Revenue optimization in energy networks involving self-scheduled demand and a smart grid

Sezin Afsar, Luce Brotcorne, Patrice Marcotte, Gilles Savard

Version of Record: https://www.sciencedirect.com/science/article/pii/S0305054821001416
Manuscript_3d7335fe23de2496d056b53ae184f80d

Revenue optimization in energy networks involving self-scheduled demand and a smart grid

Sezin Afşar[a], Luce Brotcorne[b,*], Patrice Marcotte[c], Gilles Savard[d]

[a] *Department of Information Technology, University of Oviedo, Campus de Gijòn, 33204 Gijòn, Spain*
[b] *Inria Lille-Nord Europe, 40 Avenue Halley 59650 Villeneuve d'Ascq, France*
[c] *DIRO, Université de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal (QC) H3C 3J7, Canada*
[d] *Département de Mathématique et Génie Industriel, École Polytechnique, C.P. 6079, Succursale Centre-Ville, Montréal (QC) H3C 3A7, Canada*

Abstract

In this paper, we address a day-ahead pricing and load balancing problem, within an environment involving self-scheduled users whose utilities are optimized via a smart grid. The model is formulated as a mixed integer bilevel program for which we propose a single level reformulation. Next, we design two heuristic algorithms which, by relying on the structure of the problem, provide high quality solutions in moderate computation time.

Keywords: mixed integer bilevel programming, demand response, day ahead pricing

1. Introduction

In a context of large variations of demand over time, together with the rise of renewable power generation, the goal of demand response is to help the operators of a power grid in balancing supply and demand with respect to predefined criteria [9, 10, 11]. For one, an operator must consider the role played by electricity consumers who may reduce or shift their electricity

*Corresponding author. Tel: +33 359358628
Email addresses: afsarsezin@uniovi.es (Sezin Afşar), luce.brotcorne@inria.fr (Luce Brotcorne), marcotte@iro.umontreal.ca (Patrice Marcotte), gilles.savard@polymtl.ca (Gilles Savard)

usage in response to time-based rates or other forms of financial incentives. In this framework we extend the Pricing and Load Balancing Problem (PLB) considered in a previous work [1], by including both preemptive and non-preemptive devices. This is in contrast with [1], where only preemptive devices, i.e., devices that be turned on or off at any time, were considered.

In our modelling approach, the PLB involves two decision levels with conflicting objectives. At the upper level, an energy provider sets a dynamic price schedule that should achieve an optimal trade-off between revenues and peak load, while, at the lower level, customers react by scheduling their loads according to price signals and comfort requirements. Since customers are inter-connected via smart metering devices (automatic energy consumption scheduler [8]), they not only share an energy source, but also communicate with each other via a network of smart meters which form core of the smart grid. The bilevel framework allows the leader to set the right prices to trigger a behavioral change among customers, with the aim to decrease the peak load and hence reduce capacity installment costs, while presenting the customers with the option of decreasing their electricity bill without compromising their comfort levels. In an operational context, this methodological framework enables energy providers with a valuable decision-making tool for the investigation of innovative demand management strategies, notably in the tricky issue of balancing revenue and load charge.

This class of pricing problems involving a hierarchical decision making process fits the framework of bilevel programming, where the follower's optimization problem is integrated into the leader's decision-making process. In the context of PLB the leader is the energy provider and the follower is the smart grid in charge of scheduling the customer devices.

Relevant contributions that adopt this framework have been considered by [1, 3, 4, 7]. The authors of [7] consider a single leader, multi-follower game where, for a given price signal, customers engage into a Nash game. They propose for its solution a genetic algorithm that optimizes the leader's objective; for each price signal, linear problems are solved independently according to the different categories of devices considered at the lower level. Numerical results on medium size instances are promising in terms of both load reduction and costs for the customers. In [1, 3, 4] customers are inter-connected via smart metering devices to an aggregator in charge of defining a schedule that minimizes the total cost of all customers. The leader of the bilevel program is thus the energy provider, and the follower is the aggregator acting as a middle agent. In [4], the authors develop both an evolutionary algorithm

and a particle swarm optimization metaheuristic to solve the problem. Numerical results are discussed for small instances and compared to a strategy where an exact solution of the lower level program is integrated within the population based approach. In [3], these authors propose a semi-vectorial extension that considers two criteria at the lower level, namely dynamic time of use and prices. Among the possible selections of Pareto-optimal outcomes at the lower level, they consider both the best and worst-case scenarios for the leader. They propose for their solution a hybrid approach consisting in a genetic algorithm for the upper level problem, and an exact method to solve scalarized problems at the lower level. The algorithm is implemented on data relevant to a small-scale, real-world application.

Let us now outline the main difference between our approach and these works. In [7] the lower level program models the competitive behaviour of customers, in contrast with customers that are inter-connected via smart metering devices to allow automated scheduling. In [3, 4] the objective function of the leader only involves revenue optimization, defined as the difference between the revenue arising from the sale of energy to the aggregator and the cost of buying the electricity (on the spot market), and does not consider peak control. In [4] no inconvenience factor is considered for the customers, while in [3] it is included as a second objective at the lower level, leading to the study of a bilevel model involving a lower level bi-objective program. Rather, in our model, we consider both the cost and inconvenience factors minimization by integrating them into a single weighted objective function. While [1] only considers preemptive devices, the integration of non-preemptive devices requires the inclusion of binary variables into the customers scheduling programs, resulting in a mixed bilinear bilevel program that is numerically challenging. From the computational side, we assess the quality of our heuristics versus an optimal solution that can be obtained by considering small instances, or by allowing large running times. In contrast, no guarantee of closeness to optimality is provided in [3, 4, 7].

In summary, the contribution of the paper is twofold. First, we present a bilevel formulation of PLB that allows the energy provider to achieve an optimal trade-off between peak load and revenue, while taking into account customers' preferences with respect to the scheduling of their appliances. Next, we develop efficient solution procedures tailored to the structure of the formulation. In particular we exploit the fact that, although the modelling of non-preemptive devices requires integer binary variables at the lower level, it is yet possible to formulate the bilevel program as a single level mixed integer

linear program.

The rest of the paper is organized as follows: a bilevel formulation of PLB with preemptive and non-preemptive devices is presented in Section 2; a new reformulation of the bilevel problem as a mixed integer program is provided in Section 3; two efficient heuristic procedures that exploit the structure of the problem are detailed in Section 4; experimental results, including sensitivity analysis with respect to key parameters, are presented in Section 5; in the concluding section, extensions worth investigating are outlined.

2. The bilevel model

Let us consider a power sharing system involving a set of customers, each one equipped with a smart meter, who operate preemptive and (or) non-preemptive devices, within predetermined time slots and over a finite horizon (a week, say). Each device is characterized by its power load, the duration of its task, and an operating time window. Given this demand environment, the objective of the system operator is to set up a pricing policy that maximizes the difference between revenue and a penalty associated with peak load. The notation is as follows:
Throughout the paper, it is assumed that the initial time slot of a time window is the preferred one, and that a job cannot be performed outside its window, i.e., x^h_{n,a_1} and y^h_{n,a_2} are defined only for $h \in T_{n,a_1}$ and $h \in T_{n,a_2}$, respectively. Whenever a job is postponed with respect to the preferred time slot, a penalty that is proportional to the length of the delay, and inversely proportional to the width of the desired time window, is incurred. The latter assumption reflects the fact that customers that specify narrow time windows are likely to be sensitive to the delaying of their tasks by the smart grid operator. Without loss of generality the same reasoning can be applied for anticipation of the task.

Let λ_n denote the inconvenience coefficient of a customer n, a high value of λ_n being related to a low tolerance to delay. The inconvenience parameters $C_{n,a_1}(h)$ and $C_{n,a_2}(h)$ are defined as :

parameters

H	set of time slots
$p^h_{\max}$	maximum price for time slot h
κ	weight factor between revenue and load
A^1_n	set of preemptive devices for customer n
A^2_n	set of non-preemptive devices for customer n
$\beta^{\max}_{n,a_1}$	power limit of device $a_1 \in A^1_n$ for customer n
x^h_{n,a_1}	power level for each device $a_1 \in A^1_n$ in time slot h for customer n
E_{n,a_1}	power demand for device $a_1 \in A^1_n$
k_{n,a_2}	fixed power load for device $a_2 \in A^2_n$
l_{n,a_2}	fixed operating time for device $a_2 \in A^2_n$
$T_{n,a_i} = [TW^b_{n,a_i}, TW^e_{n,a_i}]$	time window of customer n for device $a_i \in A^i_n$
λ_n	delay sensitivity coefficient of customer n

variables

p^h (upper level)	price for time slot h
Γ (upper level)	maximum peak load over the horizon H
x^h_{n,a_1} (lower level)	power level for device $a_1 \in A^1_n$ at time slot h
y^h_{n,a_2} (lower level)	binary variable equal to 1 if device $a_2 \in A^2_n$ is started at time slot h .

