

Surrogate Estimators for Collaborative Decision

Fatiha BENDALI, Alejandro OLIVAS GONZALES
LIMOS CNRS 6158, Labex IMOBS3, Clermont-Ferrand, France

Alain QUILLIOT, Hélène TOUSSAINT
LIMOS CNRS 6158, Labex IMOBS3, Clermont-Ferrand, France
alain.quilliot@uca.fr

Keywords: Scheduling, Combinatorial Optimization, Multi-level Optimization, Machine Learning, Energy Management

Received: April 11, 2023

We deal here with job scheduling under the assumption that performing a job requires the production of encapsulated renewable and non-renewable resources. For the sake of understanding, we rely here on a case study related to energy production by a photovoltaic platform. Handling it means synchronizing resource production and consumption in order to optimize both production costs and some scheduling criterion, and induce the setting of a complex bi-level model. Moreover, this applicative context makes appear that job scheduling and resource production most often depend on distinct players, provided with their own agenda and access to information. Adopting here the point of view of one specific player, namely the job scheduler, leads us to set a model that shortcuts the production level with the help of surrogate estimators. Those estimators involve flexible pricing mechanisms and machine learning devices. According to this, we first perform a structural analysis of our model, before designing and testing several algorithms that implement this surrogate component based approach.

Povzetek:

1 Introduction

The notion of multi-level decisional model (see [16], [18]) most often derives from contexts involving several players, independent from each other or tied together by some hierarchical or collaborative link, who share the decision with respect to some system. Solving such a model aims either at providing a best scenario if all the players operate under a common authority (centralized paradigm), or (collaborative paradigm) at helping them into the search for a compromise.

Most contributions address multi-level models according to the centralized paradigm, while assuming the existence of a unique decider provided with full information. Handling a model set this way is a difficult task (see [10], [11]). It usually involves complex decision sub-models of very different types. Standard approaches rely

on decomposition schemes, hierarchical (Benders decomposition, Stackelberg Equilibrium, ...) or horizontal (Lagrangian relaxation) (see [2], [26]). In both cases a major difficulty derives from the sensitivity issue, which means the way one may retrieve information from the different levels in order to make them interact. In case the decision problem involves temporal constraints about jobs to be processed, another difficulty comes from resulting synchronization constraints (see [9]), which require the different players to meet in order to exchange resources or informations.

Yet in practice, it may be utopian, for both technical and economical reasons, to assume that all players will agree on a common agenda and on the share of information. *Game Theory*, mainly *Cooperative Game* (see [7], [8], [21], [33]), provides us with a useful tool for the anticipation of the behavior of the players and for the distribution

of the costs among them. But it does not help a specific player in making its own decision. So, when the focus is on such a specific player, one must find a way to bypass the levels related to the other players and replace, under incomplete information, the criteria and costs related to those players by surrogate estimators.

Present contribution reflects this concern. If we refer to the Frascati Manual of the European Commission, it refers to Section I.2 (Informatics and Information Sciences) while involving developments related to Combinatorial Optimization and Operations Research. It refers to fundamental research in the sense that collaborative decision and complex multi-level models are generic issues, and it has also to see with Applied Research since, in order to make things easier to understand, we start from an applicative context. This context is related to researches conducted about intelligent vehicles inside the IMOBIS3 (*Innovative Mobility Services, Systems and Structures*) Labex in Clermont-Ferrand, and to a partnership between LIMOS Laboratory and the national PGMO (*Gaspard Monge for Optimization*) program promoted by power company EDF (*Electricity De France*). It involves the joint management of energy by a local photovoltaic (PV-Plant) platform and by a consortium of users (industrial players, services providers, ...) relying on this energy in order to perform jobs according to their own purposes. The fact is that market deregulation and emergent technologies currently induce the rise of local renewable energy producers (factories, farms, householders, ...) who simultaneously remain consumers. Those new players make *self-consumption* become an issue (see [5], [22], [30], [36], [44]) whose key operational feature is the need for synchronization between time-dependent resource production and its consumption (see [12], [14], [23], [25]).

So we consider here on one side, a production manager who runs a PV-Plant (Photovoltaic Plant), that not only produces energy and distributes it among end-users (jobs) but also buys and sells energy on the market. On the other side, one or several *job schedulers* are in charge of performing specific jobs. Both meet in order to perform *recharging transactions*: in

order to avoid jobs to waste time while waiting for recharge, the PV-Plant relies on a set of batteries and implement a *swapping* policy (see [4], [40], [41]), so that the job schedulers only need to switch from a battery to another one. We suppose that this *plug out/in* operation is instantaneous. Limited storage and recharge capacities impose both players to carefully synchronize the time-dependent energy production and its consumption. This requirement makes resulting bi-level decision problem complex, even under the centralized paradigm. Many searchers recently showed interest into the decisional problems raised by the management of renewable energy. They most often focused either on production scheduling (see [1], [13], [15]) or on the issues related to consumption (see [3], [17],[19], [27], [31]). But very few dealt with the interaction between both (see [5], [6], [23], [34], [36], [40]) and they most often did it while adopting the point of view of a unique decider provided with full information. Our goal is to study here this interaction and the way players collaborate under incomplete information (see [12], [29], [34], [35]), in order to derive heuristic algorithms reflecting the point of view and the knowledge of a specific player. Though part of the difficulty of our specific application is due to uncertainty, we suppose, for the sake of simplicity, that our system behaves in a deterministic way.

Starting from this applicative context, we set a bi-level **PVSync** model. This model is a centralized macroscopic model, which relies on simplifications with respect to the behaviors of respectively the batteries and the PV-Plant (linearization of the charge and discharge processes for the batteries (see [37], [38], [39]), deterministic power prediction for the PV-Plant (see [44])). Those simplifications are justified not only by our wish to get a tractable model, but also by the fact that we intend adopting the point of view of the job scheduler and designing heuristic algorithms that reflect this point of view. We notice that this **PVSync** model may be viewed as a new variant of the well-known **RCPSP**: *Resource Project Scheduling Problem* (see [32], [24]). **RCPSP** is about scheduling jobs under temporal and resource constraints, and our model introduces non-renewable resources (energy) encapsulated

into renewable ones (the batteries).

We first study **PVSync** according to the centralized paradigm. We cast it into the MILP: *Mixed Integer Linear Programming* framework, and analyze the structure of its main components. Our main purpose being the design of heuristics reflecting the point of view of the job scheduler, we introduce a *projection* mechanism that projects the constraints related to the batteries, that the job scheduler does not control, into a surrogate *Idle Battery* constraint that he may easily handle. Next, we address the collaborative issue. We start doing it while supposing that the production manager acts as a mediator, and cast our problem into the *Cooperative Game* framework. We keep on while adopting the point of view of the job scheduler, who implicitly endorses the role of the master of the game. Our goal becomes the design of *job scheduler* oriented heuristic algorithms *SurrPVCost* that handle **PVSync** according to this restricted point of view, while short-cutting the part of the process involving the PV-Plant. We use a *surrogate* formulation of the costs together with *surrogate* constraints that aims at making the hidden production level remain feasible. The *surrogate* constraints are the *Idle Battery* projected constraints that we just mentioned above, augmented with additional constraints. As for the *surrogate* formulation of the costs, we try two approaches: the first one is based upon a pricing mechanism; the second involves machine learning and convolutional neural networks (see [28], [43]). Both approaches are well-fitted to the management of the non-deterministic case.

So the paper is organized as follows. In Section 2, we introduce the **PVSync** problem. We set it according to the centralized MILP (*Mixed Integer Linear Programming*) framework, and discuss different formulations. In Section 3, we analyze the structure of the main components of this model and the properties of the *Idle Battery* constraint. We start discussing in Section 4 the cooperative issue and first cast in Section 4.1 our problem into the *Cooperative Game* framework. Next we introduce in Section 4.2 a generic *collaborative* algorithmic scheme for the handling of **PVSync** while using surrogate estimators. We

describe in Section 4.3.1 the surrogate constraints that we derive from the projection scheme of Section 3, and propose in sections 4.3.2 and 4.3.3 2 estimators of the production costs: the first one relies on a parametric pricing mechanism while the second one involves a convolutional neural network. We describe in Section 4.4 the way those estimators may be used in order to drive a *job scheduler* oriented heuristic algorithm *SurrPVCost*, and devote Section 5 to numerical experiments.

2 The PVSync Problem

This section is devoted to the description of our case study, related to energy management, which will be the starting point for coming developments. As told in the introduction, this case study derives from a partnership between the LIMOS Laboratory and the national PGMO (*Gaspard Monge for Optimization*) promoted by power producer EDF. It provides us with a medium for the implementation and the tests of algorithmic proposals related to a more fundamental issue, namely the management of complex multi-level decision models in both centralized and collaborative contexts. According to this, our main contribution will be about the way we may endorse the point of view of a given specific player, set a model reflecting its agenda and access to information, that consequently reduces the behavior of the other players to what we call *surrogate* constraints and criteria, and accordingly design heuristic algorithms.

While setting a model for this case study, we shall notice that it extends one of the most fundamental scheduling problem, the **RCPSP**: *Resource Constrained Scheduling Problem*.

The Job Scheduler Side: We consider a set of *jobs* $\mathbf{J} = \{1, \dots, J\}$ to be performed exactly once within a time horizon $[0, N]$ divided into unit length periods. We denote by i the period $[i - 1, i]$. Any job j requires t_j periods and is constrained by a time window $\{Min_j, \dots, Max_j\}$: j must start no sooner than period Min_j and end at period no later than Max_j . Some pair of jobs (j_1, j_2) are tied together by some precedence relations $j_1 \ll j_2$ whose

meaning is that j_1 must be finished when j_2 starts. We do not allow *Preemption*: Once a job starts, it cannot be interrupted.

