Efficient Cost-Minimization Schemes for Electrical Energy Demand Satisfaction by Prosumers in Microgrids with Battery Storage Capabilities

Laura Codazzi[1] , **Gergely Csáji**[2] and **Matthias Mnich**[1]

[1]Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany
[2]HUN-REN Centre for Economic and Regional Studies, Budapest, Hungary
{laura.codazzi, matthias.mnich}@tuhh.de, csaji.gergely@krtk.hun-ren.hu

Abstract

We introduce and study various models of how prosumers in a microgrid can satisfy their demands of electrical energy while minimizing their costs over fixed time horizon. Prosumers have individual demands, which can vary day-by-day, and which they can satisfy by either consuming self-generating electrical energy locally (e.g., from operating PV panels) or from acquiring energy from other prosumers in the same microgrid.

Our models take into account two key aspects motivated by real-life scenarios: first, we consider a daily volatility of prices for buying and selling energy, and second, the possibility to store the self-generated energy in a battery of finite capacity to be either self-consumed or sold to other prosumers in the future. We provide a thorough complexity analysis, as well as efficient algorithms, so that prosumers can minimize their overall cost over the entire time horizon. As a byproduct, we also solve a new temporal version of the classical KNAPSACK problem, which may be of independent interest. We complement our theoretical findings by extensive experimental evaluations on realistic data sets.

1 Introduction

In the current transition of many national electrical energy markets towards more sustainable sources of energy, household-operated photovoltaic (PV) panels can offer an effective means of individual contribution towards this transition [Hockenos, 2019]. PV panels of several households within close geographical proximity can be connected through a microgrid; in such a grid, the households—or *prosumers*—can trade electrical energy with each other. Thus, any such prosumer of a fixed microgrid can satisfy their daily demand of electrical energy through a combination of self-generated electrical energy from their own PV panel, or from buying electrical energy from other prosumers connected to them in the microgrid. Several such models have been studied in the literature ([Hönen *et al.*, 2023]).

We introduce and analyze a model where prosumers have an additional source of energy to satisfy their energy demands: namely, the energy stored in their own private battery of capacity C. On each day t, each prosumer can store their self-generated energy up to the capacity C in their battery, and either consume it in the future, or sell some or all of the stored energy to other microgrid prosumers at price $s_{t'}$ per unit at a later day $t' > t$. Therefore, any prosumer can satisfy their demand d_t on day t by consuming energy from their own battery (for free), or from buying energy from other prosumers in the microgrid at a buying price c_t. The objective of any prosumer is to minimize their overall cost for acquiring energy to satisfy their daily demands, over a fixed time horizon. We assume that on each day the full demand of each prosumer can always be satisfied; this assumption can be justified by letting one of the prosumers to be a (large) company which operates the main grid to which each prosumer is individually connected, and whose own demand is zero on all days. We call this problem the *Prosumer Energy Acquisation Cost Minimization (PEAC-min)* problem, and this is the central problem of this paper.

The main innovation of our model is the possibility to trade energy which is stored in the battery with other microgrid prosumers; earlier studies focus on, e.g., operational strategies for increased battery lifetime with fixed feed-in tariffs under variable energy demands and variable amounts of PV-generated energy [Angenendt *et al.*, 2018]. The key motivation for storing energy in the battery and either consuming it or selling it later is that we assume not only variability of daily demand and daily amounts of PV-generated energy, but also daily fluctuation of both buying price c_t and selling price s_t. The battery offers prosumers the possibility to act strategically, compensating their own expenses for buying energy on some days by selling energy on other days. PEAC-min is a very natural scheduling problem: any prosumer, for each day, schedules (i) the amount of energy to consume from their self-generated energy through the PV panel, (ii) the amount to store in their battery for self-using or selling in the future, (iii) the amount to consume from their battery, and (iv) the amount to buy from other prosumers.

A strategic usage of the battery can indeed lead to a significant cut on the energy bill as well as more conscious usage of energy. Typically, each one of these households is equipped with a Home Energy Management System (HEMS), which is responsible for coordinating the different demands of energy (e.g., domestic appliances, EVs) with the PVs and the battery

[Beaudin and Zareipour, 2015]. The ownership of a HEMS, or equivalently a *smart meter* ([Tushar *et al.*, 2021]), is generally assumed in most of the local energy markets (LEM) among prosumers, as it is a powerful tool to submit bids. One of the key aspects of the problem is that decisions taken at a given period impact the future by changing the state of the system itself, namely the amount of energy stored.

The PEAC-min problem tackled in this work lies in between the LEM and the HEMS. We zoom out from the HEMS and consider it as a sort of black box, which gives a balance of energy for each period (e.g., one day) and optimize the decision to be taken at the given period for a single prosumer. Though we analyze the strategic actions of a single prosumer, note that their optimal strategic actions is influenced by, but also influences itself, the strategic actions of other prosumers in the same microgrid. Not only do they influence the available amounts of energy for trading, but also influence the buying and selling prices depening on their demand and decisions of how much energy to store in the battery, to buy and to sell. The relevance of studying and understanding these local energy markets, and designing efficient actions of a single prosumer, is underlined by the realistic assumption that a prosumer may not be willing or be able to coordinate their actions with other prosumers, and thus needs to know their own (selfish) optimal decision plan provided by our algorithms. With some natural assumptions on the prices for buying and selling, arranging these action plans of several prosumers which are connected by a microgrid can lead to efficient local energy markets [Capper *et al.*, 2022].

