of travel time queueing, and service time at the facility, namely

$$u_{ij} = t_{ij} + \alpha w_j, \tag{7}$$

where α is a positive weight coefficient.

3.1 Literature review

Although location problems have been widely studied, few models incorporate user behavior, congestion and competition. In our model, customers select the facilities to patronize, based on the travel time and waiting time at facilities, the latter being a congestion trait. User behavior can be modeled as a Stackelberg game, in which the company locating facilities is the leader, and users represent the follower, fitting the bilevel paradigm. When describing the patronizing behavior, we identify two possibilities: deterministic and probabilistic.

In the deterministic case, users select the facility that minimizes their disutility. In simpler models, all users originating from a demand point patronize the same facility, and disutility does not account for congestion. Under these assumptions [Marić et al., 2012] implement three metaheuristics to solve a bilevel formulation of the uncapacitated FLP with user preferences. Similarly, in [Camacho-Vallejo et al., 2014], the bilevel FLP with user preferences is solved by using a Stackelberg Evolutionary Algorithm. In [Vidyarthi and Jayaswal, 2014], a model where congestion is minimized in the objective function, and users patronize the closest facility is considered. The authors solve their model by constraint generation.

When demand can be split between several facilities, a tie resolution rule must be implemented as well. Within this framework we note the work of [Desrochers et al., 1995] who consider a centralized model of a deterministic facility location problem, where individual delays increase with distance. The authors provide a user-choice version of their model that fits the bilevel programming paradigm, although they do not propose a solution algorithm. In [Berman and Drezner, 2006], the location of congested facilities when demand is elastic with respect to distance is investigated. The objective is to maximize total demand, subject to constraints on the waiting time at facilities. Heuristic procedures are proposed.

The presence of non-linearities in the user utility makes the problem even more difficult to solve, thus, only few papers tackle this aspect. We cite here [Sun et al., 2008], who consider a generic bilevel facility location model, where the upper level is making locational decisions that minimize the sum of total cost and a congestion function, and the lower level (users) minimizes a non-linear function. The authors propose a heuristic algorithm as solution method. Another work worth mentioning is that of [Zhang et al., 2010], who propose a methodology for addressing a congested facility network design, with the aim of maximizing the participation rate, in a preventive healthcare setting. Demand is elastic with respect to total expected time (travel + waiting time at facilities) experienced by clients. Users patronize the facility minimizing the sum of waiting and travel time. The proposed solution method is a Tabu Search procedure.

In the probabilistic case, users behave according to a discrete choice model based on the random utility paradigm. In the case of Gumbel distributed random terms, this yields Logit closed form expressions for the origin-destination flows. Similar to the deterministic case, most papers consider the utility to be solely based on proximity.

A more elaborate model is proposed by [Abouee-Mehrizi et al., 2011], who consider simultaneous decision-making over the location, service rate and price, for facilities located at vertices of a network. Demand is elastic with respect to price, and congestion arising at facilities is characterized by queueing equations. Clients spread among facilities based on proximity only, according to a Multinomial Logit random utility model that takes balking into account. As solution method, the authors propose a hybrid between Tabu Search and a tailored heuristic algorithm.

When it comes to utility, most papers make simplifying assumptions. For one, they assume that users patronize facilities based solely on proximity. To the best of our knowledge, the first paper to address congestion in a competitive user-choice environment is that of [Marianov et al., 2008]. The authors consider a scenario in which a company locates p facilities on the vertices of a network where competitors are already operating. The demand is inelastic and users patronize the facility minimizing their disutility, given by the sum of travel and waiting time. The model is solved by using a two-phase metaheuristic procedure combining GRASP and Tabu Search. In the initial phase, facility locations are selected and a nonlinear assignment problem is solved using the Newton-Raphson algorithm. In [Dan and Marcotte, 2017], it is considered a bilevel network design problem, where a firm makes decisions on both location and service levels, and users patronize the facility minimizing the travel and waiting time at facilities, yielding a non-linear bilevel program. The authors propose an approximation algorithm that is asymptotically optimal, as well as a heuristic that exploits the structure of the problem.

3.2 Modeling CC–FLP

Throughout this paper we will be using the following notation.

Sets

I:	set of demand nodes;
J:	set of candidate facility locations (leader and competitor);
J_c:	set of competition's facilities;
J_1:	set of leader's candidate sites;
$J_1^* \subseteq J_1$:	set of leader's open facilities;
$J^* \subseteq J$:	set of open facilities (leader and competitor).

Parameters

d_i:	demand originating from node $i \in I$;
μ:	service rate at open facility $j \in J$;
t_{ij}:	travel time between nodes $i \in I$ and $j \in J$;
α:	coefficient of the waiting time in the disutility formula;
B:	available budget (for opening new facilities and associated service rates);
f_c:	fixed cost associated with opening a new facility;
v_c :	cost per server.

Basic decision variables

y_j:	binary variable set to 1 if a facility is open at site j, and to 0 otherwise;
c_j:	number of servers at open facility $j \in J$.

Additional variables

x_{ij}:	arrival rate at facility $j \in J$ originating from demand node $i \in I$;
λ_j:	arrival rate at node $j \in J$;
ρ_j:	traffic intensity at facility $j \in J$;
w_j:	mean queueing time at facility j;
γ_i:	disutility of users originating from node i.

Let x_{ij} be the number of clients from demand node i patronizing facility j. We define the arrival rate $\lambda_j = \sum_{i \in I} x_{ij}$ at facility j, as well as the number c_j of servers operating at j. Then, the mean waiting time (queueing plus service) at facilities w is a bivariate function depending on both the arrival rate and number of servers [Kleinrock, 1975]

$$w(\lambda_j, c_j) = \frac{1}{(\mu c_j - \lambda_j)\left(1 + \frac{(1-\rho_j)\, c_j!}{(c_j\rho_j)^{c_j}} \sum_{k=0}^{c_j-1} \frac{(c_j\rho_j)^k}{k!}\right)} + \frac{1}{\mu}, \tag{8}$$

where $\rho_j = \lambda_j/(\mu c_j) < 1$ is the traffic intensity.

In a user-choice environment, users patronize facilities minimizing their disutility. Given the facility locations and the assigned number of servers, the lower level problem is a user equilibrium problem (Wardrop). The equilibrium is defined by the complementarity system

$$0 \le x_{ij} \perp t_{ij} + \alpha w_j - \gamma_i \ge 0, \qquad i \in I;\ j \in J. \tag{9}$$

The complete model is as follows.

CC–FLP:

LEADER (COMPANY)

$$\max_{y,c,x,\gamma} \quad z = \sum_{i\in I}\sum_{j\in J_1} x_{ij} \tag{10}$$

$$\text{s.t.} \quad \sum_{j\in J_1} (f_c \cdot y_j + v_c \cdot c_j) \le B \tag{11}$$

$$c_j \le M \cdot y_j \qquad j \in J_1 \tag{12}$$

$$c_j \ge y_j \qquad j \in J_1 \tag{13}$$

$$y_j \in \{0,1\} \qquad j \in J_1 \tag{14}$$

$$c_j \ge 0,\ c_j \text{ integer} \qquad j \in J_1 \tag{15}$$

FOLLOWER (USERS)

$$0 \le x_{ij} \perp t_{ij} + \alpha w_j - \gamma_i \ge 0 \qquad i \in I;\ j \in J \tag{16}$$

$$w_j = \frac{1}{(\mu c_j - \lambda_j)\left(1 + \frac{(1-\rho_j)\, c_j!}{(c_j\rho_j)^{c_j}} \sum_{k=0}^{c_j-1} \frac{(c_j\rho_j)^k}{k!}\right)} + \frac{1}{\mu} \qquad j \in J \tag{17}$$

$$\lambda_j = \sum_{i\in I} x_{ij} \qquad j \in J \tag{18}$$

$$\rho_j = \frac{\lambda_j}{\mu c_j} \qquad j \in J \tag{19}$$

$$\sum_{j\in J} x_{ij} = d_i \qquad i \in I \tag{20}$$

$$\lambda_j \le \mu_j c_j \qquad j \in J \tag{21}$$

$$x_{ij} \ge 0 \qquad i \in I; j \in J, \tag{22}$$

where M is a suitably large constant, that we set to $(B - f_c)/v_c$, as a direct consequence of (12).

The decision variables are the vectors y (locations) and c (number of servers), while the user assignment x is the solution of an equilibrium problem that can be equivalently obtained by solving a convex optimization problem. The objective in Eq. (10) is to maximize the total number of users that patronize the leader's facilities, while constraint (11) ensures that the total cost does not exceed the budget B. Constraints (12) set the upper bound for the number of servers per facility. In order to avoid irrelevant solutions, constraints (13) specify that at least one server must be assigned to any open facility. Logical constraints (16)–(19) enforce the user equilibrium conditions. Finally, constraints (20) ensure that demand is satisfied, and constraints (21) guarantee that the arrival rate does not exceed the total service rate.

It is clear that CC–FLP fits the generic model P. We now discuss some simple but important properties of our model.

Proposition 3.1 *The waiting time function w is strictly increasing in λ and strictly decreasing in c.*

Proof. We note that $A = \dfrac{c!}{(c\rho)^c} \sum_{k=0}^{c-1} \dfrac{(c\rho)^k}{k!}$. Then, $w(\lambda, c) = \dfrac{C}{\mu c - \lambda}$, where $C = \dfrac{1}{1 + (1-\rho)\, A}$. We have that

$$\frac{\partial C}{\partial \rho} = -C^2 \left[-A + (1-\rho) \frac{\partial A}{\partial \rho} \right] = C^2 \left[\frac{c!}{c^c} \sum_{k=0}^{c-1} \frac{c^k \rho^{k-c-1}}{k!} \left((c-k)\,(1-\rho) + \rho \right) \right] > 0.$$

C is thus, strictly increasing in ρ. But $\dfrac{1}{\mu c - \lambda}$ is strictly increasing in λ and strictly decreasing in c, so the conclusion follows. □

Proposition 3.2 [Lee and Cohen, 1983] *The waiting time function w is convex in λ.*

Now, for given couples (λ, c), we consider under and over estimators of w, respectively $\tilde{w}$ and $\hat{w}$, such that

$$\tilde{w}(\lambda, c) \leq w(\lambda, c) \text{ , and } \hat{w}(\lambda, c) \geq w(\lambda, c) \ . \tag{23}$$

We will now show that, if we replace w_j with $\tilde{w}_j$ at the leader's facilities, and with $\hat{w}_j$ at competitors facilities, the optimal solution of the resulting program yields a valid upper bound on the optimum of the initial problem.