The Resources : Every job j requires some amount of (electric) energy e_j . This energy is stored inside a set $\mathbf{K} = \{1, \dots, K\}$ of identical batteries, that are assigned to the jobs at the time when they start. Every job requires exactly one battery and a battery cannot be simultaneously assigned to several jobs. It comes that no more than K jobs may be running at the same time. A battery must be periodically recharged in order to run as many jobs as possible, and *recharge* takes place when the battery is *idle* and does not run any job. The *storage* capacity of a battery is denoted by C : The amount of energy stored inside a battery cannot exceed C . The *recharge* capacity of a battery is denoted by C^R : The amount of energy which may be loaded into a battery during 1 period cannot exceed C^R . The initial load of battery $k \in \{1, \dots, K\}$ is denoted by H_k^{Init} . For technical reasons, assigning a battery to a job j takes place at the junction between 2 periods, that means at some time $i, i = 0, \dots, N - 1$. Such a *swapping* transaction is instantaneous. The battery remains *active* during the time when j is running, and cannot be recharged during this time. We introduce the following notations:

- $\hat{e} = \sum_j e_j$; $\hat{t} = \sum_j t_j$; $E^{Mean} = \frac{\hat{e}}{\hat{t}}$;
- For any j , $e_j^{Mean} = \frac{e_j}{t_j}$.

Remark 1: We proceed here in a way that is standard with respect to this kind of problems (see [3], [19], [27]) and that consists in simplifying the physics of both the batteries and the PV-Plant. We implicitly refer here to the same kind of Ion/Lithium batteries that may be embedded into electric vehicles, with storage capacities between 20 KWh and 100 KWh, and recharge capacities between 10 KW and 100 KW. However, we do as if that the charge and discharge processes of those batteries were linear, that may be considered as true as long as the current charge of the battery remains located inside some critical interval (see [37], [38]). This implicitly means that capacity C is not the real capacity of the battery, rather the size of the interval inside which the charge and discharge

processes may considered as quasi-linear. Also, we do not allow here any choice choice between different recharge capacities involving distinct economic costs.

The PV-Plant Side: In order to implement a *self-consumption* policy we rely on a *PV-Plant*, that means on a photovoltaic facility. This PV-Plant assigns the batteries to the jobs according to some swapping policy and produces its own energy that it distributes among the currently idle batteries or that it sells to the market. In case this self-produced energy is not enough, the PV-Plant can also buy energy to the market. Energy stored inside the batteries can be neither transferred to another battery nor sold. We denote by R_i the

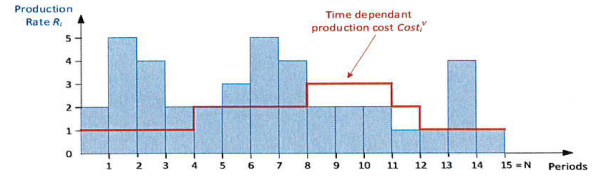


Figure 1: Time dependent production rates and costs for the micro-plant

expected production of the PV-Plant at period i , by P_i the energy unit purchase price at period i , and by S_i the energy unit sale price. Of course we have, for any i , $P_i \geq S_i$. An example of cost and production rates is provided by Figure 1. We set, for any period i :

- P^{Mean} = mean value $P_i, i = 1, \dots, N$;
- S^{Mean} = mean value $S_i, i = 1, \dots, N$;
- R^{Mean} = mean value $R_i, i = 1, \dots, N$.

Figure 2 describes the way the PV-Plant and the jobs interact in the case when the jobs are tours performed by electric vehicles, that start and end into the PV-Plant and make the vehicles visit a set of customers. It shows the swapping policy that makes some *active* batteries be embedded into the vehicles, while the other *idle* batteries remain available for recharge.

Remark 1-Bis: We bypass the physics of the PV-Plant (see [39], [44]) and consider its behavior as deterministic, following the way standard

software like PVLib Matlab ([44]) proceed.

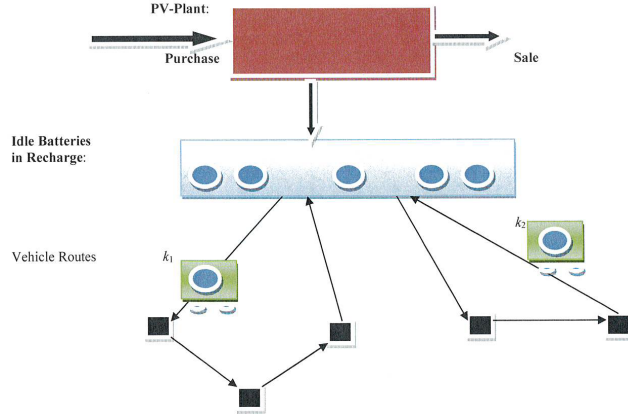


Figure 2: PV-Plant, batteries and *vehicle routing* jobs

Then we may set the **PVSync** problem:

PVSync: {Simultaneously schedule the jobs and the PV-Plant, in such a way that:

- Every job j is run once without any interruption (*Non-Preemption*) within its time window. Precedence constraints $j_1 < j_2$ are satisfied. (C1)
- Every time a job j is performed, it is provided with a battery $k(j)$ loaded with at least e_j energy. (C2)
- The global energy load of the batteries at the end of period N must be at least equal to $\sum_k H_k^{Init}$. (C3)
- Some hybrid cost $\alpha \cdot SchCost + PVCost$ is minimized. $PVCost$ is the *self-consumption* cost, equal to the difference between the energy purchase costs and the profits derived from the sales. *Scheduling* cost $SchCost$ is the sum of the completion times of the jobs.}

Remark 2: PVSync and the RCPSP problem: The **PVSync** problem may be viewed as new variant of the well-known **RCPSP**: *Resource Constrained Scheduling Problem* (see [32], [24] for standard RCPSP), that is itself an extension of the *Multi-Processor Scheduling* problem. Jobs depend here on *encapsulated* resources, which are renewable batteries and non-renewable energy embedded into the batteries. The first ones act as *containers* for the second ones. Setting this

extension in a formal way would imply introducing a lot of notations, without any fundamental impact on the methods. So, instead of doing it, we keep on with the **PVSync** setting and terminology, while referring to the batteries as renewable container resources and to the energy as non renewable content resource.

Remark 3 : We may set **PVSync** as a bi-level model by distinguishing the respective roles of the job scheduler and the PV-Plant :

Bi-level Reformulation of PVSync:
 {Schedule the jobs in such a way :

- (C1) is satisfied.
- An hybrid cost $\alpha \cdot SchCost + PVCost$ is minimized, $PVCost$ being the optimal value of the following **PVPlant** sub-problem :

PV-Plant sub-problem : {Schedule the sale, purchase and distributions operations, and assign the batteries to the jobs in such a way that (C2) is satisfied and that the difference between the energy purchase costs and the profits derived from the sales is minimal.}}

Remark 4: One might clearly make the PV-Plant side more complex, by broadening the scope of feasible sale, purchase and distribution operations, by getting closer to the physics of storage and recharge/discharge processes, by involving Heterogeneous batteries, with distinct storage and recharge capacities, and by allowing distinct recharge capacities, each recharge capacity involving its own specific costs.

Our Goal and Contribution : As a new bi-level extension of the RCPSP problem, the **PVSync** problem is interesting by itself. Yet, our purpose here is not the design of exact algorithms for this problem, rather the study of the way one might efficiently handle it if the PV-Plant processes become more complex (see Remark 4) or in the case of a collaborative context, that would impose us to focus on the job scheduler point of view and cope with restrictions on the information related to the behavior of the PV-Plant. This means that

our main contribution here is the design of heuristic algorithms that tends to emulate the decentralized point of view of a specific player, namely the job scheduler, possibly provided with partial information. In order to achieve it, we first cast **PVSync** into the MILP framework (Section 2.2), in order to get an unambiguous setting and to provide ourselves with reference results, and discuss some variants (Section 2.3). Next we perform some structural analysis (Section 3) with focus on the introduction of a mechanism that projects the constraints related to the batteries (that the job scheduler does not control) into surrogate *Idle Battery* constraints that he may easily handle. The introduction of this projection mechanism opens the way (Sections 4 and 5) to the design and test of *job scheduler* oriented heuristic algorithms *SurrPVCost* for the handling of **PVSync**, that reflect the incomplete point of view of the job scheduler. These algorithms rely on the management of surrogate constraints, (the *Idle Battery* constraints augmented with some additional constraints), and of surrogate objective functions, that express the approximation that the job scheduler may get of the part of the costs that it does not fully control. Though we stick here to our case study, this approach is generic and might be applied to other multi-level/multi-player decision problems.

2.1 An Example

Let us suppose that $J = \{A, B, C, D, E\}$, with respective durations $t_j = 2, 1, 2, 3, 1$ and energy requirements $e_j = 5, 5, 4, 9, 4$. Jobs A and B must be run between periods 3 and 5, job C between periods 5 and 8, job D between periods 2 and 7, and job E between periods 7 and 10.

We are provided with 2 identical batteries k_1 and k_2 , with initial loads respectively equal to 7 and 6. We have: $C = 12$, $C^R = 3$. The time space is divided into 10 periods and time versus money coefficient α is equal to 2. Production data come as in table 1:

Table 1: Prices and Production Coefficients.

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------|---|---|---|---|---|---|---|---|---|----|
| P_i | 2 | 3 | 7 | 7 | 3 | 2 | 6 | 7 | 4 | 2 |
| S_i | 1 | 2 | 4 | 4 | 1 | 1 | 3 | 3 | 2 | 1 |
| R_i | 4 | 4 | 3 | 5 | 2 | 6 | 4 | 4 | 4 | 5 |

Then we get (Fig. 3) a feasible **PVSync** solution:

- Battery k_2 consecutively runs jobs A and B between periods 3 and 5 and comes back to the PV-Plant at the end of period 5. It reloads until period 7, runs job E and comes back to the PV-Plant.
- Battery k_1 runs job D at period 2 and comes back to the PV-Plant at the end of period 4. Then it reloads before running job C at period 8.

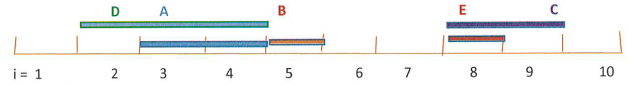


Figure 3: A feasible schedule of the jobs

For every period i , energy amounts respectively bought, sold and distributed to the batteries are given by table 2.

Table 2: Scheduling the Production.