One key aspect of these types of scheduling problems is the intrinsic uncertainty of the parameters. A popular method to deal with this stochasticity is to use a simulation approach. Closely related to the models in our paper is the work of [Hafiz *et al.*, 2019], which extends to a energy community the *stochastic dual dynamic programming* approach, already extremely popular in hydrothermal scheduling problems [Morton, 1996]. For sake of the aforementioned generality, we decided to introduce variability also on the prices for buying and selling energy. In this new setting, the use of a Monte Carlo scheme becomes very difficult as the scenario tree grows exponentially in several parameters. Thus, the decision relies on predictions about the expected amount of PV-generated energy in future time periods. Motivated by the rapid advance of HEMS and predictor technologies, we study the offline version relying on a good predictor. Even though in practice, the decisions regarding how much to buy and sell in a given period are made in an online fashion, a good offline strategy can lead to good decisions, if in each period, we base the online decision on solving an offline version for the next period according to some predicted prices, especially if these predictions are highly accurate.

In the full version, we show that if predictions have small multiplicative errors, then a solution for the offline problem remains near-optimal even in the worst case. To conclude, we mention that our setting relates to KNAPSACK, which is one of the 21 NP-complete problems studied by [Karp, 1972]. Namely, we introduce a new *temporal* KNAPSACK variant which generalizes earlier models studied by [Bartlett *et al.*, 2005; Clautiaux *et al.*, 2021; Thielen *et al.*, 2016].

To support our theoretical findings, in Section 7 we provide an experimental analysis of the performance of our algorithms, whose full code we provide in a public GitHub repository [Codazzi *et al.*, 2024].

2 Preliminaries

We consider the following model. We have a finite time horizon which is split into n periods (e.g., days). During each period $t \in [n] := \{1, \dots, n\}$, a prosumer has a given demand of $d_t \in \mathbb{N}$ units of electrical energy. Similarly, they have an amount of $e_t \in \mathbb{N}$ units of electrical energy available as being generated from the PV panels in period t. In period t, they can buy energy at a current price $c_t \in \mathbb{N}_{\geq 0}$ from other prosumers in the microgrid. Depending on the characteristics of the grid and the technology of the HEMS, the prosumer could also be allowed to sell energy back to the grid at a price $s_t \in \mathbb{N}$ per unit of energy. A natural assumption about electricity prices, which can be observed in real markets, is that typically buying is more expensive than the revenue obtained from selling to the grid[1]; thus, throughout we assume that $c_t \geq s_t$ for all $t \in [n]$. This does not mean that selling is in general not convenient; instead, degenerated behavior of a prosumer buying infinite amounts of energy and selling them in the same period is avoided.

Finally, a battery is available to each prosumer with a capacity $C \in \mathbb{N}$. At the beginning of the time horizon, the battery is charged with $soc_0 \in \mathbb{N}$ units of energy.

We observe that each period t in the time horizon falls into one of two classes. On one hand, we have periods where the amount of PV-generated energy supply is at most the demand from that period ($e_t \leq d_t$); on the other hand, we have *excess periods* where this supply exceeds the demand ($d_t < e_t$). We define two quantities $d'_t := (d_t - e_t)^+$, and $ex_t := (e_t - d_t)^+$. Hence, for each period t, we have that $d'_t \geq 0$ and $ex_t \geq 0$ (at most one of which is positive), and $d_t - e_t = d'_t - ex_t$.

3 Integer Linear Programming Formulations and Solutions

We now model the different problem variants of PEAC as integer linear programs (ILP), for which we use:

n	number of periods
c_t	price per unit for buying energy in period t
s_t	price per unit for selling energy in period t
d_t	demand in period t
e_t	amount of PV-generated energy in period t
C	capacity of the battery
soc_0	initial state of charge of the battery
x_t	units of energy bought from other pros. in per. t
y_t	units of energy sold to other prosumers in per. t
z_t	units of energy exchanged with battery in per. t
SOC_t	state of charge of battery in period t
q_t	binary variable: $q_t = 1$ if we buy in period t

[1]Germany's Federal Network Agency (Bundesnetzagentur) reports for 1 April 2023 an average household price of 45.19 ct/kWh [Bun, 2024a] for electrical energy, whereas the feed-in tariff for PV-generated energy was at most 13.40 ct/kWh [Bun, 2024b].

For each time period $t \in \{1, \ldots, n\}$, let $x_t \geq 0$ be the amount of energy bought, let $y_t \geq 0$ be the amount of energy sold, and let z_t be the amount of energy stored or discharged from the battery (so z_t can be either positive or negative or zero). Furthermore, we keep track daily of the state of the system by means of the variables SOC_t.

In the first problem we describe, the prosumer can fill up the battery by buying energy and can buy or sell an arbitrary amount of energy on each day. Therefore, we refer it to as the UNBOUNDED version and call it PEAC-BS-U. It is modeled by the following ILP:

$$\min \sum_t (c_t x_t - s_t y_t) \tag{1}$$

$$\text{s.t.} \quad x_t - y_t - z_t = d_t - e_t \tag{2}$$

$$SOC_t = SOC_{t-1} + z_t \tag{3}$$

$$SOC_0 = soc_0 \tag{4}$$

$$SOC_t \leq C \tag{5}$$

$$x_t, y_t, SOC_t \in \mathbb{N} \tag{6}$$

$$z_t \in \mathbb{Z} \tag{7}$$

PEAC-BS-U

In: $I = (c_t, s_t, d_t, e_t, C, soc_0 \mid t \in [n])$.

Out: An optimal solution to the ILP defined by (1)–(7).

The problem variant where selling is not allowed is called PEAC-B-U we define it by adding the constraints $y_t = 0, t \in [n]$, to PEAC-BS-U.

Equation (2) ensures that the amount we buy together with the amount we get from the PVs minus the amount we sell or store must be exactly the daily demand. Equation (3) ensures that the amount stored (which can be negative in case we use energy from the battery) is added to the previous state of charge of the battery. Equation (5) ensures that the state of the battery never exceeds the capacity.

