Lower and upper bounds for scheduling energy-consuming tasks with storage resources and piecewise linear costs

Sandra Ulrich Ngueveu, Christian Artigues, Nabil Absi, Safia Kedad-Sidhoum

Lower and upper bounds for scheduling energy-consuming tasks with storage resources and piecewise linear costs

Sandra Ulrich Ngueveu[1] Christian Artigues[1] Nabil Absi[2]
Safia Kedad-Sidhoum [3]

[1] LAAS-CNRS, Université de Toulouse, CNRS, INP, Toulouse, France
[2] Mines Saint-Etienne and UMR CNRS 6158 LIMOS, Gardanne, France
[3] CNAM-CEDRIC, Paris, France

Abstract

This paper considers the problem of scheduling a set of time- and energy-constrained preemptive tasks on a discrete time horizon. At each time period, the total energy required by the tasks that are in process can be provided by two energy sources: a reversible one and a non-reversible one. The non-reversible energy source can provide an unlimited amount of energy for a given period but at the expense of a time-dependent piecewise linear cost. The reversible energy source is a storage resource. The goal is to schedule each task preemptively inside its time window and to dispatch the required energy to the sources at each time period, while satisfying the reversible source capacity constraints and minimizing the total cost. We propose a mixed integer linear program of pseudo-polynomial size to solve this NP-hard problem. Acknowledging the limits of this model for problem instances of modest size, we propose an iterative decomposition matheuristic to compute an upper bound. The method relies on an efficient branch-and-price method or on a local search procedure to solve the scheduling problem without storage. The energy source allocation problem for a fixed schedule can in turn be solved efficiently by dynamic programming as a particular lot-sizing problem. We also propose a lower bound obtained by solving the linear programming relaxation of a new extended formulation by column generation. Experimental results show the quality of the bounds compared to the ones obtained using mixed integer linear program.

Keywords Energy-aware scheduling piecewise linear costs storage resources matheuristic column generation lot sizing mixed integer programming
MSC 90B05 MSC 90B30 MSC 90C39 MSC 90C59

1 Introduction and related work

Scheduling problems under energy constraints and/or objectives receive more and more attention, due to the societal and economical stakes of efficient and sustainable energy management. Such energy-aware scheduling problem occurs now in almost every scheduling applications: process industry and discrete manufacturing [14], real time computer systems [2], data centers [9] and of course smart grid and energy markets [24]. Inspired by these applications, we consider a scheduling problem involving a set of time- and energy-constrained preemptive tasks on a discrete time horizon. At each time period, the total energy required by the tasks that are in process can be provided by two energy sources: a reversible one and a non-reversible one. The non-reversible energy source can provide an unlimited amount of energy for a given period but at the expense of a time-dependent piecewise linear cost. The reversible energy source is a storage resource. The goal is to schedule each task preemptively inside its time window and to dispatch the required energy to the sources at each time period, while satisfying the reversible source capacity constraints and minimizing the total cost.

In [17], a variant of the problem without the reversible source was proposed, as well as an extended formulation. From the latter, a branch-and-price method was derived. In this paper we introduce the reversible source, which allows to store and retrieve the energy resource and better adapt the energy providing system to the demands of the energy-consuming tasks, hence reducing the total energy cost. An additional level of complexity is added since the non-reversible source can produce more energy than the demand at certain times and store the excess in the reversible sources to satisfy the demand of a later time. This also invalidates the proofs of theorems of the problem without reversible energy sources and the extended formulation presented in [17] no longer applies as is. Furthermore, as the problem considered in [17] is strongly NP-hard, the problem considered in this paper is also strongly NP-hard.

In the scheduling literature, there is a significant amount of papers dealing with storage resources or reservoirs[1], with a minimum and a maximum capacity. In the classical model originally proposed in [16], a task either consumes or produces a given amount of the resource. A consumer task may start only if the required amount of resource is available in the reservoir given the minimum capacity. As soon as the task starts, the required amount is removed from the reservoir. Conversely, a producer activity fills the reservoir with the produced resource amount exactly at its end time if the maximum capacity is not exceeded. Otherwise, the producer activity cannot be scheduled at this time period. Based on this model, several approaches have been proposed, such as list scheduling algorithms [8], mixed-integer programming approaches [11] and constraint programming algorithms [12, 21]. Recognizing that a full consumption at the beginning of the consumer task or a full production at the end of the producer task is not realistic for some applications, in particular for energy consumption and production, some authors have defined the concept of continuous filling or emptying of the reservoir, which was addressed mainly by constraint programming [23, 4]. A quite generic model described in [6], is to replace the set of tasks by a set of events, such that an event either produces or consumes the resource. This model allows to make a discrete approximation of the continuous model. Indeed, a task that continuously consumes/produces the resources can be decomposed into consecutive consuming/producing events, each having a one period duration in the discretized horizon. For such an event scheduling problem, lower bounds were proposed in [7], mixed-integer linear programming formulations were proposed in [20] and a constraint programming approach using explanations and lazy clause generation was proposed in [21]. The above-described resources accurately allow to model energy production, storage and consumption scheduling for a fixed set of producer and/or consumer events. In the problem considered in the present paper, if the set of tasks could be modeled by consumer events, there is no predefined producer event or, more precisely, there is an optional producer event per time period. Furthermore, the above model states that producer events can only be used to fill the reservoir resource, while in our problem they can also directly supply the demand at the expense of piecewise-linear costs. In summary, in the literature on scheduling with storage resources, there is no concept of global energy demand per time period that can be supplied either by the storage resource or by an optional producer event subject to a general cost.