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|----------------------------|---|---|---|---|---|---|---|---|---|----|
| Bought | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Sold | 0 | 1 | 3 | 5 | 0 | 0 | 1 | 4 | 2 | 0 |
| To k_1 | 2 | * | * | * | 3 | 3 | * | * | 3 | * |
| To k_2 | 3 | 3 | * | * | * | 3 | * | 2 | 3 | * |

2.2 A MILP Model

The purpose of this section is to cast the **PVSync** problem into the MILP framework. This will provide us with a non ambiguous formal setting of this problem, that we will refer to all along the paper, as well as with reference results for numerical experiments. Notice that it would have been possible to choose another framework, like for instance the SAT framework (while using the library Google’s CP-SAT, part of the OR-Tools Library) or the Constraint Programming framework (while relying on the IBM CPO software). But the fact is that the MILP framework is a kind of standard in Operations Research and Combinatorial Optimization, and is also widely used in the Industry, specifically in companies involved in Telecommunications, Transportation and Energy Production.

Since **PVSync** may be viewed as a **RCPSP** extension, we adapt the standard **RCPSP MILP**: *Mixed Integer Linear Program* formulation corresponding to the case when the time space is explicitly divided into unit-time periods. In such a case, **RCPSP MILP** usually relies on a $\{0, 1\}$ -valued vector $Z = (Z_{j,i}, i = 1, \dots, N, j = 1, \dots, J)$ which tells us at which period i a job j starts. In our case, we must also explicitly identify the batteries so that they may be assigned to the jobs. We link the jobs, the periods and the batteries through a $\{0, 1\}$ -valued vector $X = (X_{j,k,i}, j = 1, \dots, J, k = 1, \dots, K, i = 1, \dots, N)$: $X_{j,k,i} = 1$ means that battery k is assigned to job j , starting at period i .

On the other side, we represent the production activity of the PV-Plant by 3 non negative rational vectors $U = (U_i, i = 1, \dots, N)$, $V = (V_i, i = 1, \dots, N)$, $Q = (Q_{k,i}, k = 1, \dots, K, i = 1, \dots, N)$, with respective meanings:

- U_i means the energy purchase at period i ,
- V_i represents the energy sale at period i ,
- $Q_{k,i}$ represents the energy distributed to battery k at period i .

We link the PV-Plant, the jobs and the batteries through a last variable $W = (W_{k,i}, k = 1, \dots, K, i = 0, \dots, N) \geq 0$ with rational values: $W_{k,i}$ represents the energy inside battery k at the end of period i ($W_{k,0}$ means the initial load of battery k). Resulting model comes as follows:

PVSync_MILP Model: {Compute:

- $\{0, 1\}$ -valued vector $Z = (Z_{j,i}, i = 1, \dots, N, j = 1, \dots, J)$: $Z_{j,i} = 1$ means that job j starts at period i .
- $\{0, 1\}$ -valued vector $Y = (Y_{j,k}, k = 1, \dots, K, j = 1, \dots, J)$: $Y_{j,k} = 1$ means that battery k is assigned to job j .
- $\{0, 1\}$ -valued vector $T = (T_{k,i}, k = 1, \dots, K, i = 1, \dots, N)$: $T_{k,i} = 1$ means that battery k is active at period i .
- $\{0, 1\}$ -valued vector $X = (X_{j,k,i}, k = 1, \dots, K, j = 1, \dots, J, i = 1, \dots, N)$: $X_{j,k,i} = 1$ means that battery k is assigned to job j , starting at period i .

- $U = (U_i, i = 1, \dots, N) \geq 0$ with rational values: U_i means the energy purchase at period i .
- $V = (V_i, i = 1, \dots, N) \geq 0$, rational: V_i means the energy sale at period i .
- $Q = (Q_{k,i}, i = 1, \dots, N, k = 1, \dots, K)$, rational: $Q_{k,i}$ means the energy distributed to battery k at period i .
- $W = (W_{k,i}, i = 0, \dots, N, k = 1, \dots, K) \geq 0$, rational: $W_{k,i}$ means the load of battery k at the end of period i .

Under the constraints:

- Minimize $\sum_i (P_i \cdot U_i - S_i \cdot V_i) + \alpha \cdot (\sum_{j,i} i \cdot Z_{j,i})$. (E1)
- For any i, j , $Z_{j,i} = \sum_k X_{j,k,i}$. (E2)
- For any k, j , $Y_{j,k} = \sum_i X_{j,k,i}$. (E3)
- For any battery k , period i_1 , $T_{k,i_1} = \sum_{j,i=i_1-t_j+1, \dots, i_1} X_{j,k,i}$. (E4)
- For any j , $\sum_k Y_{j,k} = 1 = \sum_i Z_{j,i}$. (E5)
- For any j , any i s.t. $(i < Min_j)$ or $(i > 1 + Max_j - t_j)$, $Z_{j,i} = 0$. (E6)
- For any k, i , $Q_{k,i} \leq C^R \cdot (1 - T_{k,i})$. (E7)
- For any i, k , $W_{k,i} \leq C$. (E8)
- For any k , $W_{k,0} = H_{k,0}^{Init}$ and $W_{k,N} \geq H_{k,0}^{Init}$. (E9)
- For any $i \geq 1, k$, $W_{k,i} = W_{k,i-1} + Q_{k,i} - (\sum_j e_j \cdot X_{j,k,i})$. (E10)
- For any i , $U_i + R_i = V_i + (\sum_k Q_{k,i})$. (E11)
- For any j_1, j_2 s.t. $j_1 < j_2$, $\sum_i i \cdot Z_{j_1,i} + t_{j_1} \leq \sum_i i \cdot Z_{j_2,i}$. (E12)}

Proposition 1: Solving **PVSync_MILP** solves the **PVSync** problem.

Proof: One easily turns any feasible solution of **PVSync** into a feasible solution (Z, Y, T, X, U, V, W, Q) of **PVSync_MILP** with same cost. Checking that every constraint (E1), ..., (E12) is satisfied derives in a straightforward way from the meaning of those constraints: (E2) means that if job j starts at period i , then there is exactly one battery assigned to j . (E3) means that if battery k is assigned to job j , then

there exists a unique period i such that j starts with battery k at period i . (E4) means that if battery k is active at period i_1 , then there exists a unique job j which starts with battery k at i such that $i \leq i_1 \leq i + t_i - 1$. (E5) means that to any job j must be assigned exactly one battery and one starting period. (E6) means that any job j is run within its time window. (E7, E8) are capacity constraints: the load inside a battery at the end of a given period cannot exceed the *storage* capacity of the battery, and the amount of energy loaded into a battery during a given period cannot exceed the *recharge* capacity. (E9) expresses the initial and final requirements for the batteries. (E10, E11) are nothing more than balance equations, which distribute energy over the time between purchase, sale, storage and consumption by the jobs. (E12) means that if two jobs j_1, j_2 are such that j_1 precedes j_2 , then the starting period for j_2 must be larger than the ending period of j_1 . Conversely, the key point is that (E5) assigns exactly 1 battery k and 1 starting period i to every job, and that variables $X_{j,k,i}$ involved in (E2, E3, E4) allow us to characterize the periods when a battery is active or idle. This in turn enables us to control through (E7, E11) the amount of energy loaded into any battery at any period, as well as the *Non-Preemption* constraint. The rest of the proof comes in a straightforward way. **End-Proof.**

2.3 A Short Discussion: Variants of the PVSynC Problem

In practice, production configurations may be more complex (see Remark 4), and we may be imposed to get closer to the physics of both the batteries and the PV-Plant. Let us mention here 3 possible variants, whose handling would provide us with a stronger motivation for the approaches that we are going to describe in Section 4:

First Variant: *Batteries may be used in order to store energy and sell it later.* According to this hypothesis, a battery k may receive energy at period i and sell it at period $i' > i$. The **PVSynC_MILP** model must be updated through the introduction of an additional vector $V^S = (V_{k,i}^S, k = 1, \dots, K, i = 1, \dots, N)$, meaning the amount of energy sold at any period i by battery k .

Second Variant: *A fixed storage unit may be used either for later sale or battery feeding.* According to this hypothesis, such a *Buffer* battery $BUFF$, with storage capacity C^{BUFF} , initial load H^{BUFF} , recharge capacity C^{ReBUFF} and discharge capacity C^{DeBUFF} , induces the introduction of new variables into the model:

Q_i^{BUFF} = energy sent from the PV-Plant to the *Buffer* battery at period i .

V_i^{BUFF} = energy sold by the *Buffer* battery at period i .

$Q_{k,i}^{BUFF}$ = energy sent by the *Buffer* battery to battery k at period i .

W_i^{BUFF} = energy inside the *Buffer* battery at the end of period i .

Third Variant: *Periods units and time values do not coincide.*

Actually, the structure of the time space for the PV-Plant may not coincide with the structure of the time space for the jobs. More precisely, a period means a time interval during which the PV-Plant is stable (no change neither in the production rates nor in the purchase/sale prices). In most cases, the duration of a period is given as an integral number p , while the duration of a job may take any integral value. In such a case, representing the schedule of the jobs with a vector $Z = (Z_{j,i}, i = 1, \dots, N, j = 1, \dots, J)$ as in the previous section does not hold anymore and getting a **PVSynC** MILP formulation requires the introduction of flow vectors representing the way the jobs exchange the resources.

3 Structural Analysis

We analyze in this section the structure of **PVSynC** and its complexity, while adopting the standard centralized point of view (one decider, provided with all information). We describe the projection mechanism (*Idle Battery* constraints) that will allow us to bypass the battery level. We first introduce the following notations:

For any subset A of the job set \mathbf{J} , we denote by **PVSynC**(A) the restriction of **PVSynC** to the jobs of A , and by $PVSynC(A)$ its value.

For any schedule vector Z we denote by **Restrict-PVSync**(Z) the problem which derives from **PVSync** by fixing Z , and by *RestrictPVSync*(Z) its value.

For any schedule vector Z and any period i , we denote by $n(Z, i)$ the number of jobs which are active at period i according to Z , and by $L(Z, i) = K - n(Z, i)$ the number of batteries which are idle at period i , that means which are available for recharge.

Given a schedule vector Z and a job j , we denote by $Start(Z, j)$ the starting period of j according to Z .

3.1 The Different Levels of PVSync and their Complexity

PV-Sync is a multi-level problem. We may distinguish a scheduling level (variables Z), a battery level (variables Y) and a production one (variables U, V, Q). This last one is nothing but a linear program with rational variables.