Table 1: Notation

$$C_{n,a_1}(h) := \lambda_n \times E_{n,a_1} \times \frac{h - TW^b_{n,a_1}}{TW^e_{n,a_1} - TW^b_{n,a_1}} \qquad n \in N, a_1 \in A^1_n, h \in T_{n,a_1}$$

$$C_{n,a_2}(h) := \lambda_n \times k_{n,a_2} \times l_{n,a_2} \times \frac{h - TW^b_{n,a_2}}{TW^e_{n,a_2} - TW^b_{n,a_2}} \quad n \in N, a_2 \in A^2_n, h \in T_{n,a_2}.$$

The bilevel mathematical model of PLB considering preemptive and non-preemptive devices is then expressed as the mathematical program MixPLB (mixed devices) whose formulation is :

MixPLB:

$$
\begin{aligned}
\max_{p,\Gamma} \quad & \sum_{\substack{n\in N\\ a_2\in A_n^2\\ h\in T_{n,a_2}}} k_{n,a_2} y^h_{n,a_2} \sum_{h'=h}^{h+l_{n,a_2}} p^{h'} + \sum_{\substack{n\in N\\ a_1\in A_n^1\\ h\in T_{n,a_1}}} p^h x^h_{n,a_1} - \kappa\Gamma \\
\text{s.t.} \quad & \Gamma \geq \sum_{\substack{n\in N\\ a_2\in A_n^2}} \sum_{\substack{h'=h-l_{n,a_2}\\ h'\in T_{n,a_2}}} k_{n,a_2} y^{h'}_{n,a_2} + \sum_{\substack{n\in N\\ a_1\in A_n^1\\ h\in T_{n,a_1}}} x^h_{n,a_1} && h\in H \\
& 0 \leq p^h \leq p^h_{\max} && h\in H
\end{aligned}
$$

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{\substack{n\in N\\ a_2\in A_n^2\\ h\in T_{n,a_2}}} \left(k_{n,a_2} \sum_{h'=h}^{h+l_{n,a_2}} p^{h'} + C_{n,a_2}(h) \right) y^h_{n,a_2} \\
& \qquad + \sum_{\substack{n\in N\\ a_1\in A_n^1\\ h\in T_{n,a_1}}} \left(p^h + C_{n,a_1}(h) \right) x^h_{n,a_1} \\
\text{s.t.} \quad & 0 \leq x^h_{n,a_1} \leq \beta^{\max}_{n,a_1} && n\in N, a_1\in A_n^1, h\in T_{n,a_1} && (1)\\
& \sum_{h\in T_{n,a_1}} x^h_{n,a_1} \geq E_{n,a_1} && n\in N, a_1\in A_n^1 && (2)\\
& \sum_{h\in T_{n,a_2}} y^h_{n,a_2} \geq 1 && n\in N, a_2\in A_n^2 && (3)\\
& y^h_{n,a_2} \in \{0,1\} && n\in N, a_2\in A_n^2, h\in T_{n,a_2} \quad . && (4)
\end{aligned}
$$

At the upper level, the leader strives for a trade-off between revenue and peak load by maximizing the sum of revenue minus a penalty associated with peak consumption. This is achieved through a direct control of prices p^h, for each time slot $h \in H$, and indirect control of the peak load Γ resulting from the lower level solution. The first constraint of the upper level problem forces the variable Γ to exceed the load in each time slot. Since it is in the interest of the leader to minimize peak cost, Γ should actually match the maximum load at optimality. The component of Γ corresponding to preemptive devices is simply the power used at time h whereas the component associated with non-preemptive devices needs to consider the power of each device started before h and still in use. The second set of constraint sets an upper bound on the prices, and might result from government regulation or market conditions.

At the lower level, the objective function involves two terms, namely electricity bill (EB) and inconvenience cost (IC). For a given price vector set by the leader, the smart grid aggregator minimizes the objective of each individual customer defined as a weighted sum of EB and IC, which clearly conflicts with the objective of the leader. To maximize its utility, customer n selects power levels x^h_{n,a_1} and y^h_{n,a_2}, for each device $a_1 \in A^1_n$ and $a_2 \in A^2_n$ respectively, in each time slot $h \in T_{n,a_1}$ and $h \in T_{n,a_2}$. Constraint (1) is a technical constraint that limits the maximum amount of power that an device may consume, while demand satisfaction is enforced by constraints (2) and (3).

All customers being residential users, it is natural to assume that prices only depend on time slot h. Note that if price discrimination among the customers were allowed, then the problem of the leader would be much easier from a computational point of view, since it would decompose into n independent bilevel programs. Unfortunately, in the model studied in this paper, significant modifications of the schedule may result from a single price change, even for a single slot, which makes the problem computationally challenging.

Finally, if the solution to the lower level problem is not unique, i.e., the follower is indifferent to more than one solution, we assume that the most favorable solution for the leader is selected. This is referred to the optimistic approach, and is justified by the fact that the lower level solution can be made unique through an arbitrarily small perturbation of the follower's objective function.

3. Single level formulation

MixPLB is a bilinear bilevel model involving continuous variables x and binary variables y. Since no constraint links the two types of variables, and since both terms are minimized in the objective function, note that the follower problem is separable. We now show how to transform MixPLB into a single-level program. For readability considerations, we consider separately the preemptive part of the problem (related to variables x) and the non-preemptive part (related to variables y).

The model corresponding to preemptive devices is a bilevel bilinear bilinear problem with continuous variables. As performed in [1], the classical reformulation of a bilevel program as a single level mixed integer program (MIP) applies. In this approach, the lower level's optimality conditions (primal feasibility, dual feasibility, complementary slackness) are linearized and

appended to the upper level to yield an equivalent MIP formulation.

The situation is different when non-preemptive devices are considered, since their inclusion involves, at the lower level, an assignment problem (over a bipartite graph) that requires binary variables for its formulation. Given that the latter program is totally unimodular, i.e., vertex solutions of its linear programming relaxation are integer-valued, it is tempting to relax the integrality constraints on the y variables, and apply the technique used in the preemptive case. However, as illustrated in the next example, this approach may yield a non-integer solution to the bilevel program. Indeed, let us consider the instance built around a single non-preemptive device and two time slots. Let $k = 10$, $l = 1$, $p = (10, 8)$, $C = (0, 20)$ and $\kappa = 5$. Let us consider the two nontrivial lower level solutions:

- $y = (1, 0)$: the total cost for the follower is $(10 \times 10) + 0 = 100$ and the net revenue for the leader is $(10 \times 10) - (5 \times 10) = 50$;
- $y = (0, 1)$: the total cost for the follower is $(8 \times 10) + (20 \times 1) = 100$ and the net revenue of the leader is $(8 \times 10) - (5 \times 10) = 30$.

However, if $y = (0.5, 0.5)$, the total cost for the follower is $(5 \times 10) + (5 \times 8) + (20 \times .5) = 100$, and the net revenue of the leader, $(10 \times 5) + (8 \times 5) - (5 \times 5) = 65$, exceeds that of the integer solutions.

To enforce the integrality of the optimistic solution of MIP, the next results states that it suffices to move the lower level integrality constraints to the upper level.