Such an optional activation of production events was addressed in two different areas. Firstly on the application side, there is an increasing number of papers dealing explicitly with energy management and scheduling in microgrids. We cite recent surveys on this area [13, 18, 25]. Depending on the time horizon, special cases of our model correspond to unit commitment problems for day ahead scheduling or the economic dispatch problem for short term energy dispatching [13]. Mixed integer linear programming is widely used to model with binary unit commitment variables that activate the energy sources when setup costs are present. The energy demand satisfaction constraints often include efficiency functions on the different sources to represent energy losses. However, the energy demand in such cases is a fixed parameter for each period issued from a forecast. There may be controllable loads but the associated decision variables are generally continuous. A typical example is studied in [19], where a continuous variable defined in $[0, 1]$ models for each time period the curtailment or shedding

[1] In [16], such resources are called cumulative resources

of the controllable load. In our case, the load is controlled via the schedule of the energy consuming tasks, in a discrete setting.

Secondly, traditional production planning, and especially lot-sizing, models have been largely and unfortunately overlooked in the recent energy management literature. The lot-sizing research has been particularly active since the 90s, see the recent survey [5]. The basic single-item capacitated lot-sizing setting problem is closely related to a special case of the problem considered in this paper, in which the load is fixed. There is a fixed demand δ_t for an item at each time period t of a discrete horizon of length T, and the demand can be satisfied either by producing the item at the expense of a production cost or by taking the items from the inventory. In [22], a capacitated Lot-Sizing problem with capacities on the production and piecewise linear production costs was shown to be NP-hard and a pseudo polynomial dynamic programming algorithm was provided with a complexity of $O(T^2\bar{\delta}\bar{k})$, where $\bar{\delta} = \frac{1}{T}\sum_{t\in\mathcal{T}}\delta_t$ is the average demand and $\bar{k} = \frac{1}{T}\sum_{t\in\mathcal{T}}K_t$ is the average number of breakpoints. This model does not fit the considered problem, we have an unlimited production capacity but the storage resource has a limited capacity. This issue was recently addressed in [1], where the algorithm of [22] is successfully adapted to the case of a storage capacity with the same worst case time complexity. The only restriction is to have integer breakpoints on each piecewise linear function.

To summarize, in the literature on scheduling with storage resources, the selection of different sources are generally ignored and period-dependent piecewise linear costs are also absent. In the energy management literature, standard mixed-integer linear programming approaches are generally used and the subproblem of scheduling the energy consuming tasks is either ignored or highly simplified. With regard to the state-of-the-art, the contributions of this paper are the following: (i) we present a compact formulation, two decomposition schemes and a local search to solve the scheduling problem with energy sources, (ii) the first decomposition scheme results into a matheuristic, (iii) the second decomposition scheme results into an extended formulation that we show to theoretically dominate the compact formulation in terms of linear relaxation and we use column generation and branch-and-price to compute lower bounds, (iv) we evaluate the different algorithms for solving the various subproblems on random and adapted instances from the literature. The results show that the proposed matheuristic significantly outperforms a standard MILP solver on the gap, cpu time and number of feasible solutions found criteria, while the branch-and-price scheme is competitive with the standard MILP solver to obtain high quality lower bounds.

The remainder of the paper is organized as follows. In Section 2, the problem is formally defined and a mixed integer linear program is provided. Section 3 presents the matheuristic developed to compute upper bounds. Section 4 presents an extended formulation and resulting column generation algorithm to compute lower bounds. Section 5 presents the computational results before the Conclusion.