The Scheduling Level

Restricting **PVSync** to its scheduling level means short-cutting the activity of the PV-plant, and only considering the jobs, provided with durations, time windows and precedence constraints, together with K identical batteries which lose their container status and behave as machines in the standard scheduling sense. In the case of variant 3 of former section 2 this restriction of **PVSync** to its scheduling level would contain the multi-machine scheduling problem, and so would be NP-Hard (see [20], [42]). Since our time space is explicitly divided into periods, we must be more careful. Let us consider the following setting **PVSync-Sched** which corresponds to this restriction of **PVSync**:

PVSync-Sched

- **Inputs:**
 - The period set $\{1, \dots, N\}$
 - The battery set $\mathbf{K} = \{1, \dots, K\}$
 - The job set $\mathbf{J} = \{1, \dots, J\}$: A job j requires t_j periods and we suppose that $\sum_j t_j \leq K.N$. No precedence constraints are imposed.

- **Outputs:** We want to schedule the jobs of \mathbf{J} inside the periods $1, \dots, N$, in such a way that no more than K jobs are performed during a same period i .

Then we easily check that any instance of the well-known strongly NP-Complete **Bin Packing** problem (see [20], [42]) can be polynomially reduced to an instance of **PVSync-Sched** so that **PVSync-Sched** is also strongly NP-Complete.

The Battery Level.

Let us consider now the *Battery* level, which means that we suppose that the jobs of \mathbf{J} have been scheduled and that we deal with variables $Y_{j,k}$, that distribute the batteries among the jobs. We focus on the constraints (E3, E4, E5, E9, E10) of **PVSync**, without taking care of the activity of the PV-Plant, (infinite free production rates R_i), and while restricting ourselves to 2 batteries. More precisely, we consider the following restriction **PVSync-Battery** of **PVSync**:

PVSync-Battery

- **Inputs:**
 - The period set $\{1, \dots, N\}$.
 - The job set $\mathbf{J} = \{1, \dots, J\}$: A job j requires e_j energy. Every job j has been scheduled inside a single period $i(j) \in \{1, \dots, N-1\}$. We suppose that $J = 2.(N-1)$ and that no more than 2 jobs have been scheduled during a same period.
 - Two identical batteries k_1, k_2 , initially loaded with a same energy amount H_0 .
- **Outputs:** We want to assign the batteries to the jobs in a way that is consistent with the energy requirements.

Theorem 1: *PVSync-Battery is NP-Complete.*

Proof: We may consider the case when the initial load H_0 is equal to $\frac{\hat{e}}{2}$ with $\hat{e} = \sum_j e_j$. Then we see that for any period i , there must exist exactly 2 jobs $j(i)$ and $\bar{j}(i)$ which are scheduled at period i . We may suppose $e_{\bar{j}(i)} \geq e_{j(i)}$. It comes that k_1 and k_2 must be active during periods $1, \dots, N-1$ and can only recharge at period N . During those

periods $1, \dots, N-1, k_1$ and k_2 must globally provide the same energy $\frac{\hat{e}}{2}$. Thus, solving our problem means partitioning the period set $1, \dots, N-1$ into 2 subsets N_1 and N_2 in such a way that $\sum_{i \in N_1} (e_{\bar{j}(i)} - e_{j(i)}) = \sum_{i \in N_2} (e_{\bar{j}(i)} - e_{j(i)})$. We get a reduction to the **2-Partition** problem (see [20], [42]) and we conclude. **End-Proof.**

3.2 Merging the Batteries: A Projection Mechanism

PVSync is a complex problem, with 2 encapsulated decision levels, respectively related to job scheduling and battery assignment, which both involve their own NP-Complete satisfiability sub-problems. The discussion about the variants of **PVSync** and Remark 4 showed that the production level may itself become more complex, even when remaining inside the **P-Time** class. So, even if we work according to the centralized paradigm, dealing with large size or real time constrained instances of **PVSync** should push us to find a way to partially bypass some levels and among them the battery level. This trend will of course be reinforced as soon as we address the collaborative issue.

According to this purpose, we shortcut the battery level by merging the K batteries into a unique virtual one. Relaxing this way **PVSync** will later help us in a significant way, when it will come to the design of *job scheduler* oriented algorithms that schedule the jobs while partially bypassing the production issue. More precisely, we start from a **PVSync** instance and perform the following construction:

- We replace the K batteries, by a single *macro-battery* with storage capacity $\hat{C} = K.C$. For any period i , \hat{Q}_i will mean the energy amount loaded into this *macro-battery* during period i .
- We forbid more than K jobs to be running during a same period and we impose the *Idle Battery* constraint (E13):
For any period i ,

$$\hat{Q}_i \leq C^R.(K-n(Z, i)). \quad (\text{E13})$$
This constraint bounds the energy loaded into the jobs without explicitly involving the batteries. It implies that no more than K jobs are simultaneously running.

Remark 5: The *macro-battery* is completely virtual and can be viewed as a *projection* of batteries onto the job scheduler. It provides us with a way to shortcut the battery level.

This construction leads us to set a relaxation of **PVSync**, denoted by **PVSync-Merge**. In order to analyze the complexity of the *Idle Battery* constraint (E13), let us restrict **PVSync** to 4 periods, unit-period jobs, no temporal constraints, and a PV-Plant activity reduced to energy purchase. More precisely, let us set the following problem **PVSync-Idle**:

PVSync-Idle

- **Inputs:**
 - The period set $\{1, 2, 3, 4\}$.
 - The unit-period job set $\mathbf{J} = \{1, \dots, J\}$: A job j requires e_j energy.
 - *Battery* parameters K, C, C^R .
- **Outputs:** We deal with a *macro-battery* initially loaded with an energy amount $H_0 = \frac{(\sum_j e_j)}{2}$, and provided with a storage capacity $\hat{C} = H_0$ and a *recharge* capacity coefficient $C^R = \frac{H_0}{J}$. This *macro-battery* can buy energy according infinite purchase costs in periods 1 and 3 and null purchase costs in periods 2, 4. Then we want to schedule under a null cost the jobs of \mathbf{J} inside the periods $\{1, 2, 3, 4\}$, in such a way that:
 - The *Idle Battery* constraint is satisfied: For any period i ,

$$\hat{Q}_i \leq C^R.(K-n(Z, i)). \quad (\text{E13})$$
 - For any period i , the load \hat{W}_i of the *macro-battery* at the end of i does not exceed $\hat{C} = H_0$.

Then we check that solving **PVSync-Idle** means partitioning the job set \mathbf{J} into 2 subsets J_1, J_2 such that $\sum_{j \in J_1} e_j = \sum_{j \in J_2} e_j$. We derive from the NP-Completeness of the 2-Partition (see [20], [42]) problem the NP-Completeness of the **PVSync-Merge** problem.

Of course, both the strong NP-Completeness of **PVSync-Sched** and the NP-Completeness of **PVSync-Idle** imply that **PVSync-Sched** is strongly NP-Hard.

Extending a Schedule Vector Z into a Feasible Solution of **PVSync-Merge**

Since the leader object in **PVSync-Merge** is the schedule vector $Z = (Z_{j,i}, j = 1, \dots, J, i = 1, \dots, N)$, we are now going to characterize the conditions which make possible to extend Z into a full feasible solution of **PVSync-Merge**, in a way which does not involve the variables U, V, \hat{Q}, W . This characterization will help us in dealing with *schedule* vector Z while bypassing the lower levels of **PVSync**. In order to provide it, we need some additional notations:

– For any periods i, i_1 , we set:

- $Conso(i, i_1, Z) = \sum_{j \text{ s.t. } i \leq \text{Start}(Z,j) \leq i_1} e_j$.
 $Conso(i, i_1, Z)$ means the energy consumption by the jobs which start no sooner than i and no later than i_1 .
- $ProdMax(i, i_1, Z) = \sum_{i_2 \text{ s.t. } i \leq i_2 \leq i_1} (K - n(Z, i_1)).C^R$.
 $ProdMax(i, i_1, Z)$ means the maximal energy that the macro-battery may load during periods i, \dots, i_1 .

Then we may state:

Theorem 2. *Schedule vector Z may be extended into a feasible solution of the **PVSync-Merge** problem if and only if:*

- For any period i : $H_0 + ProdMax(1, i, Z) \geq Conso(1, i + 1, Z)$. (E14)
- For any periods i, i_1 , s.t. $i \leq i_1$: (E15)
 $\hat{C} + ProdMax(i, i_1, Z) \geq Conso(i, i_1 + 1, Z)$.

Proof: Above conditions (E14, E15) are clearly necessary: The first one tells us that at any period i , the macro-battery should have received enough energy in order to ensure that any job j starting no later than period $i + 1$ might be achieved. The second one tells us that, if we consider a sequence of periods i, \dots, i_1 , then the macro-battery should receive enough energy during those periods in order to achieve all jobs starting no sooner than i and no later than $i_1 + 1$. The key point is about sufficiency. We proceed by induction on i , and suppose the converse. We suppose that i_1 is such that we could schedule the production in such a way that all jobs starting no later than i_1 could be achieved, and that at the end of period i_1 , we

cannot provide the jobs j starting in $i_1 + 1$ with the energy that they require. We may impose our production strategy to be such that at any period i , we load the macro-battery with as much energy as possible, taking into account the *Idle Battery* constraint (E13) and the storage capacity \hat{C} . Then we see, by moving backward from i_1 to 1, that either we reach some period i such that (E15) is violated or we reach period $i = 1$ in such a way that (E14) is violated. We conclude. **End-Proof**

4 The Cooperative Issue

There are several way to address this *collaborative* issue. One may adopt the point of view of a specific player, and emulate the interaction that this player is likely to develop with its partners. Another approach consists in keeping on with the centralized paradigm, and supposing that some mediator player fairly distributes costs and profits among the other players. This second point of view corresponds to the *Cooperative Game* theoretical framework. Though our purpose here is to focus on the first approach, we are first going to briefly describe the way **PVSync** may be cast into the *Cooperative Game* framework.