The objective function defined by Eq. (10) can be expressed as $z = \sum_{j \in J_1} \lambda_j$, where λ_j is the total number of users patronizing facility j. Let CC–FLP' be a modified version of CC–FLP, where w_j in constraints (16)–(17) has been replaced by $\tilde{w}_j$ at leader's facilities, and with $\hat{w}_j$ at competitors' facilities. Let x', λ' and γ' be an optimal solution of CC–FLP', with objective $z' = \sum_{j \in J_1} \lambda'_j$.

Proposition 3.3 *For any feasible pair (y, c), the optimal objective function of CC–FLP' is greater or equal than the optimal objective of CC–FLP.*

Proof (by contradiction). Assume that the proposition is false. Then, $\sum_{j \in J_1} \lambda_j > \sum_{j \in J_1} \lambda'_j$ implies the existence of a facility $j \in J_1$ such that $\lambda'_j < \lambda_j$, and there exists $k \in J_c$ such that $\lambda'_k > \lambda_k$. From Proposition 3.1, it follows that

$$w(\lambda_k, c) < w(\lambda'_k, c) \leq \hat{w}(\lambda'_k, c) \tag{24}$$

$$w(\lambda_j, c) > w(\lambda'_j, c) \geq \tilde{w}(\lambda'_j, c) \ . \tag{25}$$

In other words, a fraction of the population patronizing the leader's facilities in CC–FLP switches to competitors' facilities in CC–FLP'. Therefore, there exists a demand node i, together with a population ϵ that originates from i, patronizes both facility $j \in J_1$ in CC–FLP and facility $k \in J_c$ in CC–FLP'. Then,

$$0 < x'_{ij} + \epsilon = x_{ij} \text{ , and } \quad x_{ik} + \epsilon = x'_{ik} \ .$$

From Eq.(16), we have that $t_{ij} + \alpha w(\lambda_j, c) = \gamma_i$ (since $x_{ij} > 0$). If follows from Eq.(25) that $t_{ij} + \alpha w(\lambda_j, c) > t_{ij} + \alpha \tilde{w}(\lambda'_j, c) \geq \gamma'_i$, and we have $\gamma_i > \gamma'_i$. Similarly, from Eq.(16), we have that $t_{ik} + \alpha w(\lambda_k, c) \geq \gamma_i$, and from Eq.(24) we have $t_{ik} + \alpha w(\lambda_k, c) < t_{ik} + \alpha \hat{w}(\lambda'_k, c) = \gamma'_i$, which yields $\gamma_i < \gamma'_i$, a contradiction. □

3.3 Linearization

The first phase of our algorithmic framework is the linearization of the non-linear terms and constraints, in order to reformulate CC–FLP as an MILP. The key task is the linearization of the highly nonlinear two-variable waiting time function. As previously mentioned for CC–FLP, we will consider two approximation schemes, respectively single-level and bilevel.

3.3.1 Single-level linearization

The underlying idea of this method is to directly linearize the complementarity condition (5), i.e., Wardrop conditions (9) and, independently, the waiting time functions, with the aim to obtain an MILP. This is outlined below.

Linearization of equilibrium constraints

For a given set of open facilities y and their assigned number of servers c, the lower level flows x_{ij} are solution of the non-linear complementarity problem

$$0 \leq x_{ij} \perp t_{ij} + \alpha w_j - \gamma_i \geq 0 \qquad i \in I;\ j \in J \tag{26}$$

which is linearized through the introduction of binary variables and big-M constants:

$$t_{ij} + \alpha w_j - \gamma_i \leq M_\gamma s_{ij} \qquad i \in I;\ j \in J \tag{27}$$

$$x_{ij} \leq d_i(1 - s_{ij}) \qquad i \in I;\ j \in J \tag{28}$$

$$s_{ij} \in \{0,1\} \qquad i \in I;\ j \in J\ . \tag{29}$$

Tight values for the constant M_γ can be derived from information concerning the maximal queueing time $w_{\max}$ and maximal travel time $t_{\max} = \max_{i\in I, j\in J}\{t_{ij}\}$. Since the total demand must be strictly less than the total service rate, there must exist a minimum number of servers C so that $\sum_{i\in I} d_i < C\mu$. Let $\ell = \lfloor B/f_c \rfloor$ be the maximum number of facilities allowed by the budget, and let $\varepsilon = (C - \sum_{i\in I} d_i)/(\ell + |J_c|)$. Then, an upper bound $\overline{C}$ on the number of servers at leader's or competitor's facilities is

$$\overline{C} = \max\left\{\max_{j\in J_c}\{c_j\}, (B - f_c)/v_c\right\}.$$

At optimality, there exists at least one facility j_1, belonging either to the leader or to the competition, such that $c_{j_1}\mu \geq \lambda_{j_1} + \varepsilon$. A priori, facility j_1 is unknown, but its number of servers can vary from 1 to $\overline{C}$. There follows the upper bound

$$w_{\max} \leq \max_{c\in\left\{1,\ldots,\overline{C}\right\}, c\mu > \varepsilon} \{w(c\mu - \varepsilon, c)\}\,. \tag{30}$$

Let $\gamma_{\max}$ and $\gamma_{\min}$ be the maximum and minimum values that γ can assume at optimality, respectively. Since $\gamma_{\min} \geq 0$ and $\gamma_{\max} \leq t_{\max} + \alpha w_{\max}$, one can set $M_\gamma = t_{\max} + \alpha w_{\max}$. Note that for a given number N_o of leader's open facilities the value of M_γ can be further reduced, yielding

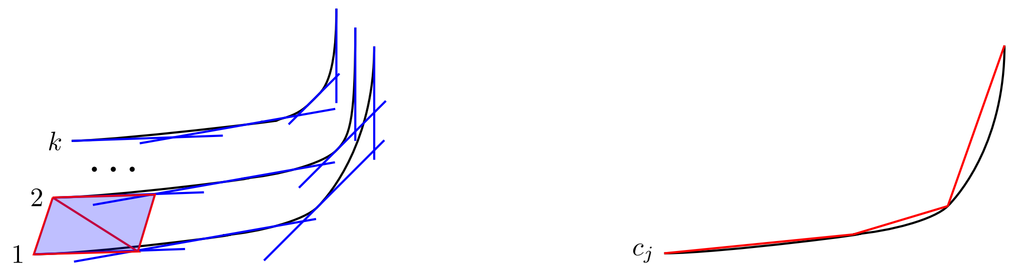
$$\overline{C} = \max\left\{\max_{j\in J_c}\{c_j\}, (B - N_o f_c)/v_c\right\}$$

and $\varepsilon = (C\mu - \sum_{i\in I} d_i)/(N_o - |J_c|)$. In some cases, this yields a smaller value of M_γ.

Linearization of waiting time

In order to derive an upper bound on the leader's profit, we perform an under approximation of queueing delays at its facilities, and an over approximation of the delays at the competitor's facilities, respectively, as illustrated in Figure 1.

The waiting time at the leader's facilities is a nonlinear bivariate function of variables λ_j and c_j. Given $w_{\max}$, and for each number of servers k, we select a number $\lambda^k_{\max} < \mu k$ such that $w(\lambda^k_{\max}, k) \geq w_{\max}$. Considering the maximum number of servers $c_{\max}$ allowed by the budget B, one samples each interval $[0, \lambda^k_{\max}]$ using N_l points $\lambda^{kn}, k = 0, \ldots, c_{\max}, n = 1, \ldots, N_l$ such that $\lambda^{ki} < \lambda^{kj}$ for all $1 \leq k \leq c_{\max}$ and $1 \leq i < j \leq N_l$. Next, we compute the tangent lines at λ^{kn}, as well as the intersections between each two consecutive lines, yielding $N_l + 1$ points, including the endpoints 0 and $\lambda^k_{\max}$. We denote these points as $(\tilde{\lambda}^{kn}, \underline{w}^{kn})$, where $\tilde{\lambda}^{kn}$ corresponds to the sample on λ, and $\underline{w}^{kn}$ is an under approximation of w at $(\tilde{\lambda}^{kn}, k)$. Since w_j



(a) Leader: under approximation $\tilde{w}$ (b) Follower: over approximation $\hat{w}$

Figure 1: Piecewise linear approximation of queueing delay w.