Theorem 1. *The bilevel program obtained by moving integrality constraints (4) of MixPLB to the upper level and relaxing them in the lower level is equivalent to MixPLB.*

Proof. In general, moving constraints from the lower to the upper level of a bilevel program restricts the feasible domain of the leader. However, since the lower level assignment problem associated with non-preemptive devices is totally unimodular, there always exist lower level optimal solutions that meet the integrability requirements and these, according to the 'optimistic' assumption, can be selected by the leader. □

Theorem 1 yields, in a straightforward fashion, a single-level reformulation of MixPLB, achieved by relaxing the lower-level integrability requirements, writing the optimality conditions of the resulting linear programming,

and re-introducing into this primal-dual system the integrability constraints. In this process, the bilinear term px can be linearized by using the equality of the primal and dual objective functions, as was done in [1]. This yields, for the non-preemptive part of the lower-level program, the set of constraints

$$\sum_{h \in T_{n,a_2}} y^h_{n,a_2} \geq 1 \qquad n \in N, a_2 \in A^2_n$$

$$v_{n,a_2} - w^h_{n,a_2} \leq k_{n,a_2} \sum_{h'=h}^{h+l_{n,a_2}} p^{h'} + C_{n,a_2}(h) \qquad n \in N, a_2 \in A^2_n, h \in T_{n,a_2}$$

$$k_{n,a_2} \sum_{h'=h}^{h+l_{n,a_2}} p^{h'} + C_{n,a_2}(h) - v_{n,a_2} + w^h_{n,a_2} \leq M_1(1 - y^h_{n,a_2}) \qquad n \in N, a_2 \in A^2_n, h \in T_{n,a_2}$$

$$w^h_{n,a_2} \leq M_2 y^h_{n,a_2} \qquad n \in N, a_2 \in A^2_n, h \in T_{n,a_2}$$

$$y^h_{n,a_2} \in \{0,1\}, w^h_{n,a_2} \geq 0 \qquad n \in N, a_2 \in A^2_n, h \in T_{n,a_2}$$

leading to the following MIP formulation of MixPLB:

MixMIP:

$$\max_{x,y,v,w,\Gamma} \quad \sum_{\substack{n\in N\\ a_2\in A_n^2\\ h\in T_{n,a_2}}} v_{n,a_2} - \sum_{\substack{n\in N\\ a_2\in A_n^2\\ h\in T_{n,a_2}}} w^h_{n,a_2} - \sum_{\substack{n\in N a_1\in A_n^1\\ h\in T_{n,a_1}}} \beta^{\max}_{n,a_1} w^h_{n,a_1} + \sum_{\substack{n\in N\\ a_1\in A_n^1}} E_{n,a_1} v_{n,a_1}$$
$$- \sum_{\substack{n\in N\\ a_2\in A_n^2\\ h\in T_{n,a_2}}} C_{n,a_2}(h) y^h_{n,a_2} - \sum_{\substack{n\in N\\ a_1\in A_n^1\\ h\in T_{n,a_1}}} C_{n,a_1}(h) x^h_{n,a_1} - \kappa\Gamma$$

$$\text{s.t.} \quad \Gamma \geq \sum_{\substack{n\in N\\ a_2\in A_n^2}} \sum_{\substack{h'=h-l_{n,a_2}\\ \text{s.t.} h'\in T_{n,a_2}}}^{h} k_{n,a_2} y^h_{n,a_2} + \sum_{\substack{n\in N\\ a_1\in A_n^1\\ \text{s.t.} h\in T_{n,a_1}}} x^h_{n,a_1} \qquad \forall h \in H$$

$$0 \leq p^h \leq p^h_{\max} \qquad h \in H$$

$$0 \leq x^h_{n,a_1} \leq \beta^{\max}_{n,a_1} \qquad n \in N, a_1 \in A_n^1, h \in T_{n,a_1}$$

$$\sum_{h\in T_{n,a_1}} x^h_{n,a_1} \geq E_{n,a_1} \qquad n \in N, a_1 \in A_n^1$$

$$\sum_{h\in T_{n,a_2}} y^h_{n,a_2} \geq 1 \qquad n \in N, a_2 \in A_n^2$$

$$-w^h_{n,a_1} + v_{n,a_1} - p^h \leq C_{n,a_1}(h) \qquad n \in N, a_1 \in A_n^1, h \in T_{n,a_1}$$

$$-w^h_{n,a_2} + v_{n,a_2} - k_{n,a_2} \sum_{h'=h}^{h+l_{n,a_2}} p^{h'} \leq C_{n,a_2}(h)$$
$$n \in N, a_2 \in A_n^2, h \in T_{n,a_2}$$

$$k_{n,a_2} \sum_{h'=h}^{h+l_{n,a_2}} p^{h'} + C_{n,a_2}(h) - v_{n,a_2} + w^h_{n,a_2} \leq M_1(1 - y^h_{n,a_2})$$
$$n \in N, a \in A_n^2, h \in T_{n,a_2}$$

$$w^h_{n,a_2} \leq M_2 y^h_{n,a_2} \qquad n \in N, a \in A_n^2, h \in T_{n,a_2}$$

$$-x^h_{n,a_1} \leq M_3(1 - \xi^h_{n,a_1}) - \beta^{\max}_{n,a_1} \qquad n \in N, a \in A_n^1, h \in T_{n,a_1}$$

$$w^h_{n,a_1} \leq M_3 \xi^h_{n,a_1} \qquad n \in N, a \in A_n^1, h \in T_{n,a_1}$$

$$\sum_{h\in T_{n,a_1}} x^h_{n,a_1} \leq M_4(1 - \epsilon_{n,a_1}) + E_{n,a_1} \qquad n \in N, a \in A_n^1$$

$$v_{n,a_1} \leq M_4 \epsilon_{n,a_1} \qquad n \in N, a \in A_n^1$$

$$
\begin{aligned}
& w^h_{n,a_1} - v_{n,a_1} + p^h \leq M_5(1-\psi^h_{n,a_1}) - C_{n,a_1}(h) && \\
& && n \in N, a \in A^1_n, h \in T_{n,a_1} \\
& x^h_{n,a_1} \leq M_5 \psi^h_{n,a_1} && n \in N, a \in A^1_n, h \in T_{n,a_1} \\
& y^h_{n,a_2} \in \{0,1\} && n \in N, a_2 \in A^2_n, h \in T_{n,a_2} \\
& \xi^h_{n,a_1}, \psi^h_{n,a_1} \in \{0,1\} && n \in N, a \in A^1_n, h \in T_{n,a_1} \\
& \epsilon_{n,a_1} \in \{0,1\} && n \in N, a \in A^1_n.
\end{aligned}
$$

4. Heuristic methods

Being generically non-convex and non-differentiable, due to the complementarity constraints implicit in the first order optimality conditions of the lower level program, bilevel programs are computationally challenging. Therefore, in order to be able to solve medium to large size instances in moderate computation time, we need to design heuristic solution methods that exploit the structure of the problem. In this section, we introduce two such algorithms: the Price Heuristic PH and the Peak Search Heuristic PSH, each one based on combining in different fashions subproblems embedded in the structure of MixPLB. These are (i) the *inverse optimization problem*, that consists in finding optimal prices compatible with a given schedule and thus a given peak, (ii) the *minimum peak problem*, that computes a feasible peak-minimizing price schedule, and (iii) the *fixed peak problem*, that computes a schedule that must not exceed a predetermined load value. For simplicity, all subproblems are described in terms of non-preemptive devices ($\overline{\text{Pr}}$PLB). The same procedure can be applied to preemptive devices.

4.1. Inverse optimization (IO)

As defined in [2], inverse optimization 'consists of inferring the values of the model parameters, given the values of observable parameters'. In an optimization context, this procedure computes values of the model parameters that induce a predetermined optimal solution. Adapted to a bilevel context, one computes optimal upper level decisions that are compatible with a given lower level solution [5, 6]. In this variant of inverse optimization the parameters (prices) are actually decision variables, and a measure of proximity to the solution to be replicated is replaced by the leader's objective.

The IO formulation of $\overline{\text{Pr}}$PLB, denoted Inv$\overline{\text{Pr}}$, is the linear program (with respect to variables p) defined as:

$$
\begin{aligned}
\text{Inv}\overline{\text{Pr}}: \quad \max_{p} \quad & \sum_{n\in N}\sum_{a\in A_n}\sum_{h\in T_{n,a}} k_{n,a}\tilde{y}^h_{n,a}\sum_{h'=h}^{h+l_{n,a}} p^{h'} \\
\text{s.t.} \quad & 0 \le p^h \le p^h_{\max} \qquad\qquad h \in H \\
& w^h_{n,a} = \begin{cases} \ge 0 & \text{if } \tilde{y}^h_{n,a} = 0 \\ = 0 & \text{if } \tilde{y}^h_{n,a} = 1 \end{cases} \\
& \qquad \text{for all } n \in N, a \in A_n \text{ and } h \in T_{n,a} \text{ such that} \\
& \qquad k_{n,a}\sum_{h'=h}^{h+l_{n,a}} p^{h'} + C_{n,a}(h) - u_{n,a} + w^h_{n,a} \begin{cases} \ge 0 & \text{if } \tilde{y}^h_{n,a} = 0 \\ = 0 & \text{if } \tilde{y}^h_{n,a} = 1 \end{cases}.
\end{aligned}
$$

Note that, for fixed prices between 0 and $p_{\max}$, the lower level problem MixPLB becomes a linear program which can be easily solved. It means that any price vector can result in a feasible lower level solution, i.e., a corresponding schedule. However, the opposite is not true. There may not exist a price vector compatible with a given feasible schedule.