Depending on the characteristics of the grid, prosumers may face restrictions on the feasible amounts of energy to buy. For instance, it could be that—for fairness reasons—prosumers are not allowed to buy more than the amount they need on that day. As a consequence, the battery can only be charged only with self-generated energy from PVs (see [Hafiz *et al.*, 2019]). To express this restriction, we bound the amount of energy which can be bought by adding the following constraint to PEAC-BS-U:

$$x_t \leq (d_t - e_t)^+ \tag{8}$$

This allows us to formally define our next problem variant:

PEAC-BS

In: An instance $I = (c_t, s_t, d_t, e_t, C, soc_0 \mid t \in [n])$.

Out: An optimal solution to the IP defined by constraints (1)–(8).

Analogously, we define the variant PEAC-B, where selling is not allowed, by adding the constraints $y_t = 0$ to PEAC-BS.

We proceed to define a common generalization of PEAC-BS-U and PEAC-BS called PEAC-BS-GENERAL. Here, instead of (8) we add inequalities of the type

$$x_t \leq \alpha_t \tag{9}$$

$$y_t \leq \beta_t, \tag{10}$$

where $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ are given in the input.

PEAC-BS-GENERAL

In: An instance $I = (c_t, s_t, d_t, e_t, \alpha_t, \beta_t, C, soc_0 \mid t \in [n])$.

Out: An optimal solution to the ILP defined by constraints (1)–(7) and (9)–(10).

Clearly, PEAC-BS-U corresponds to the case when we set each α_t and β_t to be sufficiently large, e.g., $\alpha_t = C + d'_t - ex_t$ and $\beta_t = C + ex_t - d'_t$, and PEAC-BS corresponds to the case when we set $\alpha_t = (d_t - e_t)^+$ and $\beta_t = C + ex_t - d'_t$.

The variant PEAC-B-GENERAL is defined similarly, by adding constraints $y_t = 0$; equivalently, we can set $\beta_t = 0$.

Hence, it is trivial that PEAC-B-GENERAL is a special case of PEAC-BS-GENERAL, and so are PEAC-B-U and PEAC-BS-U. However, we will show that PEAC-BS-GENERAL also reduces to PEAC-B-GENERAL in $\mathcal{O}(|I|)$ time for a given instance I, and hence it is enough to give an algorithm for PEAC-B-GENERAL to solve all the above problem variants.

Finally, we consider a scenario in which if the prosumer buys energy from the grid, they are then forced to buy the whole amount needed. This implies that at every time period, the prosumer can either buy or satisfy the whole demand by means of the energy stored in the battery. To model these last two cases, we introduce the binary variable q_t which takes value 1 in case the prosumer decides to buy. To emphasize that the choices of the prosumers are mutually *exclusive*, we call this variant PEAC-BS-EX. PEAC-BS-EX can be obtained from PEAC-BS-U by adding the constraints

$$x_t = (d_t - e_t)^+ q_t \tag{11}$$

$$q_t \in \{0, 1\} \tag{12}$$

PEAC-BS-EX

In: An instance $I = (c_t, s_t, d_t, e_t, C, soc_0 \mid t \in [n])$.

Out: An optimal solution to the IP defined by constraints (1)–(7) and (11)–(12).

The variant PEAC-B-EX, where selling is not allowed, is defined by adding the constraints $y_t = 0$ to PEAC-BS-EX.

A common generalization for these two problem variants can be defined similarly as before, by adding the following inequalities to PEAC-BS-U:

$$x_t = \alpha_t q_t \tag{13}$$

$$y_t \leq \beta_t \tag{14}$$

$$q_t \in \{0, 1\} \tag{15}$$

PEAC-BS-EX-GEN

In: An instance $I = (c_t, s_t, d_t, e_t, \alpha_t, \beta_t, C, soc_0 \mid t \in [n])$.

Out: An optimal solution to the ILP defined by constraints (1)–(7) and (13)–(15).

	PEAC-BS-GENERAL		PEAC-BS-EX-GEN
complexity	polynomial-time solvable		NP-hard
algorithm	Buying Backwards		Dynamic programming
selling forbidden	PEAC-B-U $\alpha_t = C + d'_t - ex_t, \beta_t = 0$	PEAC-B $\alpha_t = d'_t, \beta_t = 0$	PEAC-B-EX $\alpha_t = d'_t, \beta_t = 0$
selling allowed	PEAC-BS-U $\alpha_t = C + d'_t - ex_t, \beta_t = C + ex_t - d'_t$	PEAC-BS $\alpha_t = d'_t, \beta_t = C + ex_t - d'_t$	PEAC-BS-EX $\alpha_t = d'_t, \beta_t = C + ex_t - d'_t$

Table 1: An overview of our results.

It is easy to see that the values to variables z_t and SOC_t are uniquely determined by the values of variables x_t and y_t for $t \in [n]$ in all cases. Hence, we usually refer only to $(x_1, y_1, \ldots, x_n, y_n)$ as the solution.

4 Structural Insights of the ILPs

In this section we analyze the structure of the ILPs. Recall that a matrix $A \in R^{m \times n}$ is *totally unimodular (TU)* if each subdeterminant of A is from $\{0, \pm 1\}$. It is well-known (cf. [Schrijver, 1998, Theorem 19.1]) that any Linear Program (LP) whose constraint matrix A is TU admits an integer optimum solution.

Theorem 1. *The constraint matrix of the ILP* (1)–(7) *with* (9)–(10) *is totally unimodular.*

Theorem 1 has the following corollary.

Corollary 2. PEAC-BS-GENERAL *can be solved in polynomial time.*

Even if it is possible to solve the problem by means of an LP solver in polynomial time, we provide a combinatorial algorithm that can solve these problems $\mathcal{O}(n^2)$ steps. Not only is this a fast theoretical run time, we will later show that it is also fast in experiments. It also helps to better understand the structure of the problem.