is convex in λ_j, the piecewise linear approximation provides a valid lower bound. Next, we base our linear approximation of the w constraint on the triangle technique described in [D'Ambrosio et al., 2010]. This yields

$$\sum_{k=0}^{c_{\max}} \sum_{n=1}^{N_l+1} \left(\underline{l}_{jkn} + \underline{l}_{jkn}\right) = 1, \qquad j \in J_1 \tag{31}$$

$$\tilde{s}_{jkn} \le \bar{l}_{jkn} + \underline{l}_{jkn} + \bar{l}_{jkn-1} + \qquad \underline{l}_{jk-1n-1} + \bar{l}_{jk-1n-1} + \underline{l}_{jk-1n}$$
$$j \in J_1; k = 0, \dots, c_{\max}; n = 1, \dots, N_l + 1 \tag{32}$$

$$\sum_{k=0}^{c_{\max}} \sum_{n=1}^{N_l+1} \tilde{s}_{jkn} = 1, \qquad j \in J_1 \tag{33}$$

$$\lambda_j = \sum_{k=0}^{c_{\max}} \sum_{n=1}^{N_l+1} \tilde{s}_{jkn} \tilde{\lambda}^{kn}, \qquad j \in J_1 \tag{34}$$

$$c_j = \sum_{k=0}^{c_{\max}} \sum_{n=1}^{N_l+1} \tilde{s}_{jkn} k, \qquad j \in J_1 \tag{35}$$

$$\tilde{w}_j \ge \sum_{k=0}^{c_{\max}} \sum_{n=1}^{N_l+1} \tilde{s}_{jkn} \underline{w}^{kn}, \qquad j \in J_1 \tag{36}$$

$$\bar{l}_{jkn}, \underline{l}_{jkn} \in \{0, 1\} \qquad j \in J_1; k = 0, \dots, c_{\max}; n = 1, \dots, N_l + 1 \tag{37}$$

$$0 \le \tilde{s}_{jkn} \le 1 \qquad j \in J_1; k = 0, \dots, c_{\max}; n = 1, \dots, N_l + 1 \tag{38}$$

$$\bar{l}_{j,-1,n} = 0, \quad \underline{l}_{j,-1,n} = 0 \qquad j \in J_1; n = 1, \dots, N_l + 1 \tag{39}$$

$$\bar{l}_{j,c_{\max},n} = 0, \quad \underline{l}_{j,c_{\max},n} = 0 \qquad j \in J_1; n = 1, \dots, N_l + 1 \tag{40}$$

$$\bar{l}_{j,k,0} = 0, \quad \underline{l}_{j,k,0} = 0 \qquad j \in J_1; k = 0, \dots, c_{\max} \tag{41}$$

$$\bar{l}_{j,k,N_l+1} = 0, \quad \underline{l}_{j,k,N_l+1} = 0 \qquad j \in J_1; k = 0, \dots, c_{\max}. \tag{42}$$

We perform a similar linearization for the waiting time at competing facilities, where the number of servers c_j is constant. Next, we construct a piecewise linear function $\hat{w}_j$ so that $\hat{w}_j \le w_j$, $\forall j \in J_c$. Given $w_{\max}$, we compute $\hat{\lambda}^j_{\max} < \mu c_j$ such that $w_j(\hat{\lambda}^j_{\max}, c_j) \ge w_{\max}$ and sample the interval $[0, \hat{\lambda}^j_{\max}]$ using N_c points $\hat{\lambda}^{jn}, n = 1, \dots, N_c$ such that $\hat{\lambda}^{jn} < \hat{\lambda}^{jm}$ for all $1 \le n < m \le N_c$. This yields the linearization

$$\sum_{n=1}^{N_c} \hat{s}_{jn} = 1 \qquad j \in J_c \tag{43}$$

$$\lambda_j = \sum_{n=1}^{N_c} \hat{s}_{jn} \hat{\lambda}^{jn} \qquad j \in J_c \tag{44}$$

$$\hat{w}_j \le \sum_{n=1}^{N_c} \hat{s}_{jn} w(\hat{\lambda}^{jn}, c_j) \qquad j \in J_c \tag{45}$$

$$\sum_{n=1}^{N_c} \hat{l}_{jn} = 1 \qquad j \in J_c \tag{46}$$

$$\hat{s}_{jn} \leq \hat{l}_{jn} + \hat{l}_{jn-1} \qquad j \in J_c;\ n = 1, \dots, N_c \tag{47}$$

$$\hat{l}_{jn} \in \{0, 1\} \qquad j \in J_c;\ n = 1, \dots, N_c \tag{48}$$

$$0 \leq \hat{s}_{jn} \leq 1 \qquad j \in J_c;\ n = 1, \dots, N_c \tag{49}$$

$$\hat{l}_{j,0} = \hat{l}_{j,N_c} = 0 \qquad j \in J_c. \tag{50}$$

The inequality form of constraints (36) and (45) ensure that the original w is feasible for the approximated problem. The MILP relaxation of CC–FLP then becomes

$$\begin{aligned} \text{CC–FLP1:} \quad & \max_{y,c,x,\gamma} \quad z = \sum_{i \in I} \sum_{j \in J_1} x_{ij} \\ & \text{s.t.} \quad \text{constraints (11)–(15), (18)–(22), (26)–(29), (31)–(50).} \end{aligned}$$

3.3.2 Bilevel linearization

An alternative to the linearization of the user equilibrium conditions (16)–(20) is to replace the latter by the convex optimization problem [Beckmann et al., 1956]

$$\text{PL:} \quad \min_{x,\lambda} \quad \sum_{i \in I} \sum_{j \in J^*} x_{ij} t_{ij} + \alpha \sum_{j \in J^*} \int_0^{\lambda_j} w_j(\tau, c_j) d\tau \tag{51}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{j \in J^*} x_{ij} = d_i && i \in I \\ & x_{ij} \geq 0 && i \in I; j \in J^* \\ & \lambda_j = \sum_{i \in I} x_{ij} && j \in J^*, \end{aligned}$$

whose KKT conditions match the Wardrop conditions (26). Based on this formulation, an MILP approximation can be obtained by (i) performing a piecewise linearization of the sole nonlinear term $\int_0^{\lambda_j} w_j(q, c_j)$, (ii) writing an equivalent linear program, (iii) linearizing the complementarity term of the optimality conditions.

We now provide the technical details. Let us construct piecewise linear functions $\tilde{W}(\lambda, c)$ (leader) and $\hat{W}(\lambda, c)$ (competition), whose respective derivatives $\tilde{w}(\lambda, c)$ and $\hat{w}(\lambda, c)$ satisfy (23). This is achieved as follows. Given $w_{\max}$ as defined by Eq. (30), and for each number of servers c, let $\tilde{\lambda}^c_{\max} < \mu c$. We sample N_l points $\tilde{\lambda}^{cn}$ ($c = 1, \dots, c_{\max}, n = 1, \dots, N_l$), sorted in increasing order, in each interval $[0, \lambda^c_{\max}]$. Then, for each value of c, the integral of the waiting time at leader's facilities is approximated by the piecewise linear function

$$\begin{aligned} \tilde{W}_j(\lambda, c) &= (\lambda - \tilde{\lambda}^{cl}) w(\tilde{\lambda}^{cl}, c) + \sum_{k=2}^{l} (\tilde{\lambda}^{ck} - \tilde{\lambda}^{c,k-1}) w(\tilde{\lambda}^{c,k-1}, c) \qquad \text{if } \lambda \in [\tilde{\lambda}^{cl}, \tilde{\lambda}^{c,l+1}] \\ &= \lambda \underbrace{w(\tilde{\lambda}^{cl}, c)}_{\tilde{f}^{cl}_j} + \underbrace{\sum_{k=2}^{l} (\tilde{\lambda}^{ck} - \tilde{\lambda}^{c,k-1}) w(\tilde{\lambda}^{c,k-1}, c) - \tilde{\lambda}^{cl} w(\tilde{\lambda}^{cl}, c)}_{\tilde{g}^{cl}_j} . \end{aligned} \tag{52}$$

Similarly, we linearize the integral of the waiting time at competitors' facilities. Given $w_{\max}$, we set $\hat{\lambda}^j_{\max} < \mu c_j$ such that $w_j(\hat{\lambda}^j_{\max}, c_j) \geq w_{\max}$. We sample the interval $[0, \hat{\lambda}^j_{\max}]$ using N_c points $\hat{\lambda}^{jn}, n = 1, \dots, N_c$, sorted in increasing order and consider the piecewise linear approximation

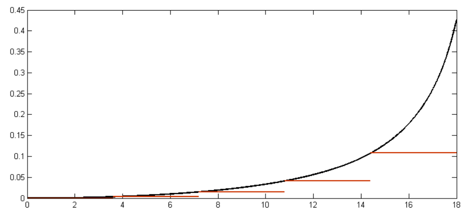
$$\hat{W}(\lambda, c_j) = (\lambda - \hat{\lambda}^{jn}) w(\hat{\lambda}^{j,n+1}, c_j) + \sum_{k=2}^{n} (\hat{\lambda}^{jk} - \hat{\lambda}^{j,k-1}) w(\hat{\lambda}^{jk}, c_j) \qquad \text{if } \lambda \in [\hat{\lambda}^{jn}, \hat{\lambda}^{j,n+1}]$$

$$= \lambda \underbrace{w(\hat{\lambda}^{j,n+1}, c_j)}_{\hat{f}^{jn}} + \underbrace{\sum_{k=2}^{n} (\hat{\lambda}^{jk} - \hat{\lambda}^{j,k-1}) w(\hat{\lambda}^{jk}, c_j) - \hat{\lambda}^{jn} w(\hat{\lambda}^{j,n+1}, c_j)}_{\hat{g}^{jn}} \ . \tag{53}$$

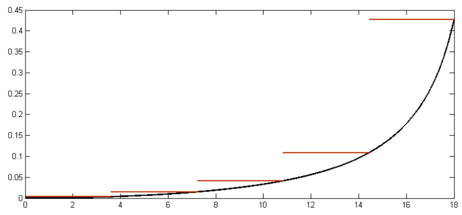
Let $\tilde{w}$ and $\hat{w}$ denote the derivatives of $\tilde{W}$ and $\hat{W}$, respectively. It is easy to check that

$$\begin{aligned} \tilde{w}(\lambda, c) &= w(\tilde{\lambda}^{c,l-1}, c) && \text{if } \lambda \in [\tilde{\lambda}^{c,l-1}, \tilde{\lambda}^{c,l}) \text{ and } l \in \{2, 3, \ldots, N_l\} \\ \hat{w}(\lambda, c_j) &= w(\hat{\lambda}^{jn}, c_j) && \text{if } \lambda \in [\hat{\lambda}^{j,n-1} \hat{\lambda}^{jn}) \text{ and } n \in \{1, 2, \ldots, N_c - 1\} \ . \end{aligned}$$

Then, $\tilde{w}$ and $\hat{w}$ are piecewise constant approximations of w in λ, satisfying (23), as illustrated in Figure 2.



(a) $\tilde{w}$



(b) $\hat{w}$

Figure 2: Piecewise constant approximations of w.