2 Problem definition and compact mixed integer linear programming formulation

This paper considers the problem of scheduling a set of time- and energy-constrained preemptive tasks $\mathcal{J} = \{1, \ldots, n\}$ on a discrete time horizon $\mathcal{T} = \{1, \ldots, T\}$. Each task $i \in \mathcal{J}$ has a duration $p_i \in \{1, \ldots, T\}$, a release date $r_i \in \mathcal{T}$ and a due date $d_i \in \mathcal{T}$. The release date denotes the first period at which the task is available for processing and the due date $d_i \geqslant r_i$ denotes the last one. Between these periods the task must be in process during p_i non necessarily consecutive periods. At each of its active periods, a task i needs to consume $b_i \geqslant 0$ units of an energetic resource. Summing up all energy demands of tasks in process at a given period gives a total energy demand for the period, which can be provided by two energy sources: a reversible one and a non-reversible one. The non-reversible energy source can provide an unlimited amount of energy for a given period but at the expense of a period-dependent non decreasing piecewise linear cost $f_t(B)$ defined on $[0, +\infty[$ with a set $\mathcal{K}_t = \{1, \ldots, K_t\}$ of breakpoints, where $t \in \mathcal{T}$ is the considered period and B is the total energy demand. Such a resource typically models the grid and the possibly time varying piecewise-linear electricity costs. A

breakpoint $k \in \mathcal{K}_t$ is defined by a consumption b_{kt} and a cost c_{kt}, as illustrated on Figure 2 on page 12. The reversible energy source is a storage resource, such as a battery, and is defined by a limited capacity $Q > 0$, an initial level $s_0 \geqslant 0$ and a target final level s^*. The goal is to schedule each task preemptively inside its time window and to dispatch the required energy to the sources at each time period, while satisfying the reversible source capacity constraints and minimizing the total cost. Let (P) denote this problem which can be formulated by a mixed integer linear program, introducing the following decision variables. A period-indexed variable $x_{it} \in \{0, 1\}$ indicates if task $i \in \mathcal{J}$ is in process at time $t \in \mathcal{T}$. A continuous variable $w_t \geqslant 0$ gives the energy amount supplied by the non-reversible source at period $t \in \mathcal{T}$. A continuous variable $s_t \geqslant 0$ gives the remaining amount of resource in the reversible source at the beginning of $t \in \mathcal{T}$. The compact mixed integer programming formulation (CF) of problem (P) can be stated as follows.

$$
\begin{aligned}
\text{(CF)} \quad & \min \sum_{t \in \mathcal{T}} f_t(w_t) && && (1)\\
& \sum_{t \in \mathcal{T}} x_{it} \geqslant p_i, && \forall i \in \mathcal{J} && (2)\\
& w_t - \sum_{i \in \mathcal{J}} b_i x_{it} + s_{t-1} - s_t = 0, && \forall t \in \mathcal{T} && (3)\\
& s_t \leqslant Q, && \forall t \in \mathcal{T} && (4)\\
& s_T = s^* && && (5)\\
& \sum_{i \in \mathcal{J}} \sum_{t \in \mathcal{T} \setminus \{r_i, \ldots, d_i - 1\}} x_{it} = 0 && && (6)\\
& x_{it} \in \{0, 1\}, && \forall i \in \mathcal{J}, \forall t \in \mathcal{T} && (7)\\
& s_t \geqslant 0, && \forall t \in \mathcal{T} && (8)\\
& w_t \geqslant 0, && \forall t \in \mathcal{T} && (9)
\end{aligned}
$$

The objective-function (1) is a sum of piecewise linear functions that gives the total cost of the non-reversible source usage. Constraints (2) enforce each task $i \in \mathcal{J}$ to be in process at least during p_i time periods. Note that equality constraints are not necessary because the cost functions are non-decreasing. Constraints (3) are the inventory balance constraints for the reversible source: the total tasks energy consumption is either covered by the energy supplied by the non-reversible source or the reversible source or both. Note that for period $t = 1$, the initial amount of stored energy s_0 is taken into account. Constraints (4) prevent the reversible source capacity from being exceeded. Constraints (5) strictly enforce to reach the final energy level target in the reversible source. Constraint (6) prevents tasks from being scheduled outside their time window. Constraints (7–9) define the variable domains.

(CF) can be defined as a compact mixed-integer linear programming formulation if the number of time periods T is part of the input and if the piecewise-linear function is modeled through SOS2 constraints for example. SOS2 (special ordered sets of type 2) constraints are defined on an ordered set of positive variables. An SOS2 constraint imposes that the sum of the values assigned to the variables of the set is equal to one, but only two consecutive variables can be non-zero [3]. To achieve the piecewise-linear modeling, we first define the continuous variable $m_{kt} \geqslant 0$, for each time period $t \in \mathcal{T}$ and each breakpoint $k \in \mathcal{K}_t$ such that the list of variables m_{kt} for a fixed t forms a special ordered set of type 2 (SOS2). This means that only two consecutive variables m_{kt} and $m_{k+1,t}$ may have non zero values. This way, combined with $\sum_{k \in \mathcal{K}_t} m_{kt} = 1$, the two unique breakpoints for which $m_{k,t} + m_{k+1,t} = 1$ define the total energy consumption at time period t as $m_{kt} b_{kt} + m_{k+1,t} b_{k+1,t}$ and the corresponding cost is equal to $m_{kt} c_{kt} + m_{k+1,t} c_{k+1,t}$. The resulting formulation is:

$$\text{(CFsos2)} \quad \min \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} c_{kt} m_{kt} \tag{10}$$

$$(2) - (9) \tag{11}$$

$$\text{SOS2}(m_{kt})_{k\in\mathcal{K}_t}, \qquad \forall t \in \mathcal{T} \tag{12}$$

$$\sum_{k\in\mathcal{K}_t} m_{kt} = 1, \qquad \forall t \in \mathcal{T} \tag{13}$$

$$w_t - \sum_{k\in\mathcal{K}_t} b_{kt} m_{kt} = 0, \qquad \forall t \in \mathcal{T} \tag{14}$$

$$m_{kt} \geqslant 0, \qquad \forall t \in \mathcal{T}, \forall k \in \mathcal{K}_t \tag{15}$$

In [17], the same problem without the reversible source, obtained here by setting $s_0 = s^* = Q = 0$, was considered and proven to be strongly NP-hard. So the problem considered in this paper is also strongly NP-hard. The following theorem shows that the case with multiple reversible sources and multiple non-reversible sources can be reduced to the single reversible and single non-reversible source, which enlarges the scope of the proposed models to applications with several storage resources (e.g. batteries with different capacities) and several non-reversible sources (e.g. different grid providers, photovoltaïc sources, etc.).

Theorem 1 *For any problem (P) with $N \geqslant 1$ non-reversible energy sources and $R \geqslant 1$ multiple reversible energy sources, there is an equivalent problem with a single non-reversible energy source and a single reversible energy source*

Proof: The proof is based on the fact that (P) with $N > 1, R > 1$ can be formulated as (*NR*-CF), then be reformulated as (CF).

$$(NR - \text{CF}) \quad \min \sum_{t\in\mathcal{T}} \sum_{n=1}^{N} f_t^n(w_t^n) \tag{16}$$

$$\sum_{t\in\mathcal{T}} x_{it} \geqslant p_i, \qquad \forall i \in \mathcal{J} \tag{17}$$

$$\sum_{n=1}^{N} w_t^n - \sum_{t\in\mathcal{T}} b_i x_{it} + \sum_{r=1}^{R} s_{t-1}^r - \sum_{r=1}^{R} s_t^r = 0, \qquad \forall t \in \mathcal{T} \tag{18}$$

$$s_t^r \leqslant Q^r, \qquad \forall t \in \mathcal{T}, \forall r \in \{1..R\} \tag{19}$$

$$s_T^r = s_0^r, \qquad \forall r \in \{1..R\} \tag{20}$$

$$\sum_{i\in\mathcal{J}} \sum_{t\in\mathcal{T}\setminus\{r_i,\ldots,d_i-1\}} x_{it} = 0 \tag{21}$$

$$x_{it} \in \{0,1\}, \qquad \forall i \in \mathcal{J}, \forall t \in \mathcal{T} \tag{22}$$

$$s_t^r \geqslant 0, \qquad \forall t \in \mathcal{T}, \forall r \in \{1..R\} \tag{23}$$

$$w_t^n \geqslant 0, \qquad \forall t \in \mathcal{T}, \forall n \in \{1..N\} \tag{24}$$

The reformulation from (*NR-CF*) to (CF) can be done, by setting $s_t = \sum_{r\in R} s_t^r, Q = \sum_{r\in R} Q^n$, then $\forall x \geqslant 0$, $f_t(x)$ is defined as the solution cost of the problem:

$$(\text{Cost}_x) \quad \min \sum_{n=1}^{N} f_t^n(w_t^n) \tag{25}$$

$$\sum_{n=1}^{N} w_t^n = x \tag{26}$$

$$w_t^n \geqslant 0, \qquad \forall n \in \{1..N\} \tag{27}$$

□

3 A decomposition matheuristic

3.1 General principle

As explained in the literature review, we have at our disposal two reasonably efficient methods: on the one hand the branch-and-price method proposed in [17] to solve the scheduling subproblem without reversible sources and on the other hand the dynamic programming algorithm proposed by [1] for solving the energy dispatch/lot-sizing subproblem with a fixed schedule. A natural heuristic decomposition of problem (P) is to alternatively solve the scheduling subproblem and the energy dispatch subproblem, which yields the proposed iterative matheuristic.