5 A Polynomial-Time Algorithm for PEAC-B-GENERAL

In this section we provide a polynomial-time algorithm to solve problem PEAC-B-GENERAL and show that this also allows to solve PEAC-B-U, PEAC-BS-U, PEAC-B and PEAC-BS efficiently. For this, we first show that an instance of PEAC-BS-GENERAL can be reduced to an instance of PEAC-B-GENERAL in linear time. When analyzing runtime, each addition and multiplication is assumed to have unit cost.

Theorem 3. *Any instance I of* PEAC-BS-GENERAL *can be reduced to* PEAC-B-GENERAL *in $\mathcal{O}(|I|)$ time.*

We proceed to informally describe our algorithm for PEAC-B-GENERAL. Due to space constraints, we defer its formal description and proof of correctness to the full version.

The algorithm iterates through periods 1 to n and at each point has a feasible solution for the first couple of periods, which it maintains by updating the amounts it bought on the previous periods. It keeps track of the latest period f, where the battery is full in the current solution, initialized to 0. It also keeps track, for each period $t' < t$, in a variable $R_{t'}$ the largest possible amount we can add to $x_{t'}$ in our current solution, without creating a period $t'' \geq t'$ where the state of charge would exceed C. We initialize this to something sufficiently large (e.g., $C + d'_t$) for each period to handle the possibility of buying even $C + d'_t$ on period t, if it is feasible (buying more can never be feasible). After period t is over, R_t gets updated to $C - SOC_t$.

In a given iteration corresponding to a period t, the algorithm does the following. If $ex_t > 0$ or $SOC_{t-1} \geq d'_t$, then we do not need to buy anything to maintain a feasible solution for the first t periods, we just update the variables R_i, f and SOC_t according to this. Otherwise, we have to satisfy the additional demand $d'_t - SOC_{t-1}$. For this, we always compute the (latest) period i^* with minimum buying price among periods where we can still buy on (i.e. the ones with $f < i$ and $x_i < \alpha_i$). Then, we look at the maximum amount we can add to x_{i^*} in a feasible solution, and if it is at least the remaining demand, then we set it to that, update the variables and proceed to the next period. Otherwise, we increase x_{i^*} by as much as we can by maintaining feasibility, update the R_i, f and SOC_i variables, and then finally update i^* and iterate with that until the demand is fully satisfied. The algorithm's pseudocode is given as Algorithm 1.

Theorem 4. *Algorithm 1 outputs an optimal solution for* PEAC-B-GENERAL *in $\mathcal{O}(n^2)$ time.*

Proof. First of all, if the algorithm outputs a solution, then it is feasible. This is because we do not proceed to the next period until the demand is fully satisfied, and if we increase x_i on some day, then because we store R_i, and only increase with at most R_i, we never exceed the capacity of the battery. Finally, we also ensure that $x_i \leq \alpha_i$ when computing $incr$.

If the algorithm returns that there is no feasible solution, then there cannot be one, as the algorithm only buys energy from other prosumers if no more energy is left in the battery to satisfy a demand. Hence, if the battery becomes overfull, then there is an interval $[t_1, t_2]$ such that $\sum_{t=t_1}^{t_2} (e_t - d_t) > C$, so there cannot be a feasible solution.

We prove optimality of the computed solution by induction on $\sum_t \alpha_t$. In case that $\sum_t \alpha_t = 0$, then the only possible feasible solution is $(0, \ldots, 0)$ and the algorithm's output is clearly optimal, if it is feasible.

Suppose that $\sum_t \alpha_t > 0$ and the instance has a feasible solution. If the algorithm never buys energy, then the output is

Algorithm 1: Buying backwards

Input: An instance $I = (c_t, d_t, e_t, \alpha_t, C, soc_0 \mid t \in [n])$ of PEAC-B-GENERAL
$\mathsf{cost} := 0, SOC_0 = soc_0, f = 0$ ▷ f is the latest full period
$x_t := 0, d'_t = (d_t - e_t)^+, ex_t = (e_t - d_t)^+$ for $t = 0, \ldots, n$
$R_t = C + d'_t$ for $t = 1, \ldots, n$ ▷ smallest remaining capacity on periods $t, \ldots, n$ - monotone decreasing, after R_t becomes 0, we never update it or any previous one
for $t = 1, \ldots, n$ **do**
  $R_t := C + d'_t - SOC_{t-1}$
  $i^* :=$ (latest) $\arg\min\{c_i \mid f < i \leq t, x_i < \alpha_i\}$ ▷ i^* is latest cheapest period when we can still buy
  if $ex_t > 0$ **then**
    $R_t := R_{t-1} - ex_t, SOC_t := SOC_{t-1} + ex_t$
    for $j = f + 1, \ldots, t - 1$ **do**
      $R_j := \min\{R_j, R_t\}$ ▷ update remaining capacities considering ex_t
    end
  end
  else if $SOC_{t-1} \geq d'_t$ **then**
    $R_t := R_{t-1} + d'_t, SOC_t := SOC_{t-1} - d'_t$ ▷ We can satisfy d'_t without buying
  end
  else
    $d'_t := d'_t - SOC_{t-1}$ ▷ The amount we need
    while $d'_t > 0$ **do**
      $incr := \min\{R_{i^*}, \alpha_{i^*} - x_{i^*}, d'_t\}$ ▷ The amount of d'_t we can satisfy on i^*
      $x_{i^*} = x_{i^*} + incr$
      $\mathsf{cost} := \mathsf{cost} + c_{i^*} incr$
      for $j = t - 1, \ldots, i^*$ **do**
        $SOC_j := SOC_j + incr$
        $R_j := R_j - incr$
        if $R_j = 0$ **then**
          $f := j$
          break ▷ update f
        end
      end
      for $j = f + 1, \ldots, i^* - 1$ **do**
        $R_j := \min\{R_j, R_{i^*}\}$
      end
      $d'_t := d'_t - incr$ ▷ amount still needed
      $i^* :=$ (latest) $\arg\min\{c_i \mid f < i \leq t, x_i < \alpha_i\}$
    end
    $SOC_t := 0, R_t := C$
  end
  if $SOC_t > C$ **then**
    No feasible solution!
  end
  else if $SOC_t = C$ **then**
    $f := t$
  end
end
return $\mathsf{cost}; x_1, \ldots, x_n$