4.1 A Cooperative **PVSync** Game

Let us recall that a *cooperative game* (Ω, Val) (see [33], [21]) is defined as a set of players Ω and a function Val which, to any subset A of Ω , called *coalition* makes correspond its *cost* value $Val(A)$. Then the problem becomes to fairly distribute the cost $Val(\Omega)$ among the players, in such a way that no coalition is tempted to leave the game. Several approaches may be tried. The most popular one is related to the *core* notion: a price vector $\pi = (\pi_\omega, \omega \in \Omega)$ is in the core of the game (Ω, Val) iff:

- $\sum_\omega \pi_\omega = Val(\Omega)$
- For any coalition A , $\sum_\omega \pi_\omega \leq Val(A)$

This core may be empty, which imposes trying weaker ways to distribute the cost $Val(\Omega)$ among the players, for instance according to the Shapley values. Conversely, it may contain too many elements, which leads to more restrictive notions such as the *Nucleolus* (see [7],[33]).

In the present case we suppose, for the sake of simplicity, that every job j is identified with exactly one player, and that the players are independent. This implies that no precedence relation $<<$ is imposed to the jobs, since such a constraint would induce a dependence between related players. Then we define a cooperative game $G\text{-}PV\text{Sync}$ by setting:

- $Val(A) = PV\text{Sync}(\mathbf{J}) - PV\text{Sync}(\mathbf{J} - A)$.
 $Val(A)$ represents the marginal cost induced by the jobs of A with respect to $PV\text{Sync}(\mathbf{J})$.

Then we may state:

Theorem 3: *If all sale prices S_i are null, then the core of $G\text{-}PV\text{Sync}$ is not empty.*

Proof: Let us recall that Bondareva/Shapley Theorem (see [8]) provides us with a characterization of the non-emptiness of the core of a cooperative game (Ω, Val) . This characterization comes as follows:

- A non negative vector $\mu = (\mu_A, A \subseteq \Omega)$ is said to be *balanced*, if, for any player ω , $\sum_{A \text{ s.t. } \omega \in A} \mu_A = 1$.
- Then the cooperative game (Ω, Val) has a non-empty core if and only if for any balanced vector μ , $Val(\Omega) \leq \sum_A \mu_A Val(A)$.
(E16)

So let us consider some balanced vector μ together with some optimal solution $Sol = (X, Y, Z, T, U, V, W, Q)$ of **PVSync**. Let us set $\bar{\mu} = \sum_A \mu_A$. (E16) requires:

$$\begin{aligned} & PV\text{Sync}(\{1, \dots, J\}) \leq \\ & \sum_A \mu_A (PV\text{Sync}(\{1, \dots, J\}) - PV\text{Sync}(\{1, \dots, J\} - A)). \end{aligned}$$

This equality is equivalent to: (E17)

$$\sum_A \mu_A (PV\text{Sync}(\{1, \dots, J\} - A) \leq (\bar{\mu} - 1) \cdot PV\text{Sync}(\{1, \dots, J\}).$$

So we only need to check that it is possible to decompose $(\bar{\mu} - 1) \cdot Sol$ into a non negative linear combination $\sum_A \mu_A \cdot Sol(\{1, \dots, J\} - A)$, where every vector $Sol(\{1, \dots, J\} - A)$ is a feasible solution of the restriction of **PVSync** to the jobs of $(\{1, \dots, J\} - A)$. If we can do it, then we conclude since every value $PV\text{Sync}(\{1, \dots, J\} - A)$ is going to be no larger than the cost of $Sol(\{1, \dots, J\} - A)$. Getting such a decomposition can be done by tracking inside Sol , for every

job j , the energy consumed by j and the energy produced in order to match this consumption. As a matter of fact, we might also represent the way energy circulates according to (U, V, W, Q) as a flow vector in a network derived from the periods and the batteries involved in **PVSync** (*time expanded* network) and decompose it along the jobs of \mathbf{J} into a multi-commodity flow vector (see [2], [9]). This process makes appear, for every battery k , the useless energy $C^{Res}(k)$ remaining inside k throughout the whole time space. It yields a decomposition of Sol as $Sol = \sum_j SolJob_j$, where every partial solution $SolJob_j$ meets reduced capacities $C - C^{Res}(k), k = 1, \dots, K$. Then $Sol(\{1, \dots, J\} - A)$ comes as $\sum_{j \in \{1, \dots, J\} - A} SolJob_j$. **End-Proof**

Remark 6: If we allow non-null sale price, then we may check that above reasoning does not hold anymore.

4.2 The Job Scheduler/PV-Plant interaction: Surrogate Estimators

We now address the collaborative issue while adopting the point of view of the job scheduler and trying to emulate the interaction that this player is likely to develop with its partner. This point of view corresponds to the most natural bi-level setting of **PVSync**. According to it, we rely on a surrogate estimator $SurrPV\text{Cost}$, which with any schedule vector Z consistent with (E5, E12), and any value of some flexible parameter vector γ , associates an estimation $SurrPV\text{Cost}(Z, \gamma)$ of the value $RestrictPV\text{Sync}(Z)$. The role of the parameter γ is to introduce flexibility in order to mitigate the fact that $SurrPV\text{Cost}(Z, \gamma)$ provides us with no more than an approximation of $RestrictPV\text{Sync}(Z)$. The effective value of γ will be tuned all along the interaction process, in such a way that this process converges to a *consensual* solution. So, at every step during this process, we shall compute Z while restricting ourselves to constraints (E5, E12) together with some additional (surrogate) constraints $Cons(\gamma, Z)$. Those surrogate constraints will aim at both avoiding **Restrict-PVSync**(Z) to be unfeasible and at providing a container for the interaction between the job scheduler and its partner. We shall perform this computation

of Z in such a way that it minimizes the cost $\alpha.(\sum_{j,i} i.Z_{j,i}) + SurrPVCost(Z, \gamma)$. Taken as a whole, this interaction will proceed as follows (see Fig. 4):

Collaborative Algorithmic Scheme

Initialize γ and constraints $Cons(\gamma, Z)$ about schedule vector Z and γ ;

Set the initial proposal of the job scheduler: Compute Z which minimizes $\alpha.(\sum_{j,i} i.Z_{j,i}) + SurrPVCost(Z, \gamma)$ while meeting (E5, E12) and $Cons(\gamma, Z)$;

While Not *Stop* do

1. Set the counter-proposal from the PV-Plant: Solve **Restrict-PVSync**(Z) and retrieve new constraints to insert into $Cons(\gamma, Z)$;
2. Update *Stop*; If Not *Stop* then
 - Update γ ;
 - Set the new proposal of the job scheduler: Compute Z minimizing $\alpha. \sum_{j,i} i.Z_{j,i} + SurrPVCost(Z, \gamma)$ while meeting (E5, E12) and updated $Cons(\gamma, Z)$;

Retrieve the best schedule vector Z obtained this way.

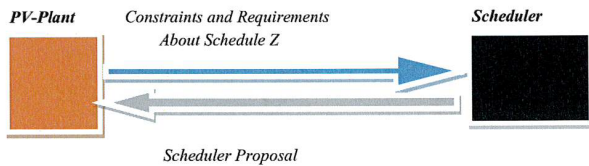


Figure 4: A Collaborative Scheme

Of course, this heuristic management of **PVSync** may also be applied under the standard *centralized* paradigm approach in the case of large size instance or real time requirements.

4.3 Surrogate Estimators and Constraints

We distinguish the surrogate estimator $SurrPVCost(Z, \gamma)$ from the additional constraints $Cons(\gamma, Z)$.

Remark 7: Notice, in the case of the surrogate estimator $SurrPVCost(Z, \gamma)$, that it does not really aim at providing an approximation in the standard sense of $RestrictPVSync(Z)$, rather at efficiently driving above collaborative algorithmic scheme towards good solutions.

4.3.1 Surrogate Constraints

Previous Section 3 leads us to insert (E14) and (E15) into $Cons(\gamma, Z)$. Still, though these constraints ensure the feasibility of the **PVSync-Merge** problem of Section 3, they may not be sufficient in the case of the full **PVSync** problem. Typically, if 2 jobs j_1 and j_2 such that $e_{j_1} = e_{j_2} = 10$, are scheduled to start at period 1, if $K = 2$ and if $H_{k_1}^{Init} = 5, H_{k_2}^{Init} = 15$, then we cannot successfully assign the batteries to the jobs, while we may extend this schedule into a feasible solution of **PVSync-Merge**.

In order to reinforce (E14, E15), we proceed in a heuristic way, while relying on the flexible vector γ :

- We first set, for any energy amount E : $m(E) = \lceil \frac{E}{C^R} \rceil$. This number $m(E)$ means the number of consecutive periods necessary in order to load a battery with E energy. Let also recall that $L(Z, i_1)$ denotes the number of idle batteries induced by Z at period i .
- Then we impose the following parametric heuristic constraints, which depend on flexible parameters γ_0 and γ_1 :
 - For any period i , let j_1, \dots, j_S be the jobs scheduled to start at period $i + 1$ according to schedule vector Z . Then, for any $i_1 \leq i$, we impose: (E18)

$$L(Z, i_1) \geq \gamma_0.|\{s \text{ s.t. } m(e_s) \geq (i - i_1 + 1)\}|.$$
 - For any period i , let us denote by j_1, \dots, j_S the jobs starting at $i + 1$, ordered according to decreasing $m(e_{j_s})$ values, and by $S_1 \leq S$ the largest s value such that $m(e_{j_{S_1}}) \geq i$. Then we impose that there exist k_1, \dots, k_{S_1} in \mathbf{K} such that for any $s \leq S_1$: (E19)

$$HInit_{k_s} + i.C^R \geq \gamma_1.e_{j_s}.$$

Constraints (E18, E19) aims at making in such a way that at any period i , there should exist batteries which have been idle for enough time in order to feed jobs starting in $i + 1$.