Since w is increasing in λ according to Proposition 3.1, for each couple(λ_j, c_j) we have that $\tilde{w}_j(\lambda_j, c_j) \leq w_j(\lambda_j, c_j)$ and $\hat{w}_j(\lambda_j, c_j) \geq w_j(\lambda_j, c_j)$.

We now replace $\int_0^{\lambda_j} w(\tau, c_j) d\tau$ in (51) with $\tilde{W}_j$ for the leader and $\hat{W}_j$ for the competitor, as defined by (52) and (53). This yields the approximated lower level

$$\begin{aligned} \text{PL2:} \quad \min_{x} \quad & \sum_{i \in I} \sum_{j \in J^*} x_{ij} t_{ij} + \alpha \left[\sum_{j \in J_1^*} \tilde{W}(\lambda, c_j) + \sum_{j \in J_c} \hat{W}(\lambda, c_j) \right] \\ \text{s.t.} \quad & \text{constraints (18), (20), (22), (52)–(53).} \end{aligned}$$

Next, we substitute variables $\tilde{v}$ and $\hat{v}$ with $\tilde{W}$ and $\hat{W}$, respectively. This allows to rewrite PL2 as a linear program, whenever the upper level vector c is fixed:

$$\text{PLL:} \quad \min_{x} \quad \sum_{i \in I} \sum_{j \in J^*} x_{ij} t_{ij} + \alpha \sum_{j \in J_1^*} \tilde{v}_j + \alpha \sum_{j \in J_c} \hat{v}_j$$

$$\text{s.t.} \quad \tilde{v}_j - \tilde{f}_j^{c_j,l} \cdot \lambda_j - \tilde{g}_j^{c_j,l} \geq 0 \qquad j \in J_1^*; l = 1, \ldots, N_l \tag{54}$$

$$\hat{v}_j - \hat{f}^{jn} \cdot \lambda_j - \hat{g}^{jn} \geq 0 \qquad j \in J_c; n = 1, \ldots, N_c \tag{55}$$

$$\sum_{j \in J^*} x_{ij} = d_i \qquad i \in I \tag{56}$$

$$\sum_{i \in I} x_{ij} - \lambda_j = 0 \qquad j \in J^* \tag{57}$$

$$x_{ij} \geq 0 \qquad \forall i \in I; j \in J^*. \tag{58}$$

Upon introduction of the dual variables $\tilde{\pi}_j^l \geq 0$, $\hat{\pi}_j^n \geq 0$, η_i, δ_j and $\phi_{ij} \geq 0$ associated with constraints (54)–(58), we derive the primal-dual optimality conditions

$$t_{ij} + \delta_j + \eta_i - \phi_{ij} = 0 \qquad i \in I; j \in J^*$$

$$
\begin{aligned}
\sum_{l=1}^{N_l-1} \tilde{f}_j^{cl}\tilde{\pi}_j^l - \delta_j &= 0 && j \in J_1^* \\
\sum_{n=1}^{N_c-1} \hat{f}^{jn}\hat{\pi}_j^n - \delta_j &= 0 && j \in J_c \\
\sum_{l=1}^{N_l-1} \tilde{\pi}_j^l &= \alpha && j \in J_1^* \\
\sum_{n=1}^{N_c-1} \hat{\pi}_j^n &= \alpha && j \in J_c \\
\left(\tilde{v}_j - \tilde{f}_j^{c_j,l}\cdot\lambda_j - \tilde{g}_j^{c_j,l}\right)\tilde{\pi}_j^l &= 0 && j \in J_1^*; l = 1,\dots,N_l-1 && (59)\\
\left(\hat{v}_j - \hat{f}^{jn}\cdot\lambda_j - \hat{g}^{jn}\right)\hat{\pi}_j^n &= 0 && j \in J_c; n = 1,\dots,N_c-1 && (60)\\
x_{ij}\phi_{ij} &= 0 && i \in I; j \in J^* && (61)\\
\text{constraints (54) -- (58)}\ .
\end{aligned}
$$

We now replace constraints (16)–(22) in the original formulation CC–FLP with the above optimality conditions. The optimum of the latter program yields an upper bound on the objective function of CC–FLP .

Linear relaxation of bilinear terms

To complete the linearization process, we approximate the bilinear terms $\tilde{f}_j^{c_j,l}\tilde{\pi}_j^l$ and $\tilde{f}_j^{c_j,l}\lambda_j$ by their McCormick envelopes, and linearize the complementarity constraints in Eqs. (59), (60) and (61) using big-M constants. This yields the final MILP formulation

$$
\begin{aligned}
&\text{CC–FLP2:} && && (62)\\
&\max_{y,c} && \sum_{i\in I}\sum_{j\in J_1} x_{ij} \\
&\text{s.t.} && \sum_{j\in J_1}(f_c\cdot y_j + v_c\cdot c_j) \le B \\
& && \lambda_j \le \mu\cdot c_j && j \in J_1\\
& && \lambda_j \le \lambda_{\max} && j \in J_c\\
& && c_j \le M\cdot y_j && j \in J_1\\
& && c_j \ge y_j && j \in J_1\\
& && c_j = \sum_{k=0}^{c_{\max}} k\cdot l_{j,k}^w, && j \in J_1\\
& && \sum_{k=0}^{c_{\max}} l_{j,k}^w = 1 && j \in J_1\\
& && y_j \in \{0,1\},\ l_{j,k}^w \in \{0,1\} && j \in J_1; k = 0,\dots c_{\max}\\
& && c_j \ge 0,\ c_j \le c_{\max},\ c_j \text{ integer} && j \in J_1\\
& && \sum_{j\in J} x_{ij} = d_i && i \in I\\
& && \sum_{i\in I} x_{ij} = \lambda_j && j \in J\\
& && \tilde{v}_j - z_{j,l}^{w\lambda} - \tilde{G}_j^l \ge 0 && j \in J_1; l = 1,\dots,N_l\\
& && \hat{v}_j - \hat{f}^{jn}\cdot\lambda_j - \hat{g}^{jn} \ge 0 && j \in J_c; n = 1,\dots,N_c\\
& && 0 \le \tilde{c}_{jk} \le 1 && j \in J_1; k = 0,\dots,c_{\max} && (63)
\end{aligned}
$$

$$\sum_{k=0}^{c_{\max}} \tilde{c}_{jk} = 1 \qquad j \in J_1 \tag{64}$$

$$c_j = \sum_{k=0}^{c_{\max}} k \cdot \tilde{c}_{jk} \qquad j \in J_1 \tag{65}$$

$$\tilde{f}_j^l = \sum_{k=0}^{c_{\max}} \tilde{c}_{jk} \cdot w(\tilde{\lambda}^{kl}, k) \qquad j \in J; l = 1, \dots, N_l$$

$$\tilde{G}_j^l = \sum_{k=0}^{c_{\max}} \tilde{c}_{jk} \cdot \tilde{g}^{kl} \qquad j \in J; l = 1, \dots, N_l$$

$$z_{j,l}^{w\lambda} \geq w_{\max} \cdot \lambda_j + \tilde{\lambda}_{\text{UB}}^l \cdot \tilde{f}_j^l - w_{\max} \cdot \tilde{\lambda}_{\text{UB}}^l \qquad j \in J_1; l = 1, \dots, N_l$$

$$z_{j,l}^{w\lambda} \leq \tilde{\lambda}_{\text{UB}}^l \cdot \tilde{f}_j^l \qquad j \in J_1; l = 1, \dots, N_l$$

$$z_{j,l}^{w\lambda} \leq w_{\max} \cdot \lambda_j \qquad j \in J_1; l = 1, \dots, N_l$$

$$z_{j,l}^{w\lambda} \geq 0 \qquad j \in J_1; l = 1, \dots, N_l$$

$$x_{i,j} \geq 0 \qquad i \in I; j \in J$$

$$t_{ij} + \delta_j + \gamma_i \geq 0 \qquad i \in I; j \in J$$

$$\sum_{l=1}^{N_l - 1} \tilde{\pi}_j^l = \alpha \qquad j \in J_1 \tag{66}$$

$$\sum_{n=1}^{N_c - 1} \hat{\pi}_j^n = \alpha \qquad j \in J_c \tag{67}$$

$$\sum_{l=1}^{N_l - 1} z_{j,l}^{\pi,f} - \delta_j = 0 \qquad j \in J_1$$

$$\sum_{n=1}^{N_c - 1} \hat{\pi}_j^n \hat{f}^{jn} - \delta_j = 0 \qquad j \in J_c$$

$$z_{j,l}^{\pi,f} \geq \alpha \cdot \tilde{f}_j^l + w_{\max} \pi_{j,l} - w_{\max} \cdot \alpha \qquad j \in J_1; l = 1, \dots, N_l$$

$$z_{j,l}^{\pi,f} \leq w_{\max} \pi_{j,l} \qquad j \in J_1; l = 1, \dots, N_l$$

$$z_{j,l}^{\pi,f} \leq \alpha \cdot \tilde{f}_j^l \qquad j \in J_1; l = 1, \dots, N_l$$

$$z_{j,l}^{\pi,f} \geq 0 \qquad j \in J_1; l = 1, \dots, N_l$$

$$\tilde{\pi}_j^l \leq \alpha \cdot \tilde{s}_{j,l}^{\pi} \qquad j \in J_1; l = 1, \dots, N_l$$

$$\tilde{v}_j - z_{j,l}^{w\lambda} - \tilde{g}_j^l \leq \left(1 - \tilde{s}_{j,l}^{\pi}\right) \cdot M_{\pi} \qquad j \in J_1; l = 1, \dots, N_l$$

$$\hat{\pi}_j^n \leq \alpha \cdot \hat{s}_{j,n}^{\pi} \qquad j \in J_c; n = 1, \dots, N_c$$

$$\hat{v}_j - \hat{f}^{jn} \lambda_j - \hat{g}^{jn} \leq \left(1 - \hat{s}_{j,k}^{\pi}\right) \cdot M_{\pi} \qquad j \in J_c; n = 1, \dots, N_c$$

$$t_{ij} + \delta_j + \gamma_i \leq s_{i,j}^{\phi} \cdot M_{\phi} \qquad i \in I; j \in J$$

$$x_{ij} \leq \left(1 - s_{i,j}^{\phi}\right) D \qquad i \in I; j \in J$$

$$s_{i,j}^{\phi} \in \{0, 1\}, \quad s_{i,j}^{\pi} \in \{0, 1\} \qquad i \in I; j \in J$$

where $\tilde{g}^{cl}$, $\hat{f}^{jn}$, and $\hat{g}^{jn}$ are computed according to (52) and (53), $\lambda_{UB}^l = \max\limits_{c=0,\dots,c_{\max}} (\tilde{\lambda}^{cl})$, and $D = \sum_{i \in I} d_i$. Valid expressions for the big-M constants M_ϕ and M_π are obtained as follows. Eqs. (66) and (67) imply that $\pi_j^l \leq \alpha$ and $\pi_j^n \leq \alpha$. We set

$$\delta_{\max}^L = \sum_{l=1}^{N_l} \max_{c=1,\dots,c_{\max}} \Big(\alpha w(\tilde{\lambda}^{cl}, c)\Big) \ , \ \delta_{\max}^C = \sum_{n=1}^{N_c} \max_{j \in J_c} \Big(\alpha w(\hat{\lambda}^{j,l}, c_j)\Big) \, ,$$