clearly optimal. Otherwise, let j be the coordinate of the output that is first increased during the algorithm and let $t \geq j$ be the last iteration where it is increased. Then, until that point, the algorithm only increased x_j, and none of the other x_i's.

Denote the output by $Sol = (x_1, \ldots, x_n)$. By the previous argument we have that $x_j \geq 1$, because the algorithm never decreases any coordinate.

Let $O^* = (x^*_1, \ldots, x^*_n)$ be an optimal solution for which $|x^*_j - x_j|$ is minimum. We claim that $x^*_j = x_j$.

Suppose first for the contrary that $x^*_j < x_j$. The fact that the algorithm needed to increase buying on j to x_j in iteration t implies that in order to satisfy all the demands of the first t periods, O^* must also buy on some periods $(j \neq) j' \leq t$, and choose j' to be the first such period. Further, by choice of the algorithm, we had that $j = i^*$, so $\alpha_j > 0, \alpha_{j'} > 0$, and $c_j \leq c_{j'}$, because c_j was minimum among those with $\alpha_i > 0$ and $f < i$ ($f < j'$ too, because the only way that $f = \ell > 0$ at the start of iteration t in the algorithm is if $\sum_{i=1}^{\ell}(d_i - e_i) + soc_0 = C$, so no feasible solution can buy in period $i \leq \ell$).

Create a solution $S = (x'_1, \ldots, x'_n)$ from O^* by decreasing $x^*_{j'}$ by 1 and increasing x^*_j by 1. Then we still have $x_{j'} \in [0, \alpha_j]$ and $x^*_j \in [0, \alpha_j]$, as Sol and O^* are feasible.

Suppose that $j < j' \leq t$. Then, $SOC^S_i \geq SOC^{O^*}_i \geq 0$ for any period i. Further, as $j' \leq t$ was the first period other than j that O^* bought energy on, and as Sol is feasible, buying 1 more unit of energy on period j cannot make the battery go above capacity until day j', so $SOC^S_i \leq SOC^{Sol}_i \leq C$ for $i < j'$. From period j', $SOC^S_i = SOC^{O^*}_i \leq C$ for $i \geq j'$. Hence, we conclude that S is feasible, but either S has smaller cost than O^*, or S is optimal but $|x^S_j - x_j| < |x^*_j - x_j|$, a contradiction.

Suppose now that $j' < j$. Then $SOC^S_i \leq SOC^{O^*}_i \leq C$ for any period i. Furthermore, $SOC^S_i \geq 0$ for $i < j$ (as Sol did not buy anything before j up until iteration j, but still all demands up to period $j - 1$ could be satisfied). Finally, $SOC^S_i = SOC^{O^*}_i \geq 0$ for $i \geq j$. Hence, S is feasible in this case too, but either S has smaller cost, or the same cost but $|x^S_j - x_j|$ is strictly smaller, contradiction.

For the other direction, suppose that $x_j < x^*_j (\leq \alpha_j)$. Then, there must be some some other period j', where the algorithm buys more than $x^*_{j'}$ (otherwise O^* is not optimal or not the closest optimal solution by $c_j \geq 0$). Let $j' \neq j$ be the coordinate of the output that is first increased strictly above $x^*_{j'}$ in the algorithm and suppose it happened in iteration $t' \geq t$. This implies that i^* has changed by that point. However, it was still true that $f < j$ and $x_j < \alpha_j$ (before this increase in $x_{j'}$, we had that $x_i \leq x^*_i$ for all i and $x_j < x^*_j$, so if $f \geq j$ or $x_j \geq \alpha_j$, then this would mean that O^* is infeasible). Hence, $c_{j'} \leq c_j$ and $j' > j$ (if $c_{j'} = c_j$, then by the definition of i^*, otherwise if $c_{j'} < c_j$ and $j' < j$, then j' would have been chosen as i^* before j, contradiction).

Create a solution S from O^* by increasing $x^*_{j'}$ by 1 and decreasing x^*_j by 1. Then $0 \leq x^*_{j'} + 1 \leq x_{j'} \leq \alpha_{j'}$ and $0 \leq x_j \leq x^*_j - 1 \leq C$. As $j < j'$, we also have $SOC^S_i \leq SOC^{O^*}_i \leq C$ for all $i \in [n]$. Also, $SOC^S_i \geq 0$ for $j \leq i < j'$, as by the end of iteration $j' - 1$, the algorithm bought

at most as much as S on each period and still satisfied all demands up until period $j'-1$. For $i < j$ and $i \geq j'$, we have that $SOC_i^S = SOC_i^{O^*} \geq 0$ too. So S is feasible, and either S has smaller cost than O^*, or the same cost but $|x_j^S - x_j| < |x_j^* - x_j|$, a contradiction. We conclude that $x_j = x_j^*$.