4.3.2 A Price Based Surrogate Estimator

We proceed here in an empirical way, while following the idea that the production cost induced by a schedule vector Z is determined by the distribution of values $L(Z, i), i \in \{1, \dots, N\}$. This suggests us to express the surrogate cost $SurrPVCost(Z, \gamma)$ as a sum $\sum_i \Pi_{i,L(Z,i)}$, where $\Pi_{i,L}$ is an estimation of the cost induced by L batteries in recharge (idle) at period i . According to this idea, we first make appear a *standard production price* $\Pi_{i,L}^{Stand}$ which, with any idle battery number L and any period i , associates a kind of a reference price of the recharge of L batteries at period i . We do it by noticing that if all batteries receive a same charge E^{Mean} at every period when they are idle, then the production cost $Cost$ should be equal to $\sum_i \Pi_{i,L(Z,i)}^{Stand}$. So we derive *standard production prices* $\Pi_{i,L}^{Stand}$ as follows:

$$\Pi_{i,L}^{Stand} = P_i \cdot (L \cdot E^{Mean} - R_i) \text{ if } L \cdot E^{Mean} \geq R_i \\ \text{and } \Pi_{i,L}^{Stand} = S_i \cdot (L \cdot E^{Mean} - R_i) \text{ else.}$$

In order to flexibilize those prices, we follow intuition which tells us:

- If $L(Z, i) \cdot E^{Mean} \geq R_i$, then $\Pi_{i,L}$ should increase with P_i ;
- If $L(Z, i) \cdot E^{Mean} \leq R_i$, then $\Pi_{i,L}$ should decrease (negative cost) as S_i increases.

This leads us to introduce 2 components γ_2 and γ_3 of flexible parameter γ and to set:

- $\Pi_{i,L} = \Pi_{i,L}^{Stand} \cdot (1 + \gamma_2 \cdot (P_i - P^{Mean}))$ if $L(Z, i) \cdot E^{Mean} \geq R_i$,
- $\Pi_{i,L} = \Pi_{i,L}^{Stand} \cdot (1 + \gamma_3 \cdot (S_i - S^{Mean}))$ else.

4.3.3 A Machine Learning Based Surrogate Estimator

Instead of relying on energy price coefficients $\Pi_{i,L}$ we use a neural network CNN_PVSync in order to provide us with the quality of a scheduled vector Z . Network CNN_PVSync involves 467 synaptic coefficients and is trained

with 9000 **PVSync**(Z) instances, among them 8110 training instances and 890 validation instances, solved with the MILP model of Section 2. This small ratio of 1/20 between the number of synaptic coefficients and the number of training instances eases the training process (stochastic gradient optimization process), making the error gap evolve in a monotonic way along the epochs and stabilize itself in a natural way in the neighborhood of some optimal error gap. The stochastic gradient algorithm behaves as if it were dealing with a standard optimization problem, with a small number of variables and an objective function defined by an average violation of a larger set of constraints. A consequence is that we do not need to observe the evolution of the error gap along the epochs in order to identify the epoch that induces the best error gap.

CNN_PVSync is designed as a *convolutional* neural network (CNN). CNNs have been mostly used for 2D-pattern recognition, since images are very large size inputs and since the convolutional masks are well-fitted to the recognition of local patterns. In the present case, our goal here is to learn the optimal value of a combinatorial optimization problem (the optimal cost related to the production sub-problem induced by fixing the decision of the job scheduler), in order to drive a heuristic scheduling process. An important feature of a CNN is that, at the contrary of most neural networks, it can deal with flexible inputs of different sizes. It is our case here, since the size of our target combinatorial optimization problem may vary. That is why we choose to work with a CNN. Notice that the error gap induced by the the CNN is not at stake here, rather its ability to drive the heuristic algorithm toward good solutions.

A convolutional network usually works in 3 steps. In the first step a same standard *perceptron CM*, called *convolutional mask*, is applied to fixed size segments of the input vector $IN = (IN_m, m \in \{1, \dots, M\})$, where M is the variable size of the input data vector. This *perceptron CM* yields an output vector $OUT = (OUT_m, m \in \{1, \dots, M\})$, with the same size as IN . In the next step, a *pooling* mechanism is applied to OUT , in order to compact it into the fixed size input vector \overline{IN}

of another perceptron $NPool$. In the last step, $NPool$ turns \overline{IN} into the final output \overline{OUT} of CNN . In the present case the final output of CNN_PVSync is a number θ between 0 and 1. This number refers to the formulation of the optimal value $RestrictPVSync(Z)$ as a barycentric combination $LowPVSync(Z) + \theta.(UpPVSync(Z) - LowPVSync(Z))$, where $UpPVSync(Z)$ and $LowPVSync(Z)$ are respectively an upper bound and a lower bound of $RestrictPVSync(Z)$. In case $RestrictPVSync(Z)$ does not exist, that means in case $PVSynC(Z)$ is not feasible, we do as if $RestrictPVSync(Z)$ were equal to $UpPVSync(Z)$ (then θ should be equal to 1). Figure 5 shows the architecture of CNN_PVSync whose main components come as follows:

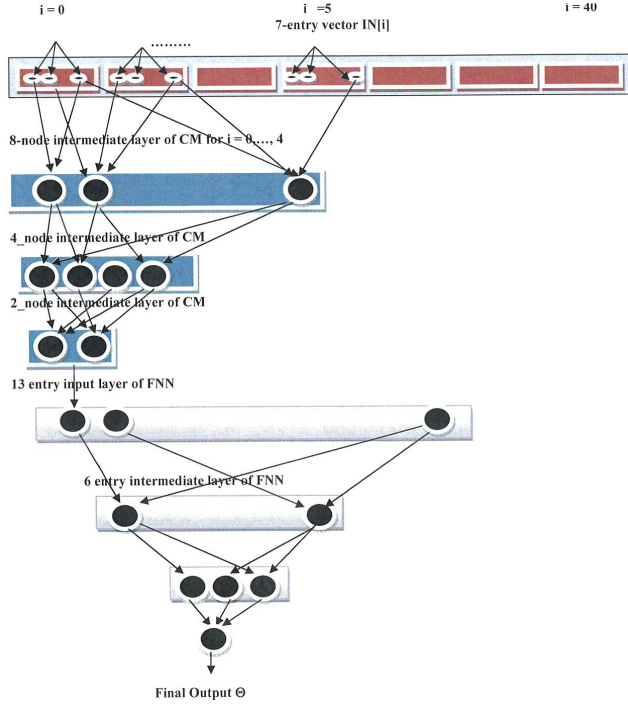


Figure 5: The Neural Network CNN_PVSync .

- **Input layer:** We homogenize any input $(Z, P, S, R, H^{Init}, C, C^R, K)$ of **Restrict-PVSynC**(Z) as a $7(N+1)$ vector IN , with $IN[i] = (\bar{P}_i, \bar{S}_i, \bar{R}_i, \bar{\mu}_i, \bar{L}_i, \bar{C}, \bar{C}^R)$, given by:

- $\bar{P}_i = \frac{P_i}{P^{Mean}}; \bar{S}_i = \frac{S_i}{P^{Mean}}.$
- $\mu_i = \sum_j \text{active at period } i = e_j^{Mean}; \mu_0 = 0; \bar{\mu}_i = \frac{\mu_i}{R^{Mean}}.$ This last quantity $\bar{\mu}_i$ identifies the normalized amount of energy consumed at any period.
- $\bar{R}_i = \frac{R_i}{R^{Mean}}; \bar{R}_0 = \frac{\sum_k H_k^{Init}}{R^{Mean}}.$

- $\bar{L}_i = \frac{L(Z,i)}{K}; \bar{C} = \frac{C}{R^{Mean}}; \bar{C}^R = \frac{C^R}{R^{Mean}}.$

- **Convolutional Mask:** CM works on any vector $IN_i^* = (IN[i], \dots, IN[i+4])$, which means an input with 35 input arcs. It contains 3 inner layers, respectively of sizes 8, 4 and 2, and ends into an output layer, that yields 1 input value OUT_i . This network is complete in the sense that all 322 inner synaptic arcs are allowed, together with standard biased sigmoid activation functions $x \rightarrow \frac{1}{1+\exp(-kx)}$, with parameter k . Thus the number of synaptic coefficients related to this convolutional mask is $320 + 35 = 355$.
- **The pooling Mechanism:** It works by merging consecutive values OUT_i into a single one, in such a way that we get an intermediate vector \overline{IN} , with 13 entries, all with values between 0 and 1.
- **The Final Perceptron $NPool$:** Once the pooling mechanism has been applied, we handle resulting 13 dimensional vector \overline{IN} with a perceptron $NPool$, with input layer of size 13, intermediate layers of size 6 and 3, and a final layer of size 1. This network is complete in the sense that all 99 inner synaptic arcs are allowed, together with standard biased sigmoid activation functions $x \rightarrow \frac{1}{1+\exp(-kx)}$, parameter k being provided with the same value as in the convolutional mask. Thus, the number of synaptic coefficients associated with this final perceptron is $13 + 99 = 112$.

Taken as a whole, CNN_PVSync involves 467 synaptic coefficients.

- **Outputs of CNN_PVSync :** As previously told, the concatenation of CM and $NPool$ yields a value $\overline{OUT} = \theta$ between 0 and 1, which refers to the barycentric setting: $RestrictPVSync(Z) = LowPVSync(Z) + \theta.(UpPVSync(Z) - LowPVSync(Z))$, where $UpPVSync(Z)$ and $LowPVSync(Z)$ are respectively an upper bound and a lower bound of $RestrictPVSync(Z)$, computed as follows:

- $LowPVSync(Z) = -\sum_i S_i.R_i.$

$$\bullet \text{UpPVSync}(Z) = (\text{Sup}_i P_i) \cdot (\sum_j e_j) - (\sum_i S_i \cdot R_i).$$

In case **Restrict-PVSync**(Z) is unfeasible, we do as if *RestrictPVSync*(Z) were equal to *UpPVSync*(Z), that means is if θ were supposed to be equal to 1.

The way we train the network *CNN_PVSync* in order to make it learn its synaptic coefficients will be described in the next section 5, devoted to numerical experiments.

4.4 Job Scheduler Oriented Heuristics for the Search for Schedule Vector Z

Dealing with a blackbox objective function such that *CNN_PVSync* can only be done through a heuristic scheme. So we design a simple *job scheduler* oriented heuristic *SurrPVCost* that works in 2 steps and that we may adapt to both the price based version and the machine learning version of the *SurrPVCost* estimator.