At step I of the method, we solve the scheduling subproblem, obtained after removing the reversible source from (P). The scheduling subproblem can be modeled with the formulation (CSF) below, where f'_t is the piecewise linear cost function for period $t \in \mathcal{T}$, initially set equal to $f_t(w_t)$. The different methods for solving (CSF) will be described in Section 3.2. Let $x^{(\text{CSF})}, w^{(\text{CSF})}$ be the solution obtained at the end of step I.

$$(\text{CSF}) \quad \min \sum_{t \in \mathcal{T}} f'_t(w_t) \tag{28}$$

$$\sum_{t \in \mathcal{T}} x_{it} \geqslant p_i, \qquad \forall i \in \mathcal{J} \tag{29}$$

$$w_t - \sum_{i \in \mathcal{J}} b_i x_{it} = 0, \qquad \forall t \in \mathcal{T} \tag{30}$$

$$x_{it} = 0, \qquad \forall i \in \mathcal{J}, \forall t \in \mathcal{T} \backslash \{r_i, \ldots, d_i - 1\} \tag{31}$$

$$x_{it} \in \{0, 1\}, \qquad \forall i \in \mathcal{J}, \forall t \in \mathcal{T} \tag{32}$$

$$w_t \geqslant 0 \qquad \forall t \in \mathcal{T} \tag{33}$$

At step II, for each period $t \in \mathcal{T}$, a demand δ_t is fixed to the value $\sum_{i \in \mathcal{J}} b_i x_{it}^{(\text{CSF})}$. The resulting energy dispatch subproblem, described by formulation (CEF) below is solved to adjust the storage resource usage to the previously computed schedule. This problem is a lot-sizing problem with inventory bounds and a piecewise linear production cost function. The inventory costs are not considered. The different methods for solving (CEF) will be described in Section 3.3.

$$\text{(CEF)} \quad \min \sum_{t \in \mathcal{T}} f_t(w_t) \tag{34}$$

$$w_t + s_{t-1} - s_t = \delta_t, \qquad \forall t \in \mathcal{T} \tag{35}$$

$$s_t \leqslant Q, \qquad \forall t \in \mathcal{T} \tag{36}$$

$$s_T = s_0 \tag{37}$$

$$s_t \geqslant 0, \qquad \forall t \in \mathcal{T} \tag{38}$$

$$w_t \geqslant 0, \qquad \forall t \in \mathcal{T} \tag{39}$$

Step III of the method consists in integrating the reversible source usage decisions in the scheduling subproblem. Thanks to the structure of the piecewise-linear cost function, this can be achieved without altering the combinatorial structure of the scheduling subproblem (CSF), simply by updating the piecewise-linear cost functions f'_t for each $t \in \mathcal{T}$. This will be explained in Section 3.4.

After step III the matheuristic returns to step I. The algorithm stops when no cost variation is observed at the end of step II in comparison to step I, or if a predefined maximum number of iterations is reached.

3.2 Step I: solving the scheduling subproblem

As stated in [17], the scheduling subproblem is NP hard. We propose three approaches to solve this subproblem. The first one is to solve directly formulation (CSF) with a general-purpose MILP solver. We will use in this case the IBM CPLEX solver and its build-in piecewise linear constraints ILOPWL. The second approach is to use the branch-and-price procedure proposed in [17], which was shown to have better results than solving (CSF) using a MILP solver on a set of randomly generated problem instances. The only careful adaptation to consider, is that in [17] $f_t(0) = 0$ whereas in our case it is possible to have $f'_t(0) > 0$, due to possible backward shifting described in Section 3.4 and illustrated on Figure 1. Because the scheduling subproblem has to be solved multiple times during the matheuristic, we also design a two-phase heuristic which constitutes the third approach, described hereafter.

The first phase of the heuristic is a constructive phase. Tasks are first sorted randomly in a list and the schedule is empty. Then, for each task i taken in the list order, each of its p_i units is scheduled at the time period not already occupied by an occurrence of the task and that leads to a minimal cost increase. A feasible solution is obtained once all tasks have been scheduled. The process is repeated *nbs* number of times to obtain *nbs* feasible solutions.

The second phase of the heuristic is based on a tabu search algorithm which moves from one solution to another by selecting a task $i \in \mathcal{J}$ and changing the time periods to which k of its units are assigned. At each tabu search iteration the best of all potential moves is applied. It can be computed efficiently as described hereafter.