Now create a new instance I' from I by increasing e_j by 1 and decreasing α_j to $x_j - 1$. Then $(x_1^*, \ldots, x_j^* - 1, \ldots, x_n^*)$ is feasible for I'. Also, one sees that the algorithm outputs $(x_1, \ldots, x_j - 1, \ldots, x_n)$ for this new instance (until iteration t, if it buys, then it still only buys on period j and in iteration t, it increases this amount only to $x_j - 1$, but since e_j is increased by 1, from this point every R_i, SOC_i and $x_i (i \neq j)$ is the same, so it runs the same way from this point as for I). By induction, this is optimal for I', so $\sum_t c_t x_t - c_j \leq \sum_t c_t x_t^* - c_j$, hence $\sum_t c_t x_t \leq \sum_t c_t x_t^*$, so the output is optimal for I, too.

To analyze the runtime, note that the outer **for** loop has n iterations. In the first two cases ($ex_t > 0$ or $SOC_{t-1} \geq d'_t$), we make $\mathcal{O}(n)$ steps per iteration. In the third case ($d'_t > SOC_{t-1} \geq 0$), in each step, either the demand of period t gets satisfied and we move on to the next period, or period i^* gets saturated ($x_{i^*} = \alpha_{i^*}$), or f gets updated to a later period. Hence, the number of all such updates over all iterations of the outer **for** loop is at most $3n$, which together with the inner **for** loops to update SOC_j, R_j and f gives $\mathcal{O}(n^2)$ steps. Thus, the runtime is bounded by $\mathcal{O}(n^2)$. □

6 A Dynamic Programming Approach to PEAC-BS-Ex-Gen

In this section we consider the problems PEAC-B-Ex and PEAC-BS-Ex. Due to space constraints, the proofs of this section are deferred to the full version. Instead, we provide the intuition behind all algorithms given here.

Our first result is an NP-hardness reduction.

Theorem 5. PEAC-B-Ex *and* PEAC-BS-Ex *are* NP*-hard, even if* $d_t \geq e_t$ *for all periods* $t \in [n]$.

By Theorem 5, PEAC-B-Ex and PEAC-BS-Ex are NP-hard even when there are no excess periods. However, it is exactly the excess periods which differentiate our problem from earlier studied temporal versions of Knapsack where the knapsack capacity changes over time. Later, we introduce a new variant of Knapsack that captures the features of our problem, and we explain the connections between them. But first we provide a dynamic programming algorithm for PEAC-BS-Ex-Gen, which also solves problems PEAC-B-Ex and PEAC-BS-Ex as a corollary.

The intuition of the algorithm is that we iteratively solve the subproblems $SP(t, C-k)$ for $k \in [0, C]$ and $t \in [0, n]$. Subproblem $SP(t, C-k)$ consists of finding the maximum of $\sum_{i=1}^{t}(y_i s_i - x_i c_i)$ restricted to the first t periods for a solution $(x_1, y_1, \ldots, x_t, y_t)$ such that $SOC_t = C - k$ holds exactly. At the end of each iteration t, we store the optimum values of subproblems $SP(t, C-k)$ for $k \in [0, C]$ in a vector R.

The algorithm's pseudocode is given in Algorithm 2. In the algorithm, the maximum over the empty set is set to $-\infty$.

We suppose that $ex_t \leq C$ for any $t \in [n]$. Otherwise, we must sell at least $ex_t - C$ units in period t, so then we can reduce our instance by setting $\beta_t := \beta_t - (ex_t - C)$ and $ex_t := C$ and solve the new instance. To align with the maximization goal of the corresponding Knapsack instance defined later, we equivalently change our objective function to $\max \sum_{t=1}^{n}(s_t y_t - c_t d_t)$.

We remark that we can compute the maxima in the algorithm in a clever iterative way (details in the full version) that leads to a significant speedup. Thereby, we obtain that:

Theorem 6. *Algorithm 2 outputs an optimal solution for* PEAC-B-General *in* $\mathcal{O}(nC \log C)$ *time. For* PEAC-B-Ex *and* PEAC-BS-Ex*, the time is* $\mathcal{O}(nC)$.

Algorithm 2: PEAC-BS-Ex-Gen

Input: An instance $I = (c_t, s_t, d_t, e_t, \alpha_t, \beta_t, C, soc_0 \mid t \in [n])$ of PEAC-BS-Ex-Gen

$ex_t := (e_t - d_t)^+$ for $t \in [n]$

$R[C-k] := \begin{cases} 0 & \text{if } C - k = soc_0 \\ -\infty & \text{otherwise} \end{cases}$ ▷ We have no choice but to leave soc_0 for the first period

Set $Q := [0, \ldots, 0]$ ▷ we store the optima for the new subproblems in Q before updating R

for $t = 1, \ldots, n$ **do**

  for $k = 0, \ldots, C$ **do**

    $a_1^k = \min\{\beta_t, k + ex_t - d'_t\}$ ▷ Max amount possible to sell in case we choose $x_t = 0$

    $a_0^k = \max\{0, k + ex_t - d'_t - C\}$ ▷ Min possible

    $A(C-k) := \max_{a_0^k \leq y \leq a_1^k}\{R[C-k+y-ex_t+d'_t] + y \cdot s_t\}$

    $b_1^k = \min\{\beta_t, k + \alpha_t + ex_t - d'_t\}$ ▷ Max amount possible to sell in case we choose $x_t = \alpha_t$

    $b_0^k = \max\{0, k + \alpha_t + ex_t - d'_t - C\}$ ▷ Min possible

    $B(C-k) := \max_{b_0^k \leq y \leq b_1^k}\{R[C-k+y-ex_t-\alpha_t+d'_t] + y \cdot s_t - \alpha_t \cdot c_t\}$

    $Q[C-k] = \max\{A(C-k), B(C-k)\}$

  end

  Set $R := Q$ and $Q := [0, \ldots, 0]$.

end

return $\max_k\{R[k]\}$.