4.2. Minimum peak problem

The minimum peak problem consists in computing a schedule that minimizes the peak load, irrespective of price decisions and inconvenience costs. Its mathematical formulation is:

$$
\begin{aligned}
\text{Mpp}\overline{\text{Pr}}: \quad \min_{\Gamma, y} \quad & \Gamma \\
\text{s.t.} \quad & \Gamma \ge \sum_{\substack{n\in N \\ a\in A_n}} \sum_{\substack{h'=h-l_{n,a} \\ h'\in T_{n,a}}}^{h} k_{n,a} y^{h'}_{n,a} \qquad h \in H \\
& \sum_{h\in T_{n,a}} y^h_{n,a} = 1 \qquad\qquad n \in N, a \in A_n \\
& y^h_{n,a} \in \{0,1\} \qquad\qquad n \in N, a \in A_n, h \in T_{n,a}.
\end{aligned}
$$

Note that, since customer choice is not taken into account, the optimal schedule computed by the minimum peak subproblem might be price infeasible.

4.3. Fixed peak problem (FP)

The fixed peak algorithm computes a schedule that minimizes the total inconvenience cost (IC) under a fixed peak capacity constraint, without taking prices into account. Since setting the peak load to an arbitrary value might yield an unfeasible problem, we instead specify an interval of admissible values $[\Gamma', \Gamma'']$. Note that, since the lower level is not totally unimodular any more, the integrality constraint cannot be relaxed. This yields

$$
\begin{aligned}
\text{Peak}\overline{\text{Pr}}: \quad & \min_{y} \quad \sum_{n\in N}\sum_{a\in A_n}\sum_{h\in T_{n,a}} C_{n,a}(h)y^h_{n,a} \\
& \text{s.t.} \quad \Gamma' \le \sum_{n\in N}\sum_{a\in A_n}\sum_{\substack{h'=h-l_{n,a}\\ h'\in T_{n,a}}}^{h} k_{n,a}y^h_{n,a} \le \Gamma'' \quad h\in H \\
& \qquad \sum_{h\in T_{n,a}} y^h_{n,a} = 1 \qquad\qquad n\in N, a\in A_n \\
& \qquad y^h_{n,a}\in\{0,1\} \qquad\qquad n\in N, a\in A_n, h\in T_{n,a}\ .
\end{aligned}
$$

Although prices are not included in the model, it is important to note that the solution of this subproblem is price-feasible if the peak load is fixed to a suitably large value, i.e., if there always exists a price vector compatible with this schedule. However, if the peak load is set too low (e.g., below the lower bound computed by the Minimum Peak subproblem), the schedule might assign to some jobs IC values larger than their price ceiling.

4.4. Price heuristic (PH)

The rationale behing the price heuristic is to generate high revenue for the leader through iteratively generating a sequence of prices and corresponding follower schedules. More precisely, for fixed prices, a schedule is computed by solving the lower level problem; next, optimal prices associated with the schedule are computed by solving the IO problem. If there are several price vectors corresponding to the same schedule, the one most favourable to the leader is selected. The iterative procedure is halted whenever no improvement is achieved.

Precisely, let x_j, y_j, p_j denote a feasible solution, Γ_j the corresponding peak load, and i the time slot (*peak slot*) associated with Γ_j. The algorithm is composed of four main steps. First, the prices associated with k time slots following peak slot i are decreased by a fixed fraction d, i.e., are multiplied

by $1-d$, and we let $\tilde{p}_j$ denote the updated price vector. Second, the lower level problem is solved with respect to $\tilde{p}_j$, yielding the new schedule $(\tilde{x}_j, \tilde{y}_j)$. Third, the inverse optimization problem is solved to determine the optimal prices compatible with $(\tilde{x}_j, \tilde{y}_j)$. Finally the current best solution is updated, whenever required.

initialization
– solve m instances of the lower level problem with respect to randomly generated prices, and derive the corresponding schedules
– select the schedule (p_0, x_0, y_0) with largest leader' objective z_0
– $(\bar{p}, \bar{x}, \bar{y}) \leftarrow (p_0, x_0, y_0)$
– $\bar{z} \leftarrow z_0$
– let ϵ be a small number (tolerance), $d \in (0,1)$ and $j_{\max}$ the maximum number of iterations
– $j \leftarrow 1$

while $j \leq j_{\max}$ **do**
1. let i be the time slot when the peak load Γ is achieved with respect to (x_{j-1}, y_{j-1}), and $k \in \{0, \ldots, H-i\}$ and set $\tilde{p}_j \leftarrow (p^0_{j-1}, \ldots, p^i_{j-1}, (1-d)p^{i+1}_{j-1} \ldots, (1-d)p^{i+k}_{j-1}, p^{i+k+1}_{j-1}, \ldots, p^H_{j-1})$
2. solve the lower level problem to obtain $(\tilde{x}_j, \tilde{y}_j)$
3. perform IO with respect to $(\tilde{x}_j, \tilde{y}_j)$ to obtain (p^*_j, z^*_j)
4. **if** $z^*_j > \bar{z}$ **then**
   let $(\bar{p}, \bar{x}, \bar{y}, \bar{z}) \leftarrow (p^*_j, \tilde{x}_j, \tilde{y}_j, z^*_j)$, $(p_j, x_j, y_j, z_j) \leftarrow (p^*_j, \tilde{x}_j, \tilde{y}_j, z^*_j)$ and $j \leftarrow j+1$
   else
   if $(1-\epsilon)\bar{z} \leq z^*_j$ **then** $(p_j, x_j, y_j, z_j) \leftarrow (p^*_j, \tilde{x}_j, \tilde{y}_j, z^*_j)$ and $j \leftarrow j+1$
   else $j \leftarrow j_{\max}+1$
   end

end

post-processing: solve MixMIP from $(\bar{p}, \bar{x}, \bar{y}) \rightarrow$ final schedule

Algorithm 1: Price heuristic (PH)

We next detail the initialization phase and justify why an MIP procedure is required at the end of the iterative process, where the *base case* (BC) refers to setting prices are at their upper bounds, and scheduling jobs at the beginning of their respective time windows. At an iteration of PH, prices are set to the values that occur before or at peak slot, and randomly generated following the peak slot, according to a uniform distribution between 0 and $p_{\max}$. The lower level problem is solved with these prices. Next, the optimal prices corresponding to the lower level solution are computed by solving Inv. At the end of the process and in order to guarantee an optimistic solution, the solution with the best leader's objective function is selected as initial solution to MixMIP, which is then solved by an off-the-shelf solver, under a time limit constraint.

4.5. Peak search heuristic (PSH)

A preliminary analysis (see Figure 2 in Section 5) confirms the intuition that the net revenue is *roughly* pseudo-concave with respect to peak. In other words the leader is willing to give up on some revenue to decrease peak up to a threshold level. Any peak value lower than this threshold is offset by a loss of revenue, resulting in a decrease of the objective function. The rational behind the heuristic is to find an optimal peak value through binary search (see Figure 3).

First, an upper and lower bounds on peak loads are computed; these define the feasible peak interval. Next a *combing* procedure is performed to narrow down the peak interval and isolate the value leading to the highest leader's revenue. More precisely, the peak interval is divided into equal subintervals and, for each fixed peak value bounding the interval, a feasible schedule, together with the optimal compatible prices, are obtained by solving InvMix. Two new peak values with the highest leader's objective value are selected, thus defining a narrower search interval. Next a binary search is performed on the current interval to determine the *best* peak value, i.e., the one corresponding to the best schedule and associated prices. For each peak value, the fixed peak problem yields a schedule, and InvMix is invoked to define the optimal compatible prices and the objective. The iterative process halts when the width of the search interval falls below some predetermined value.