Let $\widetilde{x}_{i,t}$ be a feasible solution of (CSF), let $\mathcal{T}_i = \{r_i, ..., d_i - 1\}$ be the time window of task i, let $D_i \subset \mathcal{T}_i$ be the subset of $\mathcal{T}_i$ where a unit of task i is in process (i.e $D_i = \{t \in \mathcal{T}_i : \widetilde{x}_{i,t} = 1\}$), let $A_i = \mathcal{T}_i \backslash D_i$ and let $\mathcal{J}_t = \{i \in \mathcal{J} : t \in \mathcal{T}_i\}$ be the subset of tasks with a time window that includes time period t. We use equation (40) to compute the cost variation $\delta_{i,t}$ induced by the removal or the addition of a unit of task i at time period t. We denote $\mathcal{A}_i$ and $\mathcal{D}_i$ the sets obtained after sorting A_i and D_i in decreasing order of $\delta_{i,t}$. Finally we denote $\mathcal{A}_i(u)$ and $\mathcal{D}_i(u)$ the u^{th} elements of $\mathcal{A}_i$ and $\mathcal{D}_i$. The relocation of a unit of task i from time period $D_i(u)$ to time period $A_i(u)$ yields a solution cost variation of $\Delta_i(u) = \delta_{i,\mathcal{D}_i(u)} - \delta_{i,\mathcal{A}_i(u)}$. Function Δ_i is a decreasing function of $u \geqslant 1$ and $\Delta_i(0) = 0$. Given a task i, the best number of units to relocate is $k^* = \max\{1, \max_{\Delta_i(u) \geqslant 0} u\}$, to obtain a cost variation of $\Gamma(i) = \sum_{u=1}^{k^*} \Delta_i(u)$. Therefore each tabu search move consists in identifying the task $i^* = \arg\max_{i \in \mathcal{J}} \Gamma(i)$, then relocating its k^* units scheduled at the time periods $\mathcal{D}_i(1), ..., \mathcal{D}_i(k^*)$ to the time periods $\mathcal{A}_i(1), ..., \mathcal{A}_i(k^*)$. After each move, the update of the sorted sets impacted by the

move and the identification of the task i^* for the next move can be done with a worst case complexity of $O(\overline{\mathcal{J}_t}\ \overline{K}\ \log \overline{K} + n\overline{K})$ where $\overline{K} = \max_{i\in\mathcal{J}}\{p_i, d_i - r_i - p_i\}$ and $\overline{\mathcal{J}_t} = \max_{t\in\mathcal{T}} |\mathcal{J}_t|$.

$$\delta_{i,t} = \begin{cases} f_t(\sum_{j\in\mathcal{J}_t:\tilde{x}_{j,t}=1} b_j) - f_t(\sum_{j\in\mathcal{J}_t\setminus\{i\}:\tilde{x}_{j,t}=1} b_j) & \text{if } t \in D_i \\ f_t(\sum_{j\in\mathcal{J}_t:\tilde{x}_{j,t}=1} b_j) - f_t(b_i + \sum_{j\in\mathcal{J}_t:\tilde{x}_{j,t}=1} b_j) & \text{if } t \in A_i \end{cases} \tag{40}$$

A tabu list mechanism has also been implemented. After each move, the task involved is added at the end of the tabu list. If the size limit *tls* of the tabu list is exceeded after adding a task, then the task at the first position is removed from the list. At each iteration, moves that involve tasks that are in the tabu list are forbidden, except for those that lead to a new best known solution (aspiration criterion). The tabu list mechanism allows the algorithm to escape from local optima. The list is emptied when a move improves the current solution, as it is assumed that the algorithm is entering a new region of the solution space. The stopping criterion of the algorithm is a predefined number of moves without improvement of the best known solution.

The second phase of the heuristic is applied on each of the *nbs* initial feasible solutions before returning the best solution found.

3.3 Step II: solving the energy dispatch subproblem

In order to solve the energy dispatch subproblem, we propose two approaches. The first one consists in solving the MILP model (CEF) using a general-purpose MILP solver. This is performed by using the built-in piecewise linear functions (ILOPWL) of the IBM CPLEX solver. The second approach consists in solving the energy dispatch subproblem using a pseudo polynomial dynamic programming algorithm proposed in [1]. The authors show that the energy dispatch subproblem is weakly NP-Hard. They also show that an optimal solution is integer if we assume that the parameters are integers.

In the following, we provide the general idea behind this dynamic programming algorithm. The rationale of the algorithm relies on the same arguments as the one developed by [22] to solve the capacitated single-item lot-sizing problem. In [22], capacity constraints are imposed on the production while for our problem, they limit quantities in the inventory.