SurrPVCost Job Scheduler Oriented Algorithm

Initialization : Pick up the jobs j according to increasing Min_j values (updated through constraint propagation) and assign them starting periods which maintain constraints (E5, E12, E14, E15, E17, E18, E19), meet time windows and the precedence constraints, and minimize $\alpha \cdot (\sum_{j,i} i \cdot Z_{j,i}) + \text{SurrPVCost}(Z, \gamma)$;

While Not Stop do : Remove some job j from the current schedule and reinsert it (while possibly delaying other jobs) in such a way that $\alpha \cdot (\sum_{j,i} i \cdot Z_{j,i}) + \text{SurrPVCost}(Z, \gamma)$ decreases.

5 Numerical Experiments

Purpose: We want to evaluate the behavior of the *job scheduler* oriented heuristic *SurrPVCost*, implemented along the two surrogate estimators described in sections 4.3.2 and 4.3.3, and its ability to yield under small computational costs a good approximation of the optimal value of **PV-Sync**. According to this purpose, the MILP

formulation of **PV-Sync** is used only in order to provide us with benchmark results.

Technical Context: We use a processor IntelCore i5-6700@3.20 GHz, with 16 Gb RAM, together with a C++ compiler, Linux as O.S. libraries CPLEX20.1 (for ILP models) and TensorFlow/Keras (for machine learning).

Instances: As for the PV-Plant side, we generate 2 integers M and N , N being a multiple of M , with $N = 10, \dots, 40, M = 2, \dots, 5$. Then we split the period set into M macro-periods, corresponding to different mean production rates and prices. Related coefficients R_i, P_i, S_i are generated accordingly. Introducing those macro-periods provides us with realistic prices and production levels that may be related to human activity and to the weather.

As for the scheduling part, we try to both ensure feasibility and put stress on the instances. So we generate J , together with mean duration and mean energy coefficients t^{Mean} and E^{Mean} . We set $C = 2 \cdot \text{Sup}_j e_j$ and $K = \lambda \cdot J \cdot (\frac{t^{\text{Mean}}}{N})$, where λ is some control parameter. For any $k = 1, \dots, K$, we generate H_k^{Init} between $\frac{C}{3}$ and C . In order to make the PV production match the demand, we update the R_i by doing in such a way that $\sum_i R_i = \tau \cdot J \cdot C$, τ being a control parameter with value between 0.5 and 2. Finally we generate the key parameter C^R in such a way that batteries globally receive at least $\beta \cdot J \cdot C$ energy units during the whole process, β being a control parameter with value between 1.5 and 4. So the main parameters of an instance are: $N = \text{Period Number}$, $J = \text{Job Number}$, $M = \text{Macro-period Number}$, $K = \text{Battery Number}$, $t^{\text{Mean}} = \text{Mean Job Length}$, $\alpha = \text{Time versus Money Value}$, $\beta = \text{Recharge Stress Value}$, $\tau = \text{Production Stress}$.

According to this, we generate 10 groups of instances, every instance group *G_Id* containing 30 instances consistent with the characteristics $N, J, M, t^{\text{Mean}}, K, \alpha, \beta, \tau$ described in the following table 3.

5.1 Training the Neural Network CNN_PVSync.

For every group instance *G_Id* described above and for every instance *Id* in *G_Id*, we randomly

Table 3: Instance Group Characteristics.

| G_Id | N | J | M | t^{Mean} | K | α | β | τ |
|---------|-----|-----|-----|------------|-----|----------|---------|--------|
| 1 | 40 | 21 | 3 | 4 | 4 | 1 | 2 | 0,5 |
| 2 | 40 | 23 | 4 | 5 | 4 | 0,5 | 3 | 1 |
| 3 | 40 | 20 | 5 | 6 | 5 | 0,2 | 4 | 2 |
| 4 | 40 | 24 | 3 | 4 | 5 | 1 | 2 | 0,5 |
| 5 | 40 | 32 | 4 | 6 | 4 | 0,5 | 3 | 1 |
| 6 | 40 | 34 | 5 | 8 | 4 | 0,2 | 4 | 2 |
| 7 | 60 | 43 | 4 | 5 | 3 | 1 | 3 | 1 |
| 8 | 60 | 47 | 6 | 10 | 3 | 0,5 | 4 | 2 |
| 9 | 60 | 53 | 4 | 5 | 5 | 1 | 3 | 1 |
| 10 | 60 | 61 | 6 | 10 | 5 | 0,5 | 4 | 2 |

generate 30 schedule vectors Z , which meet the constraints (E5, E12, E14, E15, E17, E18, E19). We do it by turning the initialization procedure of the *SurrPVCost* Algorithm into a partial enumeration procedure, in such a way it can randomly generate several vectors Z meeting constraints (E5, E12, E14, E15, E17, E18, E19). Notice that those vectors may not be extended into a feasible solution of **PVSync** and that in such a case we consider that $RestrictPVSync(Z)$ is equal to $UpPVSync(Z)$, requiring related value θ to be equal to 1. Then we split resulting set of 9000 instances between a training set of 8110 instances and a validation set of 890 instances. The high ratio of 20 between the number of instances and the number of synaptic coefficients significantly eases the training process, making the error gap evolve in a monotonic way until stabilizing itself in a natural way around some approximation of the optimal error gap. We perform the training process while testing 12 sets Hy of hyperparameters, described in the following Table 4. Those hyperparameters are the parameter k of the sigmoid activation function, the batch size Ba , the number of epochs Ep , the loss formula Los (among MS = Mean Square Error, MA = Mean Absolute Error, MSL = Mean Square Logarithmic Error), the optimizer Op (among Ad = *Adam*, Nad = *Nadam*, Amx = *Adamax*, RMS = *RMSprop*). This table also makes appear, for every hyperparameter set Hy , the gaps (in %) between $RestrictPVSync(Z)$ and $CNN_PVSync(Z)$, computed by referring to the value θ . Gap_T means here the value (in %) of this gap related to the training instance, and Gap_V means the same value (in %) related

to the validation instances.

Table 4: Hyperparameter Sets.

| Hy | k | Ba | Ep | Los | Op | Gap_T | Gap_V |
|------|-----|------|------|-------|-------|----------|----------|
| 1 | 1 | 2 | 30 | MS | Ad | 9.6 | 12.9 |
| 2 | 0.5 | 4 | 30 | MS | Ad | 11.2 | 14.8 |
| 3 | 2 | 1 | 30 | MS | Ad | 9.4 | 12.1 |
| 4 | 3 | 2 | 30 | MS | Amx | 10.5 | 14.0 |
| 5 | 1.5 | 4 | 30 | MS | Nad | 8.6 | 11.3 |
| 6 | 2.5 | 8 | 30 | MS | RMS | 9.1 | 12.5 |
| 7 | 1.5 | 4 | 50 | MA | Ad | 8.7 | 11.8 |
| 8 | 1.5 | 4 | 50 | MA | Ad | 8.1 | 11.0 |
| 9 | 1.5 | 4 | 50 | MSL | Ad | 8.4 | 11.2 |
| 10 | 1.5 | 2 | 50 | MSL | Nad | 8.7 | 11.9 |
| 11 | 1.5 | 4 | 50 | MA | Amx | 10.9 | 14.7 |
| 12 | 1.5 | 8 | 50 | MS | RMS | 8.3 | 11.1 |

Comments: The best gap between $RestrictPVSync(Z)$ and $CNN_PVSync(Z)$ corresponds to $Hy = 8$, with value a little bit larger than 8% for the training instances and 11 % for the validation instances. Recall that this error gap is not really at stake, rather the ability of $CNN_PVSync(Z)$ to drive the heuristic *job scheduler* oriented *SurrPVCost* algorithm towards satisfactory solutions.

5.2 Evaluating the Heuristic Management (*job scheduler* oriented algorithm *SurrPVCost* of Section 4.4) of the **PVSync** Problem through Surrogate Components.

For any instance group G_Id described in Table 3, we generate exactly one instance Id and, for every such instance:

- We apply the CPLEX12 library (Table 5) to the **PVSync_MILP** model. Then we get (in less than 1 CPU h), a lower bound LB , an upper bound UB and CPU time T_MILP .
- We apply (Table 6) the *job scheduler* oriented algorithm *SurrPVCost* of Section 4.4 while relying on the pricing mechanism described in Section 4.3.2 and setting $\Pi_{i,L} = \Pi_{i,L}^{Stand}$ for any i, L . We denote by $PR1$ resulting **PVSync** value. We do the same (Table 6)

with 8 combinations of γ values and denote by $PR8$ resulting value. We provide the CPU times T_{PR8} (in seconds) required by the scheduling heuristic *SurrPVCost* described in Section 4.4.

- We apply (Table 7) the *job scheduler* oriented algorithm *SurrPVCost* of Section 4.4 while relying on the machine learning estimator described in Section 4.3.3 and denote by ML related value. We provide the CPU times T_{ML} required by the *SurrPVCost* *job scheduler* oriented algorithm described in Section 4.4.

Those results may be summarized into the following tables:

Table 5: Behavior of the **PVSync_MILP** Model

| Id | LB | UB | T_{MILP} |
|------|-------------|-------------|-------------|
| 1 | - 239.64 | - 235.91 | 3600 |
| 2 | - 85.56 | - 82.47 | 3600 |
| 3 | - 1439.38 | - 1311.60 | 3600 |
| 4 | 488.45 | 619.15 | 3600 |
| 5 | 212.90 | 248.33 | 3600 |
| 6 | 177.46 | 198.35 | 3600 |
| 7 | 1738.56 | 1760.33 | 3600 |
| 8 | <i>Fail</i> | <i>Fail</i> | <i>Fail</i> |
| 9 | 571.31 | 3742.10 | 3600 |
| 10 | - 52.92 | 155.27 | 3600 |

Table 6: Behavior of the Pricing Scheme.

| Id | UB | $PR1$ | $PR8$ | T_{PR8} |
|------|-------------|-----------|-----------|-----------|
| 1 | - 235.91 | - 216.68 | - 216.68 | 48.22 |
| 2 | - 82.47 | - 75.84 | - 78.52 | 71.83 |
| 3 | - 1311.60 | - 1165.59 | - 1208.45 | 76.82 |
| 4 | 619.15 | 659.07 | 659.07 | 62.64 |
| 5 | 248.33 | 406.56 | 327.05 | 131.83 |
| 6 | 198.35 | 221.16 | 200.04 | 83.49 |
| 7 | 1760.33 | 1971.18 | 1904.09 | 579.50 |
| 8 | <i>Fail</i> | - 397.6 | - 397.6 | 1000,6 |
| 9 | 3742.10 | 1267.59 | 1267.59 | 1008.17 |
| 10 | 155.27 | 167.52 | 139.68 | 478.62 |

Comments: As expected, the global ILP model is in trouble, even on small instances. We notice that we may get negative cost values, because of the sales. This makes difficult reasoning in terms

Table 7: Behavior of Machine Learning.

| Id | UB | ML | T_{ML} (s) |
|------|-------------|----------|--------------|
| 1 | - 235.91 | - 193.38 | 10.85 |
| 2 | - 82.47 | - 50.08 | 21.28 |
| 3 | - 1311.60 | - 890.00 | 28.60 |
| 4 | 619.15 | 766.79 | 15.50 |
| 5 | 248.33 | 398.20 | 45.60 |
| 6 | 198.35 | 254.06 | 25.08 |
| 7 | 1760.33 | 1861.24 | 72.25 |
| 8 | <i>Fail</i> | - 359.07 | 105.10 |
| 9 | 3742.10 | 1548.25 | 180.05 |
| 10 | 155.27 | 199.00 | 80.05 |

of percentages. Still, for small J , UB looks close to optimality.