A New Temporal Knapsack Variant. We now introduce a new variant of the Knapsack problem that closely relates to PEAC-BS-Ex-Gen. It is formally defined as follows:

Temporal Knapsack with Item Copies and Lower and Upper Bounds (TKICLUB)

In: A time horizon $\tau = [n]$, a set of m items with weights $w_i \in \mathbb{N}$, profits $p_i \in \mathbb{N}$, a number of copies N_i and an arrival time $t_i \in \tau$ for $i \in [m]$, and time-dependent monotone increasing lower and upper bounds $L_t \leq U_t$ for $t \in [n]$.

Out: An integral vector $x = (x_1, \ldots, x_m)$ such that $L_t \leq \sum_{i|t_i \leq t} w_i x_i \leq U_t$ for all $t \in \tau$, $x_i \in [0, N_i]$ for all $i \in [m]$ and $\sum_{i=1}^{n} p_i x_i$ is maximum.

Intuitively, the connection between PEAC-BS-EX-GEN and TKICLUB is that we can view PEAC-BS-EX-GEN as a problem, where we start from (a not necessarily feasible) solution, where we buy the demand each period, and then we want to maximize our savings from there. That is, we choose the periods where we use the battery instead of buying d'_t and the periods and amounts that we sell. As our battery usage is limited by the excess energy up to that point on any given day, we have a time dependent upper bound on the corresponding KNAPSACK instance, but also because of the battery constraint, we have time dependent lower bounds on at least how much item weight must be in the knapsack. Also, as the amount we sell is not a binary choice, the KNAPSACK instance has items that we can put inside the knapsack multiple times (given by $N_i = \beta_t$), which we emphasize is fundamentally different from having β_t copies of the item, since in the former case, β_t may not be polynomial in the input size. In particular, we show:

Lemma 7. *Any instance I of* PEAC-BS-EX-GEN *can be reduced to an instance I' of* TKICLUB *in $\mathcal{O}(|I|)$ time such that an optimal solution to I can be constructed from an optimal solution to I' in polynomial time.*

In the remainder of this section, we extend our dynamic programming algorithm to TKICLUB. First, we order the items $i \in [m]$ such that their arrival times are monotone increasing, that is $t_j \geq t_l$ if $j > l$. Then we have subproblems $SP(i, U_{t_i} - k)$ for each $i \in [m]$ and $0 \leq k \leq U_{t_i} - L_{t_i}$, which correspond to the restriction of the instance to the first t_i periods and the first i items, such that in the end exactly $U_{t_i} - k$ weight is in the knapsack. We store these optima in $R[k]$ for each iteration i. Beyond this, the idea of the algorithm remains as before.

Algorithm 3: TKICLUB

Input: A TKICLUB instance $I = (n, m, w_i, p_i, N_i, t_i, L_i, U_i \mid i \in [m])$

$R[k] := \begin{cases} 0 & \text{if } k = 0 \\ -\infty & \text{otherwise} \end{cases}$ for $k = 0, 1, \ldots, \max_j\{U_j - L_j \mid j \in [n]\}$. ▷ initialization for the empty subproblem

$L_0 := 0, U_0 := 0$

$Q := [0, \ldots, 0]$ ▷ we store the optima for the new subproblems in Q before updating R

for $i = 1, \ldots, m$ **do**

  for $k = 0, \ldots, U_{t_i} - L_{t_i}$ **do**

    $a_0 = \max\{0, \lceil \frac{U_{t_i} - U_{t_{i-1}} - k}{w_i} \rceil\}$

    $a_1 = \min\{N_i, \lfloor \frac{U_{t_i} - L_{t_{i-1}} - k}{w_i} \rfloor\}$ ▷ these bounds force that the below max only queries R-values from the range $[0, U_{t_{i-1}} - L_{t_{i-1}}]$

    $Q[k] = \max_{a_0 \leq x_i \leq a_1}\{R[U_{t_{i-1}} - U_{t_i} + k + w_i x_i] + p_i x_i\}$

  end

  Set $R := Q$ and $Q := [0, \ldots, 0]$.

end

return $\max_k\{R[k]\}$

Theorem 8. *Algorithm 3 solves TKICLUB optimally in time $\mathcal{O}(mW \log W)$, where $W = \max_t\{U_t - L_t\}$.*

7 Experimental Analysis

We implemented all combinatorial algorithms in Python. To measure their experimental efficiency, we also implemented the ILP formulations of Section 3 and solved them with Gurobi 10.0.2 on simulated instances. We ran all experiments on an Intel iCore5 CPU at 2.3 GHz with 4 GB of memory.

We took a systematic approach to simulate instances. To ensure variability in the instances ([Johnson, 1999]), we generated them randomly from different distributions. Firstly, the battery capacity C is simulated from a uniform distribution with support $[0, 1000]$. Then C is taken as upper bound for generating the initial state of battery soc_0. The values d'_t, ex_t were generated by first sampling X_t from a normal distribution with mean $C/4$ and then adding another uniform random variable Y_t with support $[-C/6, C/2]$ and setting $ex_t = \max\{X_t - Y_t, 0\}$ and $d'_t = \max\{Y_t - X_t, 0\}$. The prices for buying and selling are simulated by means of two Gaussian distributions tuned such that their probability density functions overlap. Such a choice is an effort to simulate real market conditions. Indeed, even if on average the prices for buying energy are higher than those for selling, if we consider prices on different periods, it could be that the price for selling is higher.

We start by comparing the performance of Algorithm 2 and Gurobi for the corresponding ILP. We simulated 20 different instances with the number of periods ranging from 100 to 600 and with a capacity of at most 1000. Figure 1 shows the average runtimes for each time horizon selected. As expected, the times for solving instances with Algorithm 2 were significantly shorter and remained consistently within seconds even for the largest simulated values of C and n, whereas Gurobi often exceeded the time limit of 180 seconds to solve them.