Similar to the price heuristic, the solution computed at the end of the iterative process may not be the one with highest revenue for the leader, i.e., there might exist another schedule with the same inconvenience cost and

peak load that yields a larger revenue. To address this issue, the incumbent solution of the algorithm is fed as initial solution to MixMIP, which provides an optimistic solution.

initialization

– let Γ_L and Γ_U be lower and upper bounds on peak loads
– combing procedure: determine f peak values Γ_i located at (Γ_L - Γ_U)/f distance in the interval [Γ_L, Γ_U]
– for each Γ_i :

- solve FP to obtain x_i and y_i
- solve IO to obtain (p_i, z_i^*)

– let Γ_a and Γ_b be the two peak values associated with highest revenue z_a^* and z_b^*.

while $|\Gamma_b - \Gamma_a| \geq \epsilon$ **do**
  1. $\Gamma_m \leftarrow (\Gamma_a + \Gamma_b)/2$
  2. solve FP with respect to Γ_m to obtain x_m and y_m
  3. solve IO with respect to (x_m, y_m) to obtain (p_m^*, z_m^*)
  4. **if** $z_m^* > z_a^*$ **then**
    $(p_a, x_a, y_a, z_a) \leftarrow (p_m^*, x_m, y_m, z_m^*)$
  else
    $(p_b, x_b, y_b, z_b) \leftarrow (p_m^*, x_m, y_m, z_m^*)$
  end
end

post-processing

solve MixMIP from (p_m^*, x_m, y_m, z_m^*) to obtain final schedule

Algorithm 2: Peak search heuristic (PSH)

5. Experimental Results

In this section we assess the performance and scalability of the price (PH) and peak search (PSH) heuristics. In particular, we demonstrate the relevance of the additional step implemented at the end of each heuristic, as well as the impact of the combing procedure for PSH. A significant part of the numerical results is devoted to sensitivity analyses with respect to the penalty parameter κ associated with peak demand. In this section for simplicity reasons, we denote by MixPLB the resolution of the model by a commercial solver.

5.1. Instances' characteristics and preliminary observations

The mixed integer formulation MixMIP of MixPLB together with the heuristics developed in this paper, have been tested on randomly generated instances. Restricted to preemptive devices (respectively non-preemptive devices), the simplified version of MixPLB is denoted PrPLB (respectively $\overline{\text{Pr}}$PLB). The size of initial small instances has been set according to the following data:

$\overline{\text{Pr}}$PLB:	12 customers $\times$ 5 devices
PrPLB:	7 customers $\times$ 3 devices
MixPLB:	7 customers $\times$ 5 devices
	(3 preemptive and 2 non-preemptive).

We observed that the complexity of each instance is influenced by the schedules of the devices more than by their nature. This prompted us to generate instances that induce scheduling conflicts through the introduction of a scaling parameter.

Time windows are characterized by their width TWW and by MCT, the minimum number of time slots required to meet demand $E_{n,a}$ if all devices are set at their maximum level $\beta_{n,a}^{\max}$. MCT is set to $l_{n,a}$ for non-preemptive devices, and to $\lceil E_{n,a}/\beta_{n,a}^{\max} \rceil$ for preemptive devices. The early time slot of the time window for customer n and device a, $TW_{n,a}^{b}$ is generated within 0 and $24 - \lceil (1 + TWW) \times MCT \rceil$. The end of the time window for customer n and device a, $TW_{n,a}^{e}$, is set to $TW_{n,a}^{b} + \lceil (1 + TWW) \times MCT \rceil$. Parameters $\lambda_{n,a}$, $\gamma_{n,a}$, $E_{n,a}$, $k_{n,a}$, $l_{n,a}$ have been generated according to a uniform distribution over the intervals, and are displayed in Table 2. The values of $p_{\max}^{h}$ have been fixed for all models and instances to 100.

parameters	intervals
$\lambda_{n,a}$	[1, 5]
$\gamma_{n,a}$	[4, 12]
$E_{n,a}$	[12, 48]
$k_{n,a}$	[8, 15]
$l_{n,a}$	[2, 9]

Table 2: parameter intervals

Initial tests have been performed to assess the impact of the post-processing procedure implemented at the end of the heuristics. Since the behavior of the procedure is identical for both heuristics, we focus on the price heuristic. As illustrated in Figure 1, post-processing yields improved solutions, with respect to both the leader's objective function (higher) and peak load (smaller).

The impact of the combing step of the peak heuristic is illustrated on Figure 2, that represents the variation of the leader's objective with respect to peak demand, on two instances. More precisely, at the end of the combing procedure, the peak values 42 and 43 lead to the highest objective function values, respectively 18704 and 18734. Binary search, initiated at the interval [42, 43], then computes the objective function value associated to the midpoint 19196. The incumbent is updated and the interval [42.5, 43] becomes the search interval. The objective function value associated with the midpoint (42.75) is equal to 18956, and the next search interval is [42.5, 42.75]. The algorithm continues until the stopping criterion is satisfied.

A key observation, which heuristically warrants our use of binary search in PSH, is the fact that the value of the leader's objective as a function of peak load is nearly pseudo-concave. This property actually holds over the entire feasibility interval for Instance 1 and fails over a short interval (periods 34 and 35) for Instance 2.

The search continues until the stopping criteria is satisfied.

5.2. Results on medium size instances

The mixed integer programs occurring in MixMIP, PS and PSH are solved with CPLEX version 12.3 on a computer with 2.66 GHz Intel Xeon CPU and 4 GB ram, running on the Windows 7 operating system, with a time limit of 4 hours set for MixMIP. For the instances that are not solved to optimality

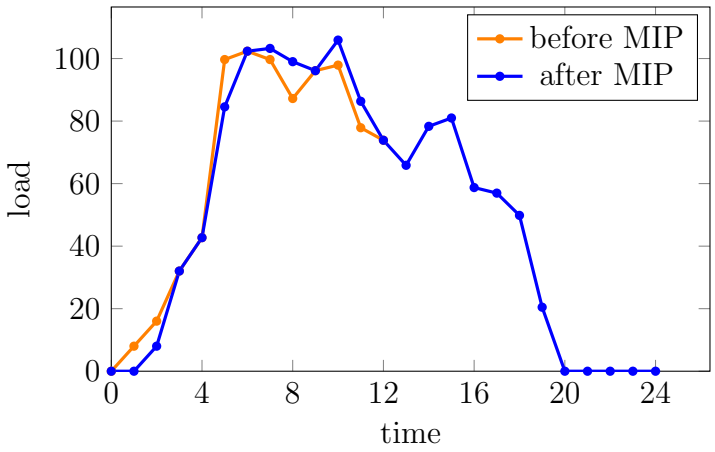
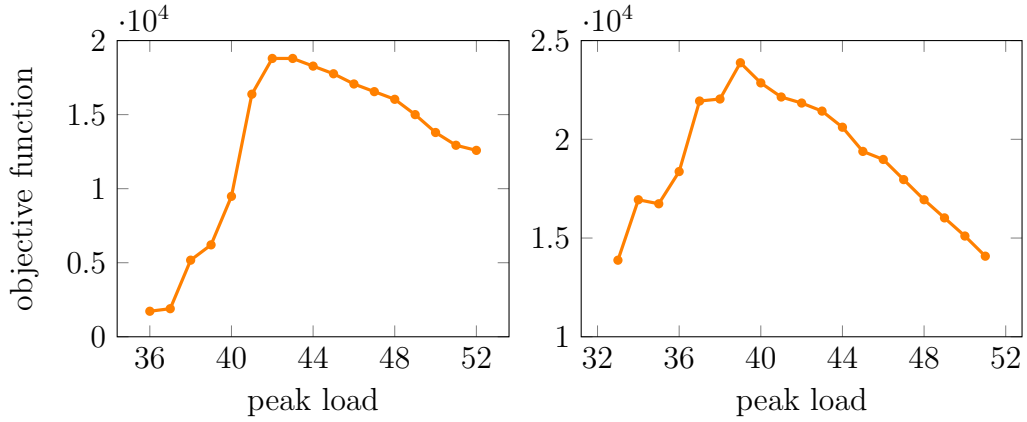


Figure 1: load curves before and after the MixMIP procedure of the post-processing step (PH and PSH)