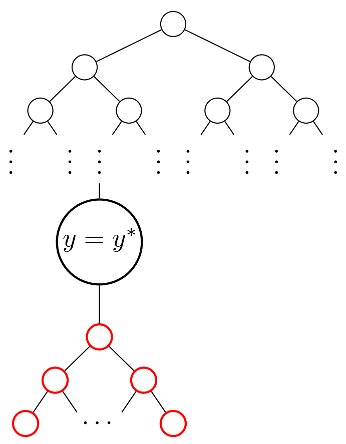
and $\delta_{\max} = \max(\delta^L_{\max}, \delta^C_{\max})$. It follows that $\gamma_i \in [-t_{\max} - \delta_{\max}, 0]$, and $M_\phi = t_{\max} + \delta_{\max}$. Similarly, we set $M_\pi = \max(v^L_{\max}, v^C_{\max})$, where

$$v^L_{\max} = \max_{c=1,\ldots,c_{\max}} \left\{ \max_{l=1,\ldots,N_l} \left(D \cdot w(\tilde{\lambda}^{cl}, c) + \tilde{g}^{cl} \right) \right\}$$
$$v^C_{\max} = \max_{j \in J_c} \left\{ \max_{n=1,\ldots,N_c} \left(D \cdot w(\hat{\lambda}^{jl}, c_j) + \hat{g}^{jn} \right) \right\} .$$

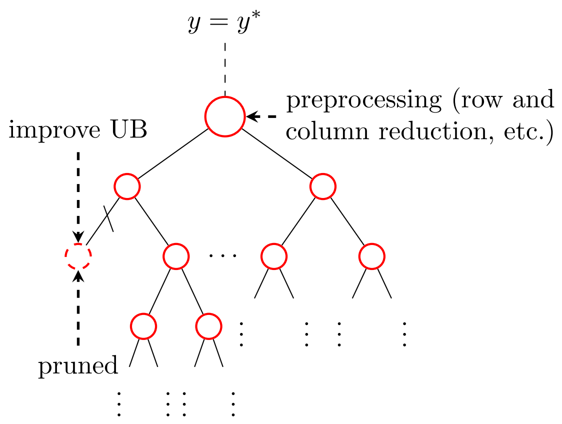
4 Branch-and-Bound Algorithm

This section is devoted to an exact algorithm for CC–FLP that exploits the upper bound on the objective provided by the approximate programs CC–FLP1 and CC–FLP2 while, for a given leader solution (y, c), a lower bound is obtained solving the corresponding lower level program for an equilibrium assignment of users to facilities. Our main issue is that, in sharp contrast with 'standard' B&B, there is a gap between the objective of the true formulation CC–FLP and that of the approximation CC–FLP1. Our aim is to overcome this difficulty through the efficient interaction with the Branch-and-Bound software, through callbacks. While our implementation is based on the IBM CPLEX suite, any software that allows for callbacks could have been used.

Our algorithm is based on a nested B&B tree structure. Nodes of the *main tree* relate to the location variables y_j and are labeled by the y vector. Whenever, at a given node of the main tree, the y-solution of the relaxed problem is integer-valued (such a node is called a 'leaf'), we grow an *inner subtree* (Figure 3) that focuses on the c_j variables (number of servers) and other intermediate variables by defining and solving a new and separate MILP. Due to the gap between the true objective and that of the relaxed problem, 'manual' interaction with the software is required in order not to wrongly fathom nodes of the main tree. In particular, the true value of a leaf's objective must be retrieved before it can be used for fathoming or pruning purposes, both in the main tree and the inner subtrees.



(a) Main B&B tree



(b) Inner B&B subtree

Figure 3: The nested B&B trees.

We now detail the implementation and functionality of the nested structure. As discussed in Section 1, the reason for treating each of those nodes as a separate MILP is to leverage at the same time the CPLEX preprocessing capability and the pieces of information collected until that point in the tree to tighten the formulation of the subproblem originated at this node.

4.1 Main B&B tree

Let CC–FLP1(y^*) and CC–FLP2(y^*) denote the restriction of CC–FLP1 and CC–FLP2 obtained by fixing the location vector y at y^*. The main B&B solves a relaxation of the MILP approximation (CC–FLP1 or CC–FLP2), where the integrality requirement is relaxed on all variables, with the exception of the location vector y. The initial lower and upper bounds for the MILP are computed by the solver. At every integer node, we use the current solution y^*, call the subroutine solving for the optimal number of servers, and append the no-good global cut ([DAmbrosio et al., 2010])

$$\sum_{j\in J_1, y_j^*=0} y_j + \sum_{j\in J_1, y_j^*=1} (1-y_j) \geq 1 \tag{68}$$

to the model. The latter ensures that the current solution does not appear elsewhere in the tree. The rejection of all feasible solutions is required to guarantee the validity of the branching and pruning rules. Indeed, if a solution were not rejected, CPLEX would accept it along with its approximate objective, and its update of bounds might be inconsistent with the values of the true model. Whenever an improved solution is obtained while solving a restricted problem, the corresponding cut is global and allows to update the lower bound, while the incumbent is updated externally.

We have implemented the sequence of operations described above in a *Lazy Constraint Callback*, called at every feasible (integer) node. A feature of this operation is that it is called only and in all feasible integer nodes, and that it allows for appending user cuts that might cut off integer parts of the domain. Its pseudocode is given in Algorithm 2.

Algorithm 2 Lazy Constraint Callback in the main B&B

```
1: $y^* \leftarrow currentSolution$
2: $restrictedBestObj \leftarrow$ Solve CC–FLP1($y^*$) or CC–FLP2($y^*$)
3: if $restrictedBestObj > bestFound$ then
4:     $objectiveCut \leftarrow (z > restrictedBestObj)$
5:     addGlobalCut($objectiveCut$)
6:     $bestFound \leftarrow restrictedBestObj$
7: $yCut \leftarrow Eq.$ (68)
8: addGlobalCut($yCut$)
```

4.2 Inner subtrees

When the approximation CC–FLP1 is used, a subtree is associated with y^* of the main tree. One then solves subproblem CC–FLP1(y^*), to which one appends a set of constraints whose role is to discard feasible nodes, as achieved in the main B&B tree by means of constraints (68). Precisely, for all $j \in J_1,\ k = 0,\ldots,c_{\max}$, we introduce binary variables $\tilde{c}_{jk}$ and consider the following unary representation of c_j.

$$c_j = \sum_{k=0}^{c_{\max}} k \cdot \tilde{c}_{jk} \qquad j \in J_1 \tag{69}$$

$$\sum_{k=0}^{c_{\max}} \tilde{c}_{jk} = 1 \qquad j \in J_1 \tag{70}$$

$$\tilde{c}_{jk} \in \{0,1\} \qquad j \in J_1; k = 0,\ldots,c_{\max}. \tag{71}$$

On the other hand, if CC–FLP2 is solved, no additional constraints are required since Eq. (63)–(65) are already part of the formulation.

At each integer node, the true objective is computed by solving the lower level assignment problem, for instance using the Frank-Wolfe algorithm. If an improved solution is uncovered, the incumbent is saved externally, and the lower bound on the objective is updated through the addition of a global cut. Next, we reject the current integer solution by adding the global cut

$$\sum_{k=0}^{c_{\max}} \sum_{\substack{j\in J_1,\\ \tilde{c}^*_{jk}=0}} \tilde{c}_{jk} + \sum_{k=0}^{c_{\max}} \sum_{\substack{j\in J_1,\\ \tilde{c}^*_{jk}=1}} (1-\tilde{c}_{jk}) \geq 1, \tag{72}$$

thus guaranteeing the consistency of the branching and pruning rules. Eqs. (69)–(71), together with Eq. (72), ensure that the current vector c will not reappear later in the tree. These operations have been implemented in the *Lazy Constraint Callback* described by Algorithm 3.

Algorithm 3 Lazy Constraint Callback in the inner B&B

```
1: $\tilde{c}^* \leftarrow$ current $\tilde{c}$; $c^* \leftarrow$ current $c$
2: $currentObj \leftarrow$ Frank-Wolfe($c^*$)
3: if $currentObj > bestFound$ then
4:     $objectiveCut \leftarrow (z > currentObj)$
5:     $bestFound \leftarrow currentObj$
6:     addGlobalCut($objectiveCut$)
7: $cCut \leftarrow Eq.\ (72)$
8: addGlobalCut($cCut$)
```

In other words, the unary representation of c is necessary to impose the no-good constraints (72), which would be otherwise hard to write for general integer variables.