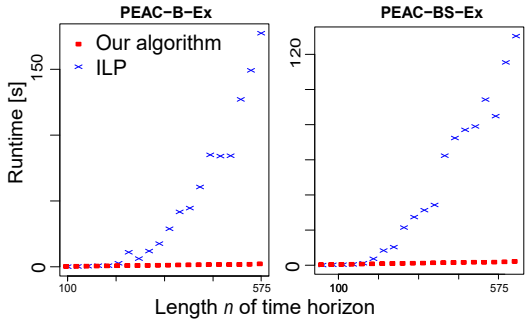

Figure 1: Runtimes for PEAC-B-Ex and PEAC-BS-EX-GEN.

We also conducted simulations regarding PEAC-B-U, PEAC-BS-U, PEAC-B and PEAC-BS with 50 different lengths n of time horizons, ranging from 100 to 10000. For each time horizon considered, we simulated 100 instances and computed the average runtime. As expected from Theorem 4, the battery size had little impact on the algorithm performance. Our experiments showed that the runtime grows linearly in n in the simulated instances both for Algorithm 1 and the IP. We observe that the ILP solver performs better when there are many periods with high demands and few periods with excess energy, otherwise Algorithm 1 performed better. In both cases, the runtimes were similar to each other. We defer full results for these experiments to the full version.

Acknowledgements

Laura Codazzi was supported by the Graduate School "sharing.city.college" of ahoi.digital. Gergely Csáji was supported by the Hungarian Scientific Research Fund, OTKA, Grant No. K143858, by the Momentum Grant of the Hungarian Academy of Sciences, grant number 2021-1/2021 and by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation fund, financed under the KDP-2023 funding scheme (grant number C2258525).

References

[Angenendt *et al.*, 2018] Georg Angenendt, Sebastian Zurmühlen, Hendrik Axelsen, and Dirk Uwe Sauer. Comparison of different operation strategies for pv battery home storage systems including forecast-based operation strategies. *Applied Energy*, 229:884–899, 2018.

[Bartlett *et al.*, 2005] Mark Bartlett, Alan M Frisch, Youssef Hamadi, Ian Miguel, S Armagan Tarim, and Chris Unsworth. The temporal knapsack problem and its solution. In *Proc. CPAIOR 2005*, pages 34–48, 2005.

[Beaudin and Zareipour, 2015] Marc Beaudin and Hamidreza Zareipour. Home energy management systems: A review of modelling and complexity. *Renewable and Sustainable Energy Reviews*, 45:318–335, 2015.

[Bun, 2024a] Durchschnittlicher Preis Haushaltsstrom zum 1. April 2023, May 2024. https://www.bundesnetzagentur.de/DE/Vportal/Energie/PreiseAbschlaege/Tarife-table.html.

[Bun, 2024b] Einspeisvergütung Solarstrom 2023 nach EEG, May 2024. https://www.bundesnetzagentur.de/DE/Fachthemen/ElektrizitaetundGas/ErneuerbareEnergien/EEG_Foerderung/Archiv_VergSaetze/start.html.

[Capper *et al.*, 2022] Timothy Capper, Anna Gorbatcheva, Mustafa A Mustafa, Mohamed Bahloul, Jan Marc Schwidtal, Ruzanna Chitchyan, Merlinda Andoni, Valentin Robu, Mehdi Montakhabi, Ian J Scott, et al. Peer-to-peer, community self-consumption, and transactive energy: A systematic literature review of local energy market models. *Renewable and Sustainable Energy Reviews*, 162:112403, 2022.

[Clautiaux *et al.*, 2021] François Clautiaux, Boris Detienne, and Gaël Guillot. An iterative dynamic programming approach for the temporal knapsack problem. *European Journal of Operational Research*, 293(2):442–456, 2021.

[Codazzi *et al.*, 2024] Laura Codazzi, Gergely Csáji, and Matthias Mnich. Python code for all experiments, 2024. https://github.com/lauracodazzi/CostMinimizationElectricalEnergyMicrogrid.

[Hafiz *et al.*, 2019] Faeza Hafiz, Anderson Rodrigo de Queiroz, Poria Fajri, and Iqbal Husain. Energy management and optimal storage sizing for a shared community: A multi-stage stochastic programming approach. *Applied Energy*, 236:42–54, 2019.

[Hockenos, 2019] Paul Hockenos. *Wired*, 2019. https://www.wired.com/story/in-germany-solar-powered-homes-are-catching-on/.

[Hönen *et al.*, 2023] Jens Hönen, Johann L Hurink, and Bert Zwart. A classification scheme for local energy trading. *OR Spectrum*, 45(1):85–118, 2023.

[Johnson, 1999] David S Johnson. A theoretician's guide to the experimental analysis of algorithms. *Data Structures, Near Neighbor Searches, and Methodology*, 5:215–250, 1999.

[Karp, 1972] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[Morton, 1996] David P Morton. An enhanced decomposition algorithm for multistage stochastic hydroelectric scheduling. *Annals of Operations Research*, 64:211–235, 1996.

[Schrijver, 1998] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[Thielen *et al.*, 2016] Clemens Thielen, Morten Tiedemann, and Stephan Westphal. The online knapsack problem with incremental capacity. *Mathematical Methods of Operations Research*, 83:207–242, 2016.

[Tushar *et al.*, 2021] Wayes Tushar, Chau Yuen, Tapan K Saha, Thomas Morstyn, Archie C Chapman, M Jan E Alam, Sarmad Hanif, and H Vincent Poor. Peer-to-peer energy systems for connected communities: A review of recent advances and emerging challenges. *Applied Energy*, 282:116131, 2021.