We define $F_k(s)$ as the optimal value of the energy dispatch subproblem limited to periods $k,\ldots,T$ when the remaining reversible resource at the beginning of period k is equal to s ($s_k = s$). The initial values $F_T(s)$ of the recursion function for $s \in \{0,\ldots,Q\}$ are given by:

$$F_T(s) = \begin{cases} f_T(s^T + \delta_T - s) & \text{if } 0 \leqslant s \leqslant \delta_T + s^T \\ +\infty & \text{otherwise} \end{cases}$$

The recursion formula is given by:

$$F_t(s) = \min_{0\leqslant s-\delta_t+w_t\leqslant Q; w_t\geqslant 0} \{F_{t+1}(s - \delta_t + w_t) + f_t(w_t)\} \tag{41}$$

The optimal solution of the energy dispatch subproblem is given by $F_1(s_0)$. The main limits of this dynamic programming algorithm are: (i) integer demand values, (ii) continuous time-dependent piecewise linear functions with integer abscissa of breakpoints, and (iii) reversible source with integer capacity values. In this case, the complexity of the resulting dynamic programming algorithm is $O(T^2\overline{\delta}\overline{q})$ where $\overline{\delta}$ is the average demand and $\overline{q}$ is the average number of breakpoints.

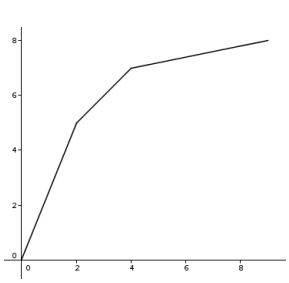
3.4 Step III: piecewise linear cost function update

After solving the energy dispatch subproblem from the previous step, the energy level variation in the reversible source is equal to $s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})}$ for each time period $t \in \mathcal{T}$. If the solution cost observed after step II differs from the one observed after step I, then each piecewise linear function f'_t is updated with equation (42) which amounts to consider a fixed reversible source usage.

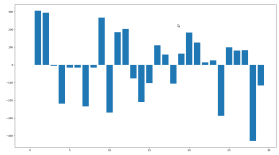
$$f'_t(w_t) = f_t(\max(0, w_t + s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})})), \quad \forall t \in \mathcal{T} \tag{42}$$

In other words, if $s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})} > 0$, the piecewise linear cost function is left-shifted by $s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})}$ energy units, to represent the fact that even if no task is scheduled by (CEF) at time t, there will still be a demand of $s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})} > 0$ to send to the reversible source. On the other hand, if $s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})} < 0$, the function is right shifted by $s_{t-1}^{(\text{CEF})} - s_t^{(\text{CEF})}$ energy units, negative amounts being ignored, to represent the fact that if the tasks scheduled have a total demand lower or equal to $s_{t-1}^{(\text{CEF})} - s_t^{(\text{CEF})}$, then the resulting cost on the non-reversible source will be nil. Figure 1 illustrates, for a function f and a period t, a left-shift when $s_{t-1}^{(\text{CEF})} - s_t^{(\text{CEF})} = 2$, and right-shift when $s_{t-1}^{(\text{CEF})} - s_t^{(\text{CEF})} = -2$.

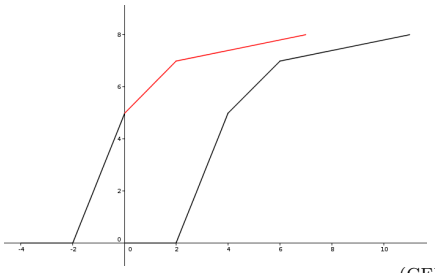
The matheuristic stops if no cost variation is observed between step I and step II, or if the maximum number of iterations *maxit* is reached. Otherwise, the matheuristic returns to step I after step III.



Original $f_t(x)$ for a period t



Variations of $s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})}$



$f'_t(x)$ after adjustment on a period t for $s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})} = -2$ and $s_t^{(\text{CEF})} - s_{t-1}^{(\text{CEF})} = 2$

Figure 1: Updating the piecewise linear cost function f'_t

4 An extended formulation and a column-generation procedure

To assess the quality of the obtained feasible solutions, a lower bounding scheme is needed. Apart from the bounds that can be derived from the compact formulations (CF) or (CFsos2), we propose a new extended formulation for the problem that allows to obtain a lower bound by column generation.

4.1 Extended formulation

The extended formulation is based on the one proposed in [17] for the scheduling subproblem (CSF), itself being issued from the formulation by [15] for the resource-constrained project scheduling problem. We first present the principle of this decomposition before showing how it can be extended to the presence of the reversible sources. The extended formulation presented in [17] is based on binary variable y_{lt} representing the simultaneous processing of a set of activities l at a given time period t. Such a set is called a feasible set. Let $\mathcal{J}_t \subseteq \mathcal{J}$ be the set of all tasks that can be processed at time period t due to the time windows restriction. There are $2^{|\mathcal{J}_t|}$ feasible sets of tasks that can be processed simultaneously at time period t. Let $\mathcal{F}_t$ be the set of all such feasible sets. Let a_{il} be the indicator of the presence of task i in the feasible set l. The energy consumption of set l is $\sum_{i \in \mathcal{J}} a_{il} b_i$. Hence if $y_{lt} = 1$, the cost to be paid at period t is a constant $c_l = f_t(\sum_{i \in \mathcal{J}} a_{il} b_i)$. It follows that the total cost of a solution of [17] described by variables y_{lt} can be written as the linear expression $\sum_{t \in \mathcal{T}} \sum_{l \in \mathcal{F}_t} c_l y_{lt}$. However, in our (CF) problem, knowing which set of tasks is in process at a given

time period t is not sufficient to derive the cost to be paid, due to the presence of the reversible source. We show hereafter how the feasible set model can be extended by adding a breakpoint index to the information carried by the y variables.