4.3 Improving the upper bound

At the fractional nodes of an inner B&B subtree, the location variables are fixed, while some c_j variables are fractional. Depending on the node, the lower and upper bounds on the integer variables might have been improved by the branching decisions taken so far. Let c_j^{UB} denote the upper bound on variable c_j, at a given fractional node in the subtree. Then we have the following result.

Proposition 4.1 *At a fractional node, let us set $c_j \leftarrow c_j^{UB}$, for all $j \in J_1$. Then, the objective value associated with this solution is a valid upper bound on the true objective, in the subtree rooted at that node.*

Proof. Since c_j^{UB} exceeds any value that c_j may achieve in the subtree rooted at the current node, it follows from Proposition 3.1 that

$$w(\lambda, c_j^{UB}) \leq w(\lambda, c_j) \quad \forall j \in J_1. \tag{73}$$

Since the competitors' service rate remains unchanged, the conclusion is a direct consequence of Proposition 3.3. □

In most cases, fixing the number of server variables may be infeasible, due to the budget constraint. Let $c_{\text{TOTAL}} = B - |J_1^*| \cdot f_c$ be the leader's available budget at the current node, to be distributed among facilities. At a a fractional node, it is very likely that $\sum_{j\in J_1} c_j^{UB} > c_{\text{TOTAL}}$, due to a number of factors. One empirically expects that, deep into the enumeration tree, bounds are tight and yield feasible solutions.

Improved upper bounds have been implemented in a *User Cut Callback* that is invoked at every fractional node. Within the callback, we retrieve the upper bounds on variables c_j and set the current solution to those bounds. If condition (74) below holds, we compute the objective value associated with this solution, by evaluating the associated equilibrium flows, i.e., by solving PL. If this operation improves the bound provided by CPLEX, we append it to the model in the form of a local user cut as described in Algorithm 4.

When computed close to the root of the tree, bounds tend to be loose, and the probability of improving over the CPLEX bound is small. On the other hand, when computed deep into the tree, the tightness of the bounds improve, but only a small portion of the tree is pruned. In our implementation, the improvement procedure has been activated whenever the inequality

$$\sum_{j \in J_1^*} c_j^{UB} \leq c_{\text{TOTAL}} + q \cdot |J_1^*| \tag{74}$$

held. In (74), q plays the role of a flexibility or frequency parameter, to be tuned offline. When $q = \infty$ the upper bound is computed at all fractional nodes, while if $q = 0$, it is only computed at the leaves of the tree.

Algorithm 4 User Cut Constraint Callback in the restricted B&B

```
1: $c^{UB} \leftarrow$ vector of upper bounds with respect to $c$
2: if Eq. (74) then
3:     $currentObj \leftarrow$ Frank-Wolfe($c^{UB}$)
4:     CPLEXUB $\leftarrow$ upper bound provided by CPLEX
5:     if CPLEXUB> $currentObj$ then
6:         $ubCut \leftarrow (z \leq currentObj)$
7:         addLocalCut($ubCut$)
```

4.4 Computing a lower bound

The performance of the exact method can be improved by computing a good lower bound at the root of the B&B tree. The underlying idea is to linearize the queueing terms at the leader's facilities, without the introduction of binary variables. More specifically, in the sampling scheme described in Section 3.3, let us introduce triangles based on consecutive samples, as detailed in [D'Ambrosio et al., 2010]. The upper triangles are defined by the points $(\tilde{\lambda}^{k,n}, k)$, $(\tilde{\lambda}^{k+1,n}, k+1)$ and $(\tilde{\lambda}^{k+1,n+1}, k+1)$, while the lower triangles are defined by $(\tilde{\lambda}^{k,n}, k)$, $(\tilde{\lambda}^{k,n+1}, k)$ and $(\tilde{\lambda}^{k+1,n+1}, k+1), n = 1, \ldots, N_l, k = 1, \ldots, c_{\max} - 1$. The coefficient of the plane equations associated with the lower and upper triangles are, for every $1 \leq n \leq N_l$ and $1 \leq k \leq c_{\max} - 1$

$$\begin{aligned}
\underline{u}_1^{kn} &= \tilde{\lambda}^{k+1,n} - \tilde{\lambda}^{k,n} & \overline{u}_1^{kn} &= \tilde{\lambda}^{k+1,n} - \tilde{\lambda}^{k+1,n+1} \\
\underline{u}_2^{kn} &= 1 & \overline{u}_2^{kn} &= 0 \\
\underline{u}_3^{kn} &= \underline{w}^{k+1,n} - \underline{w}^{k,n} & \overline{u}_3^{kn} &= \underline{w}^{k+1,n} - \underline{w}^{k+1,n+1} \\
\underline{v}_1^{kn} &= \tilde{\lambda}^{k,n+1} - \tilde{\lambda}^{k,n} & \overline{v}_1^{kn} &= \tilde{\lambda}^{k,n+1} - \tilde{\lambda}^{k+1,n+1} \\
\underline{v}_2^{kn} &= 0 & \overline{v}_2^{kn} &= -1 \\
\underline{v}_3^{kn} &= \underline{w}^{k,n+1} - \underline{w}^{k,n} & \overline{v}_3^{kn} &= \underline{w}^{k,n+1} - \underline{w}^{k+1,n+1} \\
\underline{a}^{kn} &= \underline{u}_2^{kn} \cdot \underline{v}_3^{kn} - \underline{u}_3 kn \cdot \underline{v}_2^{kn} & \overline{a}^{kn} &= \overline{u}_2^{kn} \cdot \overline{v}_3^{kn} - \overline{u}_3 kn \cdot \overline{v}_2^{kn} \\
\underline{b}^{kn} &= \underline{u}_3^{kn} \cdot \underline{v}_1^{kn} - \underline{u}_1 kn \cdot \underline{v}_3^{kn} & \overline{b}^{kn} &= \overline{u}_3^{kn} \cdot \overline{v}_1^{kn} - \overline{u}_1 kn \cdot \overline{v}_3^{kn} \\
\underline{c}^{kn} &= \underline{u}_1^{kn} \cdot \underline{v}_2^{kn} - \underline{u}_2 kn \cdot \underline{v}_1^{kn} & \overline{c}^{kn} &= \overline{u}_1^{kn} \cdot \overline{v}_2^{kn} - \overline{u}_2 kn \cdot \overline{v}_1^{kn} \\
\underline{d}^{kn} &= -\left(\underline{a}^{kn}\tilde{\lambda}^{k,n} + \underline{b}^{kn}k + \underline{c}^{kn}\underline{w}^{kn}\right) & \overline{d}^{kn} &= -\left(\overline{a}^{kn}\tilde{\lambda}^{k+1,n+1} + \overline{b}^{kn}(k+1) + \overline{c}^{kn}\underline{w}^{k+1,n+1}\right) .
\end{aligned} \tag{75}$$

The plane equations defined by the lower and upper triangles are

$$\underline{a}^{kn}\lambda_j + \underline{b}^{kn}c_j + \underline{c}^{kn}w_j + \underline{d}^{kn} = 0 \ , \quad \text{and} \quad \overline{a}^{kn}\lambda_j + \overline{b}^{kn}c_j + \overline{cb}^{kn}w_j + \overline{d}^{kn} = 0 \ .$$

Next, we convexify w_j's by setting them to the maximum of their respective linear approximations, namely

$$w_j \approx \max_{\substack{k=1,\ldots,c_{\max}-1 \\ n=1,\ldots N-l}} \left\{ \max\left(-\frac{\underline{a}^{kn}}{\underline{c}^{kn}}\lambda_j - \frac{\underline{b}^{kn}}{\underline{c}^{kn}}c_j - \frac{\underline{d}^{kn}}{\underline{c}^{kn}},\ -\frac{\overline{a}^{kn}}{\overline{c}^{kn}}\lambda_j - \frac{\overline{b}^{kn}}{\overline{c}^{kn}}c_j - \frac{\overline{d}^{kn}}{\overline{c}^{kn}} \right) \right\} \ . \tag{76}$$

From the leader's point of view, the lower the waiting time, the more customers will be attracted to her facilities. This allows to replace Eq. (76) by the inequalities

$$\begin{aligned} w_j &\geq -\frac{\underline{a}^{kn}}{\underline{c}^{kn}}\lambda_j - \frac{\underline{b}^{kn}}{\underline{c}^{kn}}c_j - \frac{\underline{d}^{kn}}{\underline{c}^{kn}} \quad k=1,\ldots,c_{\max}-1, n=1,\ldots,N-l \\ w_j &\geq -\frac{\overline{a}^{kn}}{\overline{c}^{kn}}\lambda_j - \frac{\overline{b}^{kn}}{\overline{c}^{kn}}c_j - \frac{\overline{d}^{kn}}{\overline{c}^{kn}} \quad k=1,\ldots,c_{\max}-1, n=1,\ldots,N-l\ . \end{aligned} \tag{77}$$

We can now write CC–FLP as the following MILP

$$\begin{aligned} \text{CC–FLPH:} \quad \max_{y,c,x,\gamma} \quad & z = \sum_{i\in I}\sum_{j\in J_1} x_{ij} \\ \text{s.t.} \quad & \text{constraints (11)–(15), (18), (21), (22), (26)–(29),} \\ & \text{(43)–(50), (69)–(71), (75), (77).} \end{aligned}$$

Note that, for the competitor it would be ill-advised to write a linearization similar to (76) – (77), since increasing w at competitor's facilities would actually increase the objective value (of the leader). At optimality, the waiting time at competitor facilities would be set to very high values, in an attempt to maximize the objective, and no competitor inequality would be active, yielding a very poor approximation. Additionally, we maintain the presence of binary variables for the competitor, which are limited in number and hence do not increase significantly the difficulty of the model. Of course, the heuristic scheme can be used both for computing a standalone approximate solution for CC–FLP and as warm start for its exact solution. The approximate model is then solved by the nested B&B strategy.

We close this section by mentioning that the improved 'on-the-fly' upper bounds are not implemented in the heuristic, as they significantly slow down the exploration process.