(a) Instance 1

(b) Instance 2

Figure 2: net revenue versus peak after combing

within this limit, the heuristic solutions are compared to the best integer solution computed by CPLEX. The computation time limit of the MixMIP post-processing step at the end of PSH and PH is set to 150 seconds. The performance of MixMIP, PSH and PH with respect to peak cost, peak load, net revenue and computation time are compared to the base case (BC) corresponding to setting all prices to $p_{\max}$ and scheduling all devices to their preferred time slot. Note that for instances involving only preemptive de-

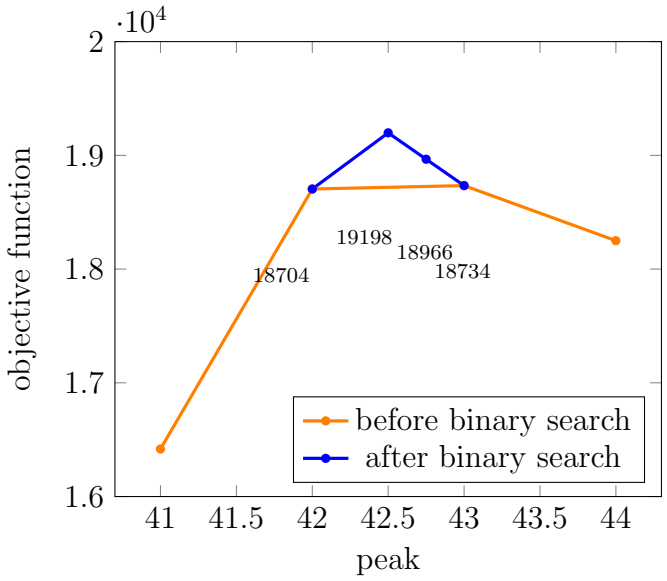


Figure 3: binary search

vices (respectively non-preemptive devices), the mixed integer formulations of PrPLB (respectively $\overline{\text{Pr}}$PLB) is denoted PrMIP (respectively $\overline{\text{Pr}}$MIP).

The numerical results are summarized in Tables 3 to 5, where each row displays the average results taken over 10 instances. The last line of each table represents the average value on the instances of the associated column.

The comparison of the peak load and the net revenue for PrMIP, PH and PSH for PrPLB instances is provided in Table 3. The heuristics' solutions are 0.31% and 0.05% away from the optimal net revenue for PH and PSH, respectively. Besides, the peak load value is 0.22% and 0.07% away from optimal peak for PH and PSH, respectively. These tests show that how fast peak loads and net revenues decrease as κ increases. Moreover, both heuristics succeed in finding solutions close to the global optimum.

As shown in Table 3, even if computation times of all three methods increase as κ increases, the heuristics provide good solutions much quicker than CPLEX for the PrMIP.

In order to assess the performance of the heuristics, the optimality gap of PH, PSH and PrMIP_{150} (PrMIP halted after 150 seconds) is compared to the optimal solution of PrMIP. The sensitivity of the net revenue gap with respect to κ is illustrated in Figure 4. Gap values increase as κ increases

	peak load				net revenue			
κ	BC	PrMIP	PH	PSH	BC	PrMIP	PH	PSH
200	65.4	56.1	55.8	56.1	51540	52358	52329	52358
400	65.4	51.0	51.0	51.0	38460	41612	41612	41612
600	65.4	47.1	47.5	47.1	25380	31898	31860	31898
800	65.4	45.0	45.0	45.5	12300	22786	22787	22758
1000	65.4	44.6	45.1	44.3	−780	13838	13648	13822
Avg Gap(%)			0.22%	0.07%			0.31%	0.05%

	CPU (seconds)		
200	31.9	21.7	13.5
400	410.4	102.9	74.7
600	2022.7	141.9	122.1
800	4244.2	154.0	152.2
1000	8717.7	153.6	154.0

Table 3: peak loads, net revenue and CPU for PrPLB

for all three methods. Figure 4 shows that reaching good quality solutions within such a short time by only solving PrMIP is not possible. In other words, the heuristics provide good starting points for the MIP phase.

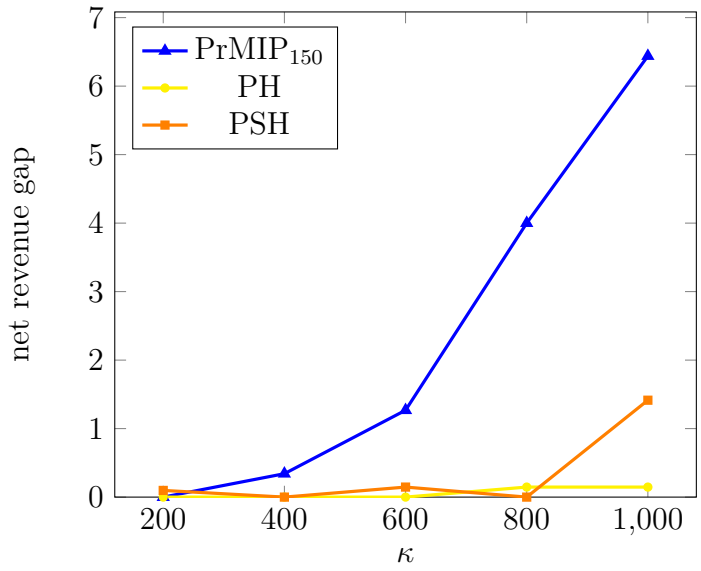


Figure 4: PrPLB: gap comparaison: PrMIP_{150}, PH and PSH versus PrMIP

A comparison of the net revenue and peak load for $\overline{\text{Pr}}$PLB is given in Table 4. Both peak load and net revenue decrease as κ increases, as expected. However, $\overline{\text{Pr}}$PLB is not as sensitive to κ as PrPLB, due to the lack of flexibility of non-preemptive appliance with respect to load distribution. Notwithstanding, the heuristics are able to find good solutions. No direct relationship can be established between the computation times and κ for $\overline{\text{Pr}}$PLB, since reshaping a schedule of non-preemptive devices is a challenging task, i.e., even a small change might result in a large perturbation, leading to a more intensive combinatorial search.

	peak load				net revenue			
κ	BC	$\overline{\text{Pr}}$MIP	PH	PSH	BC	$\overline{\text{Pr}}$MIP	PH	PSH
200	263.2	151.6	158.8	158.0	217180	231872	231469	231459
400	263.2	146.4	148.1	148.7	164540	202232	201831	201735
600	263.2	145.1	146.6	146.6	111900	173140	172443	171638
800	263.2	143.6	144.7	145.5	59260	144212	143807	143722
1000	263.2	143.5	144.2	144.2	6620	115512	115106	114526
Avg gap (%)			1.67%	1.75%			0.27%	0.45%
		CPU (seconds)						
200		9145.8	153.9	164.9				
400		7745.8	153.7	165.2				
600		4589.5	153.8	165.4				
800		7118.8	153.7	166.5				
1000		6182.9	153.7	164.6				

Table 4: comparison of peak load, net revenue and CPU for $\overline{\text{Pr}}$PLB

The net revenue gap of the heuristics is given in Figure 5. PH provides the best solutions for $\overline{\text{Pr}}$PLB, whereas both heuristics perform much better than $\overline{\text{Pr}}$MIP. As a general rule, PH outperforms PSH . This was expected since non-preemptive jobs are not as flexible with respect to peak limit as preemptive ones. Figure 5 shows that the heuristics succeed in finding solutions much closer to optimality than $\overline{\text{Pr}}$MIP within the same time limit.