In case the *job scheduler* oriented heuristic *SurrPVCost* relies on the parametric pricing mechanism described in Section 4.3.2, it seems to be rather efficient. Though it only relies on a heuristic *SurrPVCost* algorithm for the computation of schedule vector Z , it yields better results than the MILP model in the 3 instances with largest sizes. Its sensitivity to parameter γ does not look very significant: In 4 cases, the best result is obtained with standard prices $\Pi_{i,L} = \Pi_{i,L}^{Stand}$ for any i, L . In case we apply *SurrPVCost* algorithm while relying on the machine learning oriented estimator described in Section 4.3.3, we see that the gap between UB and ML is more important. Yet, this second approach offers more genericity features than the first one.

6 Conclusion

We dealt here with a complex scheduling problem, involving encapsulated resources and synchronization mechanisms. Since handling it as a whole is difficult and may not fit collaborative contexts, we shortcut the resource production sub-problem and replaced it by a parametric surrogate estimator. We tried 2 approaches: the first one relied on artificial prices; the second one on a neural network. Numerical experiments made appear, at least in our case, a better efficiency of the first approach. Still, because of the genericity of machine learning, a priori better fitted for the management of the uncertainty inherent to our problem, it would be worthwhile to try to go deeper with it. Those two points will

be the backbone of a future research.

Acknowledgement: Present work was funded by French ANR: *National Agency for Research*, by Labex IMOBS3 and by PGMO Program.

References

- [1] Adulyasak Y., Cordeau J. F., Jans R.: The production routing problem: A review of formulations and solutions. *Computers and Operations Research*, 55, p 141-152, (2015).
- [2] Ahuja R.K., Magnanti T.L., Orlin J.B., Reddy M.R.: *Applications of network optimization*. Handbook of Operation Research and Management Sci. 7, p 1-83, 1995.
- [3] Albrecht A., Pudney P.: *Pickup and delivery with a solar-recharged vehicle*. Ph.D. thesis Australian Society for O.R (2013).
- [4] Almuhtady A., Lee S., Romeijn E., Wynblatt M., Ni J.: A degradation informed battery swapping policy for fleets of electric or hybrid electric vehicles. *Transportation Science* 48, 4, p 609-618 (2014).
- [5] Balbiyad S.: *Collective self consumption: computing the optimal energy distribution coefficients considering local energy management*. ENSTA/EDF OSIRIS, (2019).
- [6] Bendali F., Mole Kamga E., Mailfert J., Quilliot A., Toussaint H.: *Synchronizing Energy Production and Vehicle Routing*. RAIRO-O.R, 55 (4), pp. 2141-2163. (2021).
- [7] Berge C. : *Théorie des Jeux à n personnes*. Gauthier-Villars, Paris, Memorial Sciences Maths 138, (1957).
- [8] Bondareva O. N. : Some applications of linear programming methods to the theory of cooperative games. *Problemy Kibernetika* 10, p 119-139, (1963).
- [9] Bsaybes S., Quilliot A., Wagler A.: Vehicle fleet management using flows in time-expanded networks. *TOP*, p 1-24, (2019).
- [10] Caprara A., Carvalho M., Lodi A., Woeinger G. J.: A study on the complexity of the bilevel knapsack problem. *SIAM Journal on Optimization* 24 (2), p 823-838, (2014).
- [11] Chen L., Zhang G.: Approximation algorithms for a bi-level Knapsack problem. *Theoretical Computer Sciences* 497, p 1-12, (2013).
- [12] Chen L., Englund C.: Cooperative intersection management: a survey. *IEEE Transactions on Intelligent Transportation Systems*, 17-2, p 570-586, (2016).
- [13] Chen Z. L.: Integrated production and distribution scheduling: Review and extensions. *Operations Research* 58, p 130-148, (2010).
- [14] Chen Z. L., Vairaktarakis: Integrated production and distribution operations. *Management Science* 51, p 614-628, (2005).
- [15] Chretienne P., Hazir O., Khadad-Sidhoum S.: Integrated batch sizing and scheduling on a single machine. *Journal of Scheduling* 14-6, p 541-55, (2011).
- [16] Colson B., Marcotte P., Savard G.: Bi-level programming: A survey. *4OR Vol 3* (2), p 87-107, (2005).
- [17] Deb S., Tammi K., Kalita K., Mahanta P.: Impact of electric vehicle charging station load on distribution network; *Energies* 11 (1), p 178-185, (2018).
- [18] Dempe S., Kalashnikov V., Perez-Valdez G., Kalashikova N.: *Bi-level Programming Problems Theory*, Springer (2015).
- [19] Erdelic T., Caric T.: A survey on the electric vehicle routing problem. *Journal of Advanced Transportation*, (2019).
- [20] Garey M. R., Johnson D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Co. Ed. (1979).
- [21] Granot D., Granot F.: On some network flow games. *Maths O.R* 17, p 792-841, (1992).
- [22] Grimes C., Varghese O., Ranjan S.: *Light, water, hydrogen: The solar generation of hydrogen by water photoelectrolysis*. Springer-Verlag US, (2008).

- [23] Hall N. G., Potts C. N.: The coordination of scheduling and batch deliveries. *Annals of Operations Research*, 135, p 41-64, (2005).
- [24] Hartman S., Briskorn D. : An updated survey of variants and extensions of the resource-constrained project scheduling problem. *EJOR* 297, 1, p 1-14, (2022).
- [25] Irani S., Pruhs K.: Algorithmic problems in power management. *SIGACT News*, 36, 2, p 63-76, (2003).
- [26] Kleinert T., Labbé M., Ljubic I., Schmidt M.: A survey on mixed integer programming techniques in bilevel optimization. *EURO Journal on Computational Optimization* 9, 21 p, (2021).
- [27] Koc C., Jabali O., Mendoza J., Laporte G.: The electric vehicle routing problem with shared charging stations. *ITOR*, 26 , p 1211-1243, (2019).
- [28] M.Krzyszton M.: Adaptive supervision: method of reinforcement learning fault elimination by application of supervised learning. *Proceedings of the 2018 FEDCSIS AI Conference*, p 139-149, (2018).
- [29] J. Kumar J., Ranga V.: Multi-robot coordination analysis, taxonomy and future scope. *Journal of Intelligent and Robotic Systems*, 102:10, (2021).
- [30] Luthander R., Widen J., Nilsson D., Palm J.: Photovoltaic self-consumption in buildings: A review. *Applied Energy* 142, p 80-94, (2015).
- [31] Macrina G., Pugliese L.D, Guerriero F.: *The green-vehicle routing problem: A survey*. In Modeling and Optimization in Green Logistics, Cham: Springer International Publishing. p. 1-26, (2020).
- [32] Orji.M.J, Wei.S. Project Scheduling Under Resource Constraints: A Recent Survey. *Inter. Journal of Engineering Research and Technology* (IJERT) Vol. 2 Issue 2, (2013)
- [33] Owen G. : *Game Theory*.Academic Press, (1982).
- [34] Rizk Y., Awad M., Tunstel E.: Cooperative heterogenous mutlti-robot systems: a survey. *ACM Computing Surveys* 29, (2019).
- [35] Sarmiento A.M., Nagi R.: A review of integrated production-distribution systems. *IEE Transactions* 31, 1061-1074, (1999).
- [36] Smarter Together: Reports on collective self-consumption of photo-voltaic. *Technical Report Smarter Together*, (2016).
- [37] Souffran G., Liegeville L., Guerin P.: Simulation of real world vehicle missions using a stochastic Markov chain model for optimal power train sizing. *IEE Transactions on Vehicular Technology* 61, 8, p 3454-3465, (2012).
- [38] Tamiselvi S., Gunasundari S., Karuppiah N., Razak A., Madhusudan S., Nagarajan V., Sathish T., Shamim M., Saleel A., Afzal A.: A review of battery modelling techniques. *Sustainability* 13, p 2-26, (2021).
- [39] Tremblay O., Dessaint L.: Experimental validation of a battery dynamic model for E.V applications. *World Electric Vehicle Journal* 3, (2009).
- [40] Trotta M., Archetti C., Feillet D., Quilliot A.: A pickup and delivery problem with electric vehicles and local energy production. *Proc. Triennial Symposium on Transportation Analysis* (TRISTAN), 6 pages, (2022).
- [41] Verma A.: Electric vehicle routing problem with time windows, recharging stations and battery swapping stations. *Euro Journal of Transportation Logistics* 7, p 415-451, (2018).
- [42] Wegener I.: *Complexity Theory*. Springer (2005).
- [43] Wojtuziak J., Warden T., Herzog O.: Machine learning in agent based stochastic simulation: Evaluation in transportation logistics. *Computer and Mathematics with Applications* 64, p 3658-3665, (2012).
- [44] Zsiboracs H., Baranyai H., Vincze A., Haber I., Pinter G.: Economic and technical aspects of flexible storage photovoltaics systems in Europe. *Energies* 11, p 1445-1450, (2018).