5 Experimental setup and results

The MILP formulations were solved by IBM CPLEX Optimizer version 12.6. All tests were performed on a computer equipped with 96 GB of RAM, and two 6-core Intel(R) Xeon(R) X5675 processors running at 3.07GHz. To assess the performance of our algorithm, we could not compare with alternative methods from literature, which are nonexistent, as discussed in Section 3.1. The model of [Marianov et al., 2008] involves facility location within a user-choice environment, but the decision variables are limited to the locations, while a heuristic is designed for its solution. This lack of alternatives prompted us to compute an optimal solution through exhaustive search, iterating through all possible combinations of number of servers that satisfy the budget constraint. A feasible flow is then computed to yield a feasible solution to the bilevel program. Some solutions are discarded without computing the flow, for the following reason. At any open facility, the arrival rate needs to be lower that the service level. Thus, if the total number of servers times the service rate is lower than the incumbent, the current solution cannot yield a better objective value, and we have no incentive to compute the lower-level optimum.

Our experiments involve networks of varying sizes, and travel times ranging from 0 to 5000. In order to generate challenging instances, the combination of budget, fixed and variable costs was selected such as to ensure the existence of feasible solutions involving a number of open facilities roughly equal to half the total number, and where the number of servers could reach 20 on small tests, and 40 or more on larger tests. Note that, at optimality less than half facilities are typically open, and service levels do not reach their upper limits. In all our tests we used 5 λ-samples for both the leader and the competitor.

An initial set of experiments was intended to probe the efficiency of linearizations CC–FLP1 and CC–FLP2, respectively, for various values of parameter q. We show only the most successful results, namely $q = 15$ for the single level linearization, and $q = 25$ for the bilevel linearization. At the end of this section we provide a deeper analysis of the impact of this parameter on the overall performance of our algorithm.

Tables 1 and 2 display the results for 15 and 20-node networks, respectively, which are compared for reference with the full enumeration scheme. The exact methods were tested with and without a warm start (columns 'w/o warm start' and 'warm start', respectively). The warm start is provided by the best solution found by heuristic model CC–FLPH within a time limit. In the case of 15-node networks, the warm start improves significantly the running time. It is not needed, however for the 20-node networks tested, as the methods perform well without it.

	single level linearization ($q = 15$)				bilevel linearization ($q = 25$)				
	w/o warm start		warm start		w/o warm start		warm start		
test #	total	opt.	total	opt.	total	opt.	total	opt.	enumeration
1	3199	577	6002	3615	87791	58457	81183	50708	170825
2	51242	44696	17237	11485	209778	168134	86522	23866	190268
3	110719	100898	38231	31913	619715	494483	1295419	1208952	1873586
4	47678	40360	11536	2423	132524	866	132761	32420	194206
5	26830	21353	14422	5552	670075	482895	227470	64780	624795
6	42051	39583	5500	2975	148006	112378	161920	126093	1419919
7	11985	11778	1268	600	2106842	2106820	73981	71867	11460
8	6840	5674	3738	3300	21935	15422	18854	15872	7098
9	10694	8798	6324	3118	147756	98541	78980	21630	300081
10	12424	10320	8474	6215	217631	202242	49474	28254	767436
Average	29951	28404	11273	7120	436205	374024	220656	164444	555967

Table 1: CPU times (seconds) on 15-node networks; 'opt.' refers to the CPU required to find an optimal solution.

	single level linearization $q = 15$		bilevel linearization $q = 25$		
test #	total	opt.	total	opt.	enumeration
1	1358	1327	1705	1687	27660
2	36	8	234	209	5892
3	550	90	1504	821	8233
4	667	211	2880	2816	149672
5	307	66	3833	3553	26183
6	338	34	1644	1495	27212
7	1301	416	2265	1673	165337
8	495	406	2438	2401	42301
9	383	265	9823	9766	9571
10	2149	1614	2555	2102	280145
11	1024	958	4789	4718	19249
12	3404	2513	4573	2913	191448
13	305	184	1654	1579	10321
14	332	105	1372	1256	20674
15	383	222	1343	1189	242524
16	1450	313	5401	4090	168118
17	2203	307	4042	3429	309729
18	2149	1132	4018	3609	727332
19	427	137	1633	1002	9731
20	1462	392	2592	1947	191918
Average	1036	535	3015	2608	131663

Table 2: CPU times (seconds) on 20-node networks; 'opt.' refers to the CPU required to find an optimal solution. The methods were not warm started.

Those initial results confirm the stability and performance of the algorithm when adopting the single level linearization scheme. This might be due to the use of McCormick envelopes. Under both schemes, the algorithm outperforms by two orders of magnitude the enumeration-based method. The single level linearization outperforms clearly the bilevel (McCormick) linearization on most 15 and 20-node instances.

We ran the same set of experiments on 25-node networks, and report the results in Table 3. All tests were warm started with the heuristic since, without it, the instances were intractable, with CPLEX running out of memory quickly. The enumeration scheme failed on all tests after running more than 15 weeks. The single level linearization performs faster than the bilevel counterpart on the fraction of tests that can be solved.

Actually, the algorithm has to stop for lack of memory on more than half of the instances, which shows the limitation of our exact method. In that respect, the bilevel scheme looks slightly more robust, being able to solve 8 instances, versus 6 in the single level case. We note, however, that the budget allows for up to 12 facilities to be open, and that the number of servers can vary between 1 and 55, which makes for a very challenging class of problems.

	single level linearization $q = 15$		bilevel linearization $q = 25$	
test #	total	opt.	total	opt.
1	37.4	6.0	53.8	35.7
2	–	–	–	–
3	–	–	–	–
4	–	–	–	–
5	–	–	–	–
6	–	–	–	–
7	–	–	613.8	17.9
8	–	–	–	–
9	–	–	–	–
10	51.3	6.5	550.5	12.5
11	–	–	–	–
12	47.2	14.1	–	–
13	–	–	563.8	31.1
14	60.1	6.4	–	–
15	–	–	78.0	29.1
16	–	–	176.0	102.0
17	21.7	0.6	125.7	35.2
18	16.2	3.3	535.7	453.2
19	–	–	–	–
20	–	–	–	–
Average	39.9	6.15	337.2	89.6

Table 3: CPU times (**hours**) on 25-node networks; 'opt.' refers to the CPU required to find an optimal solution.

The aim of the second set of experiments is to assess the accuracy of the linearization of CC–FLP1, as shown in Table 5. In this process, the MILP is solved using B&B, without using callbacks. Once CPLEX has reached and proved optimality, Frank-Wolfe's algorithm is used to retrieve its corresponding true objective value (column 'objective') and compare it with the approximated objective value ('approximated') and with the actual optimum (column 'optimal') computed by our exact algorithm. We observe that, on average, the objective value of the solution found falls within 3.6% of the optimum. Additionally solving only the MILP yields 89.17% of the optimal locations, which suggests that the approximation can be used as a good heuristic on its own, as well. The behaviour of the approximation on the 15-node networks (see Table 4) is similar, capturing on average 94.4% of the optimum, but finding only 68.3% of the optimal facilities.

test #	MILP obj.	MILP true obj.	CPU (s)	optimum	deviation (%)
1	213.57	192.29	1707	204.88	6.2
2	200.24	187.19	66915	193.02	3.1
3	151.82	138.89	158976	147.87	6.1
4	223.58	201.84	2668	212.97	5.2
5	191.29	174.53	91936	182.48	4.4
6	184.76	174.48	11288	180.61	3.4
7	230.41	210.43	27107	218.27	3.6
8	240.06	210.79	5481	220.54	4.4
9	191.69	158.74	4855	185.01	14.2
10	194.07	176.40	5911	186.02	5.2
Average	202.15	182.56	37684	193.17	5.5

Table 4: Comparison of the best solution found when solving only the approximation, versus the optimal solution, for 15-node networks.

test #	MILP obj.	MILP true obj.	CPU (s)	optimum	deviation (%)
1	287.80	270.30	150	273.55	1.2
2	330.46	312.57	38	323.56	3.4
3	287.86	256.04	102	269.00	4.9
4	296.43	264.94	333	291.95	9.2
5	293.42	277.26	89	287.57	3.6
6	274.25	252.97	146	264.37	4.3
7	268.33	257.52	58	259.14	0.6
8	273.35	258.01	48	270.19	4.6
9	293.83	262.12	68	290.98	9.9
10	248.21	237.36	106	248.21	4.3
11	269.47	249.53	40	261.39	4.6
12	252.33	242.09	71	244.63	1.1
13	291.50	280.73	47	284.58	1.3
14	282.79	264.19	77	271.94	2.9
15	262.29	259.89	237	261.64	0.7
16	242.04	225.69	130	242.04	6.7
17	253.63	252.74	103	253.63	0.3
18	221.47	205.36	583	216.55	5.1
19	289.47	273.95	376	283.10	3.2
20	245.59	245.59	125	245.59	0.0
Average	273.23	257.44	146	267.18	3.6

Table 5: Comparison of the best solution found when solving only the approximation, versus the optimal solution, for 20-node networks.

In the third set of experiments, we demonstrate why it is advantageous to use the nested B&B tree structure described in Section 4. First we used a single B&B solving for all variables, with only the no-good cuts within the associated callback, and without computing the on-the-fly upper bound. In this case the problem became quickly intractable, even for small instances. For example, on the 9-node networks that we have tested, a single tree takes more than 3 hours and still does not prove optimality. In comparison, the nested tree structure takes less than 7 minutes. One reason for this is the preprocessing at the root nodes of the subtrees. In order to investigate this further, we have measured the objective value of every leaf of the *main tree* ('Original node' in Figure 4), which is an upper bound for the subtree rooted at this node, since the main tree is solving a relaxed problem. We then generated the respective *inner subtree* and we retrieved the bounds both after presolve and after solving the root node. If the subtree was found infeasible at the root node, the bound is shown as 0. Our measurements were taken on ten 9-node instances, for a total of 2129 explored subtrees, out of which 328 were cut off or found infeasible at the root node. As shown in Figure 4, we observe a significant improvement in the upper bound, after presolve and solving the root node.