The average peak load and net revenue values for MixPLB are given in Table 5. On average, the heuristics' net revenue values are within 0.13% and 0.20% of the optimal net revenue, and peak loads are within 0.44% and 0.69% of the optimal peak for PH and PSH, respectively. It is worth noting that, although PH and PSH perform better for non-preemptive and preemptive devices, respectively, their performance are very similar in the mixed case. Note also that, while CPU increases with respect to κ for all three methods,

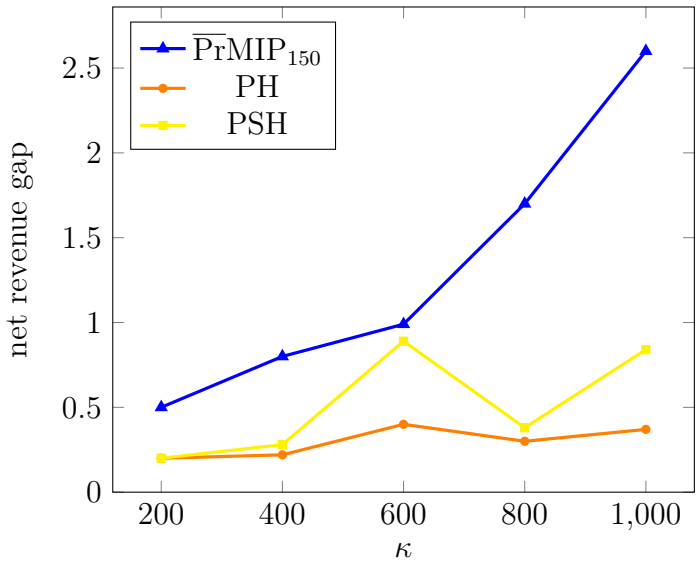


Figure 5: ($\overline{\text{PrPLB}}$): gap comparison: $\overline{\text{PrMIP}}_{150}$, PH and PSH versus $\overline{\text{PrMIP}}$

the increase is significantly larger for MixMIP. As a result, PH and PSH scale better than MIP.

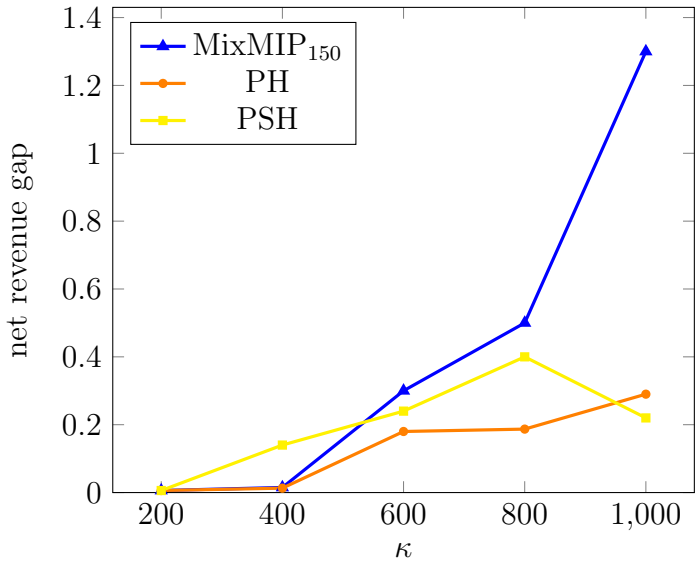


Figure 6: MixPLB: gaps: MixMIP_{150}, PH and PSH versus MixMIP

		peak load		
κ	BC	MixMIP	PH	PSH
200	127	83.10	83.30	83.3
400	127	79.40	79.40	79.9
600	127	76.35	77.15	77.35
800	127	75.75	76.70	76.3
1000	127	74.95	75.35	75.35
avg gap(%)			0.44%	0.69%
		net revenue		
200	99230	106161.34	106148.71	106148.71
400	73830	89985.32	89967.83	89894.34
600	48430	74511.72	74403.48	74338.92
800	23030	59262.69	58767.87	59019.6
1000	−2370	44193.59	44050.94	44080.8
avg gap(%)			0.13%	0.20%
		CPU (seconds)		
200		105.6	69.7	82
400		884.4	120.5	132.9
600		2844.9	142.2	146.7
800		3487.4	143.4	147.1
1000		8348.3	150.4	148.2

Table 5: MixPLB: BC versus MixMIP

The net revenue gap values of heuristics, as well as those of MixMIP after 150 seconds (MixMIP_{150}) with respect to the optimal solution of MixMIP, are presented in Figure 6. For all κ values, PH outperforms PSH, although the performance of the latter improves as κ increases.

5.3. Results on large instances

In order to analyze the scalability of the algorithms, 10 larger instances have been generated. For PrPLB and MixPLB, these involve 20 customers, each owning 5 devices. For $\overline{\text{Pr}}$PLB, 10 random instances involving 40 customers, each owning 5 devices, have been generated. Since it is not possible to solve these instances to optimality within a reasonable time, the same

time limit as the MIP part of the heuristics (150 seconds) has been applied to MixMIP. As a general rule, MixMIP cannot even find a feasible solution, whereas both heuristics provide solutions for PrPLB and MixPLB. Indeed the solution obtained by solving the follower program associated with the tariff computed at the end of the MIP is of poor quality. This led us to compare the heuristics to the base case, noting that both PH and PSH provide a good starting point for the MIP, and make it possible to find a feasible solution within 150 seconds. As a general rule, and for a given time limit, MixMIP is significantly outperformed by $\overline{\text{Pr}}$PLB. This can be inferred from the net revenues and peak loads displayed in Figures 7 and 8, with a revenue increase with respect to BC as large as 60%. Under a 150 seconds limit, we observe that PSH provides considerably higher net revenues and lower peak loads than PH. Nevertheless, both methods succeed in providing a feasible solution that constitutes a good starting point for the MIP.

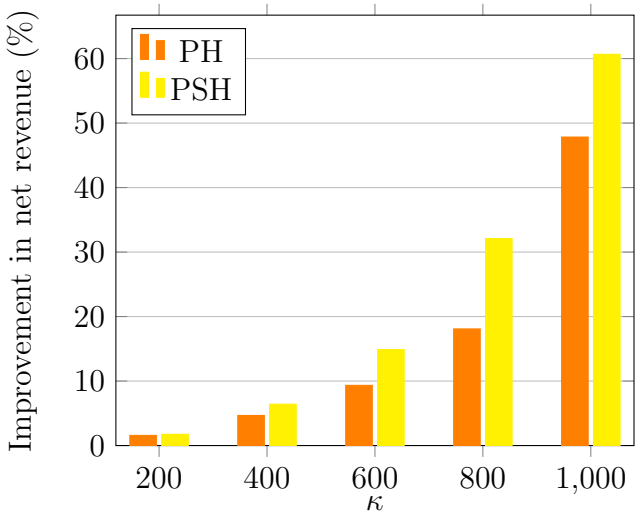


Figure 7: PrPLB: revenue comparison: PH and PSH versus BC(%) for 100 devices

The average net revenue and peak load comparisons of Price and Peak Search Heuristics to the BC for MixPLB model can be found in Figures 9 and 10, respectively. Both heuristics manage to reach good solutions within 150 seconds, yielding increases in revenue as large as 150% with respect to BC. Besides, their performances are very close to each other on MixPLB: PSH performs slightly better for higher κ values, whereas PH outperforms PSH for lower ones. Again, both methods are very efficient on instances for

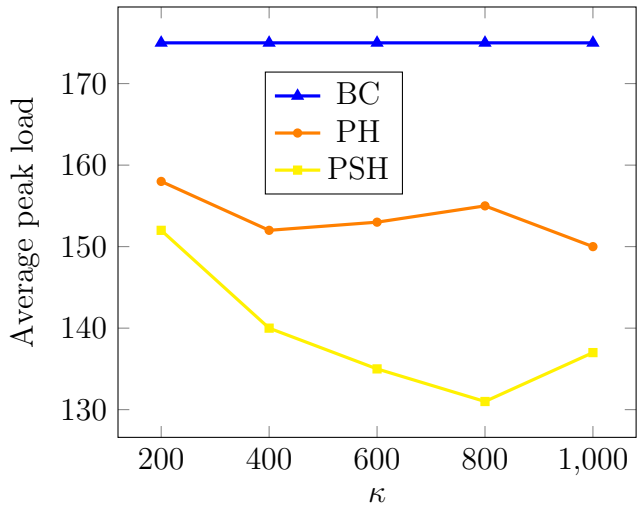


Figure 8: PrPLB: peak load comparison: PH and PSH versus BC for 100 devices

which determining mere feasible solutions for MixMIP is hard.

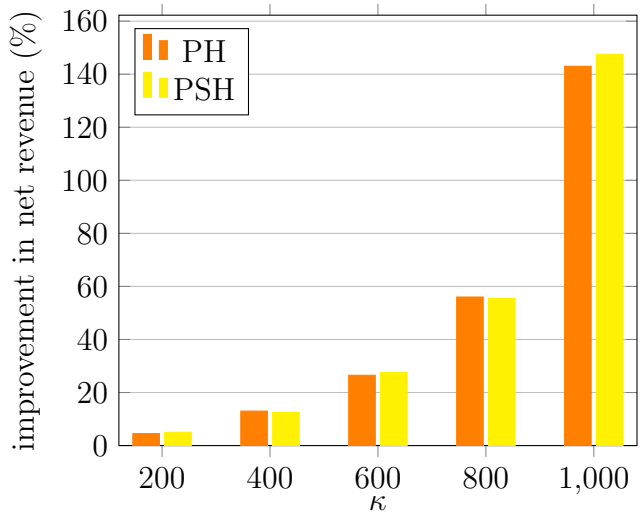


Figure 9: MixPLB: revenue comparison: PH and PSH versus BC(%) for 100 devices

For $\overline{\text{Pr}}$PLB, $\overline{\text{Pr}}$MIP can determine a feasible solution within 150 seconds. The comparison of heuristics to $\overline{\text{Pr}}$PLB with respect to net revenue (%) and peak load is given in Figures 11 and 12, respectively. It can be observed that heuristic results are significantly better than those of $\overline{\text{Pr}}$PLB, for both

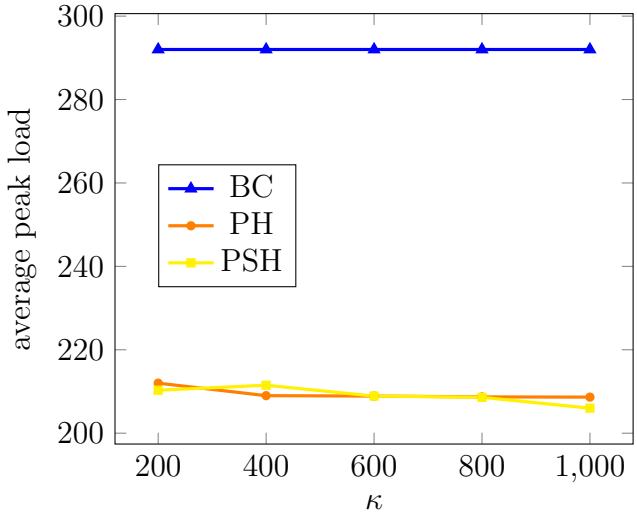


Figure 10: MixPLB: peak load comparison: PH and PSH versus BC for 100 devices

revenue and peak load criteria. Again, PSH performs better for high κ values, whereas the opposite holds for small values.

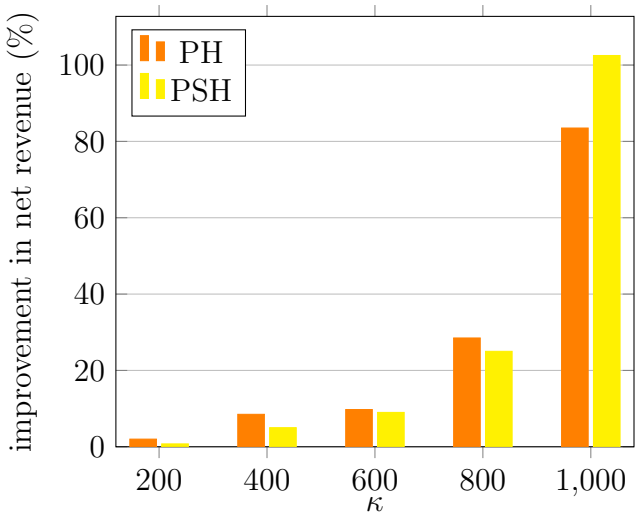


Figure 11: $\overline{\text{Pr}}$PLB: revenue comparison: PH and PSH versus $\overline{\text{Pr}}\text{MIP}_{150}$, 200 devices

The experimental setup allows to highlight the cost impact associated with a resilient grid, characterized by the peak constraint. This impact can

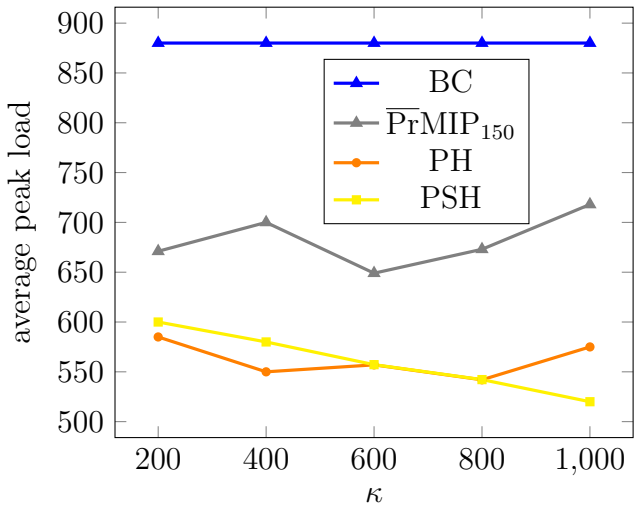


Figure 12: $\overline{\text{Pr}}$PLB: peak load comparison: PH and PSH versus BC and $\overline{\text{Pr}}\text{MIP}_{150}$, 200 devices

be mitigated by enticing users, through price signals, to postpone their consumption to off-peak periods. Clearly, such strategies can result in the desired results only if the customer reluctance is anticipated and addressed. Our results show that, for a wide range of upper bounds for the peak consumption, the heuristic algorithms achieve significant improvements with respect to the base case, both in terms of net revenue for the energy provider and cost reduction for the customers, without significant impact on the discomfort of the latter.

6. Conclusion

Due to scarce resources or environmental concerns, it is essential to apply efficient demand side management techniques in order to achieve an 'optimal' trade-off between supply and demand of the energy.

In this paper, we focused on one of these techniques, namely the day-ahead pricing and load balancing approach, from the viewpoint of a utility that anticipates the reaction of users whose needs are managed by a smart grid. This hierarchical situation is naturally modelled as a bilevel program that involves both continuous and discrete variables, for which we provided an exact linear mixed integer formulation. The resulting framework provides a valuable decision-making tool for the investigation of innovative demand management strategies.

Since, as expected, this approach does not scale well, we also proposed for its solution two heuristics that were probed on medium-size randomly generated instances. Avenues for future research will involve a detailed integration of renewable energy sources, which will yield stochastic and dynamic extensions of the current model.

References

[1] Afşar, S., Brotcorne, L., Marcotte, P., Savard, G., 2016. Achieving an optimal trade-off between revenue and energy peak within a smart grid environment. Renewable Energy 91, 293–301. doi:http://dx.doi.org/10.1016/j.renene.2016.01.055.

[2] Ahuja, R.K., Orlin, J.B., 2001. Inverse optimization. Operations Research 49, 771–783.

[3] Alves, M., Henggeler Antunes, C., 2018. A semivectorial bilevel programming approach to optimize electricity dynamic time-of-use retail pricing. Computers and Operations Research 92, 130–144.

[4] Carrasqueira, P., Alves, M., Henngeler Antunes, C., 2017. Bi-level particle swarm optimization and evolutionary algorithm approaches for residential demand response with different user profiles. Informations Sciences 418, 405–420.

[5] Didi-Biha, M., Marcotte, P., Savard, G., 2006. Path-based formulations of a bilevel toll setting, in: Dempe, S., Kalashnikov, V. (Eds.),

Optimization with Multivalued Mappings. Springer Science+ Business Media, pp. 209–225.

[6] Labbé, M., Marcotte, P., Savard, G., 1998. A bilevel model of taxation and Its application to optimal highway pricing. Management Science 44, 1608–1622.

[7] Meng, F.L., Zeng, X.J., 2014. An optimal real-time pricing demand side management: A stackelberg game and genetic algorithm approach, in: International Joint Conference on Neural Networks, Beijing, China, pp. 1703 – 1701.

[8] Mohsenian-Rad, A.H., Wong, V., Jatskevich, J., Schober, R., Leon-Garcia, A., 2010. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. IEEE Transactions on Smart Grid 1, 320–331.

[9] Muratori, M., Rizzoni, G., 2016. Residential demand response: Dynamic energy management and time-varying electricity pricing. IEEE Transactions on Power Systems 31, 1108–117.

[10] Vardakas, J., Zorba, N., V.C., 2015. A survey on demand response programs in smart grids: Pricing methods and optimization algorithms. IEEE Communications Surveys &Tutorials 17, 152–178.

[11] Zehir, M.A., Bagriyanik, M., 2012. Evaluation of management strategies for thermostatic loads in smart grid, in: 3rd IEEE PES Innovative Smart Grid Technologies Europe, Berlin,Germany, pp. 1–5.