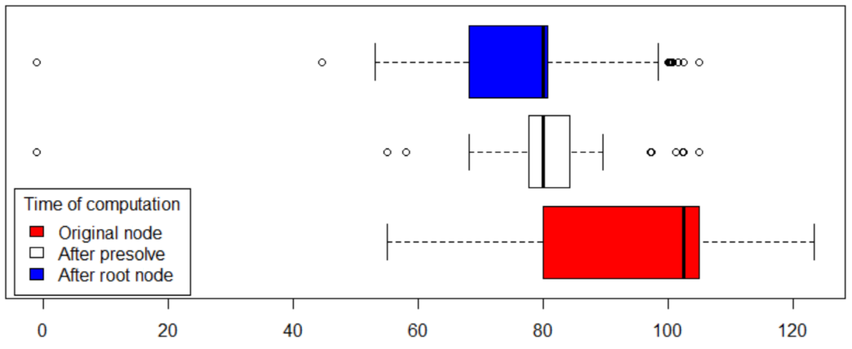


Figure 4: Distribution of the upper bounds throughout the execution of the algorithm. 'Original node' is as computed at integer nodes in the main tree. 'After presolve' and 'After Root Node' show the upper bound computed after the presolve and after solving the root node of the inner subtrees, respectively.

The following measures were also computed:

- average improvement in UB after presolve: 7.26%
- average improvement in UB after root node: 10.28%
- average LP columns reduction: 52.56
- average LP rows reduction: 47.88
- average MIP columns reduction: 156.32
- average MIP rows reduction: 16.99.

Our measurements demonstrate that by creating nested trees, rather than using a single tree, we make full use of the heuristics, cuts, rows and columns reductions, and other computations that CPLEX performs during presolve and at the root nodes. The problem becomes thus more tractable, even without the computation of the on-the-fly upper bounds.

It is important to note that the order in which the nodes are explored differ between the nested tree and single tree approach, which could impact the efficiency of the algorithm. When using nested trees, no new nodes in the main tree will be processed until the current subtree is solved. This particular behaviour cannot be easily achieved in a single tree, even if we prioritize branching on the location variables.

The fourth experiment was designed to assess the impact of the on-the-fly upper bounds on the overall performance of the algorithm. Since, without the strengthening of the UB, even small instances are intractable, we have limited our analysis to 9-node instances, using the single-level linearization. Figure 5 illustrates the typical evolution of the upper bounds and the best objective function, throughout the execution of our methods. Notice that the upper bound in the main tree does not improve fast, while the bounds on the subtrees vary significantly from one subtree to the other.

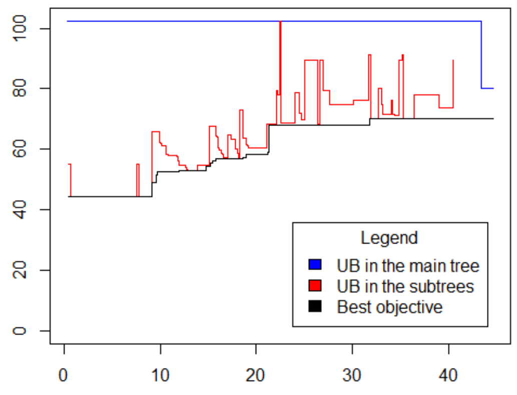


Figure 5: Typical evolution of the upper bounds and best objective during the execution of our algorithm.

We have measured the average execution time ('CPU'), the number of integer solution explored ('# of sols'), and the number of integer nodes pruned ('# of cuts'), for ten 9-node instances, as shown in Table 6. We notice that, as q increases, the average CPU has a convex-like behaviour (decreasing, levelling, increasing),

	w/o q	q=3	q=12	q=20	q=30	q=50
CPU (s)	306.7	87.4	55.4	69.7	72.2	90.3
# of sols	11049	773	512	512	512	512
# of cuts	89741	100017	100278	100278	100278	100278

Table 6: Performance of the single-level linearization for different values of q, for 9-node networks.

which is to be expected. As q increases, the on-the-fly upper bound is computed at more nodes, and higher in the subtrees. Therefore, at first, the number of integer solutions visited decreases, as more cuts are computed and more nodes are pruned. However, for large values of q, the bound is computed often (it is a costly operation), and does not improve on the bound provided by CPLEX. Thus, the CPU increases, while the number of nodes pruned stalls. It is therefore ill-advised to set q to a large value. In our tests, values between 10 and 15 were most successful for the single level linearization, and around 25 for the bilevel linearization.

6 Conclusion

The MPEC framework allows the modelling of situations that are highly relevant in practice. However, the resulting mathematical program highly challenging combinatorial and nonlinear features, which explains the frequent recourse to heuristic (meta-heuristics, math-heuristics) for its solution, and the paucity of exact methods. Nevertheless, we expect that generic algorithms that exploit MILP approximations of single-level reformulations, as well as a clever management of the B&B tree, deserve some consideration. We hope that the present work, which may be viewed as a step in that direction, will trigger further research in global approaches to bilevel programs involving a lower level variational inequality.

References

[Abouee-Mehrizi et al., 2011] Abouee-Mehrizi, H., Babri, S., Berman, O., and Shavandi, H. (2011). Optimizing capacity, pricing and location decisions on a congested network with balking. *Mathematical Methods of Operations Research*, 74(2):233–255.

[Andreani and Martínez, 2001] Andreani, R. and Martínez, J. M. (2001). On the solution of mathematical programming problems with equilibrium constraints. *Mathematical Methods of Operations Research*, 54(3):345–358.

[Baumrucker et al., 2008] Baumrucker, B., Renfro, J., and Biegler, L. (2008). MPEC problem formulations and solution strategies with chemical engineering applications. *Computers & Chemical Engineering*, 32(12):2903–2913.

[Beckmann et al., 1956] Beckmann, M., McGuire, C. B., and Winsten, C. B. (1956). *Studies in the Economics of Transportation*. Yale University Press, New Haven.

[Belotti et al., 2016] Belotti, P., Bonami, P., Fischetti, M., Lodi, A., Monaci, M., Nogales-Gmez, A., and Salvagnin, D. (2016). On handling indicator constraints in mixed integer programming. *Computational Optimization and Applications*, 65:545–566.

[Berman and Drezner, 2006] Berman, O. and Drezner, Z. (2006). Location of congested capacitated facilities with distance-sensitive demand. *IIE Transactions*, 38(3):213–221.

[Camacho-Vallejo et al., 2014] Camacho-Vallejo, J.-F., Cordero-Franco, A. E., and González-Ramírez, R. G. (2014). Solving the bilevel facility location problem under preferences by a Stackelberg-evolutionary algorithm. *Mathematical Problems in Engineering*, 2014:14.

[D'Ambrosio et al., 2010] D'Ambrosio, C., Lodi, A., and Martello, S. (2010). Piecewise linear approximation of functions of two variables in MILP models. *Operations Research Letters*, 38(1):39–46.

[Dan and Marcotte, 2017] Dan, T. and Marcotte, P. (2017). Competitive facility location with selfish users and queues. Technical Report 46, CIRRELT. Accepted for publication in Operations Research.

[Desrochers et al., 1995] Desrochers, M., Marcotte, P., and Stan, M. (1995). The congested facility location problem. *Location Science*, 3(1):9–23.

[DAmbrosio et al., 2010] DAmbrosio, C., Frangioni, A., Liberti, L., and Lodi, A. (2010). On interval-subgradient and no-good cuts. *Operations Research Letters*, 38(5):341 – 345.

[Furini and Traversi, 2013] Furini, F. and Traversi, E. (2013). Hybrid SDP Bounding Procedure. In Bonifaci, V., Demetrescu, C., and Marchetti-Spaccamela, A., editors, *Experimental Algorithms*, pages 248–259, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Hijazi and Liberti, 2014] Hijazi, H. and Liberti, L. (2014). Constraint qualification failure in second-order cone formulations of unbounded disjunctions. Technical report, NICTA, Canberra ACT Australia.

[Jeroslow and Lowe, 1984] Jeroslow, R. and Lowe, J. (1984). Modelling with integer variables. In Korte, B. and Ritter, K., editors, *Mathematical Programming at Oberwolfach II, Volume 22 of Mathematical Programming Studies*, pages 167–184. Springer, Berlin.

[Kleinrock, 1975] Kleinrock, L. (1975). *Theory, Volume 1, Queueing Systems*. Wiley-Interscience.

[Lee and Cohen, 1983] Lee, H. L. and Cohen, M. A. (1983). A note on the convexity of performance measures of M/M/c queueing systems. *Journal of Applied Probability*, 20(4):920923.

[Marianov et al., 2008] Marianov, V., Ríos, M., and Icaza, M. J. (2008). Facility location for market capture when users rank facilities by shorter travel and waiting times. *European Journal of Operational Research*, 191(1):32–44.

[Marić et al., 2012] Marić, M., Stanimirović, Z., and Milenković, N. (2012). Metaheuristic methods for solving the bilevel uncapacitated facility location problem with clients' preferences. *Electronic Notes in Discrete Mathematics*, 39(0):43 – 50. EURO Mini Conference.

[Raghunathan, 2013] Raghunathan, A. (2013). Global optimization of nonlinear network design. *SIAM Journal on Optimization*, 23:268–295.

[Sun et al., 2008] Sun, H., Gao, Z., and Wu, J. (2008). A bi-level programming model and solution algorithm for the location of logistics distribution centers. *Applied Mathematical Modelling*, 32(4):610 – 616.

[Vidyarthi and Jayaswal, 2014] Vidyarthi, N. and Jayaswal, S. (2014). Efficient solution of a class of location-allocation problems with stochastic demand and congestion. *Computers & Operations Research*, 48(0):20–30.

[Zhang et al., 2010] Zhang, Y., Berman, O., Marcotte, P., and Verter, V. (2010). A bilevel model for preventive healthcare facility network design with congestion. *IIE Transactions*, 42(12):865–880.