A set of tasks $\mathcal{J}_l \neq \varnothing$ is called a feasible set if $\forall i \in \mathcal{J}_l$, $t \in [r_i, d_i]$. Let $\mathcal{F}_t = \{0, 1, \ldots, |\mathcal{F}_t| - 1\}$ denote the set of feasible set of indices for time period t with 0 being the index of the empty set. For a given time period t, we consider the pairs made of a feasible set and a breakpoint. To each pair $(l \in \mathcal{F}_t, k \in \mathcal{K}_t)$ is assigned a release date $r_l = \max_{i \in \mathcal{J}_l} r_i$, a due date $d_l = \min_{i \in \mathcal{J}_l} d_i$, an energy demand $b_l = \sum_{i \in \mathcal{J}_l} b_i$ from tasks of $\mathcal{J}_l$, an energy amount $b_{kt} - b_l$ added to (resp. produced by) the reversible source if $b_{kt} - b_l \geqslant 0$ (resp. if $b_{kt} - b_l \leqslant 0$), and an energy cost c_k on the non-reversible source. Because f_t are non decreasing functions, and because the amount of energy added to or produced by the reversible source is limited to capacity Q, then the set of breakpoints incompatible with the tasks set l is composed of the breakpoints k that verify: $b_{k-1} - b_l > Q$ or $b_{k+1} - b_l < -Q$.

We now define the continuous variables y_{lkt}, $\forall t \in \mathcal{T}$, $\forall k \in \mathcal{K}_t$, $\forall l \in \mathcal{F}_{kt}$ as the values of m_{kt} if feasible set l is in process at time period t. The proposed extended formulation is as follows:

$$\text{(EFsos2)} \quad \min \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}_t} c_{kt} m_{kt} \tag{43}$$

$$(2), (4)-(5), (7)-(8), (12)-(14) \tag{44}$$

$$m_{kt} - \sum_{l \in \mathcal{F}_t} y_{lkt} = 0, \qquad \forall t \in \mathcal{T}, k \in \mathcal{K}_t \tag{45}$$

$$s_t - s_{t-1} - \sum_{k \in \mathcal{K}_t} b_{kt} m_{kt} + \sum_{i \in \mathcal{J}} b_i x_{it} = 0, \qquad \forall t \in \mathcal{T} \tag{46}$$

$$x_{it} - \sum_{l \in \mathcal{F}_t} \sum_{k \in \mathcal{K}_t} a_{il} y_{lkt} = 0, \qquad \forall i \in \mathcal{J}, \forall t \in \mathcal{T} \tag{47}$$

$$y_{lkt} \geqslant 0, \qquad \forall t \in \mathcal{T}, \forall k \in \mathcal{K}_t, \forall l \in \mathcal{F}_{kt} \tag{48}$$

Objective-function (43) gives the total cost computed with SOS2 variables m_{kt}. Constraints (2), (4)-(5), (7)-(8), (12)-(14) are the same as in the compact formulation. Constraints (45) link the m_{kt} variables with the y_{lkt} variables in such a way that, since m_{kt} variables are SOS2, we will have $\sum_{k \in \mathcal{K}_t} \sum_{l \in \mathcal{F}_{kt}} y_{lkt} = 1$ and there is also a unique $k \in \mathcal{K}_t$ such that $\sum_{l \in \mathcal{F}_{kt}} y_{lkt} + \sum_{l \in \mathcal{F}_{kt}} y_{l,k+1,t} > 0$. Constraints (46) are flow conservation constraints. Constraints (47) link the x and y variables. Finally, variables y_{lkt} must be non negative.

Lemma 1 *An optimal solution (x, y, s, m) of MILP (EFsos2) provides an optimal solution for problem (P).*

Proof: Proving that an optimal solution (x, y, s, m) of MILP (EFsos2) verifying $y \in \{0, 1\}^{|\mathcal{L}||\mathcal{K}||\mathcal{T}|}$ is also an optimal solution for problem (P) is straightforward. It remains to be shown, however, that the cases where $0 < y_{lkt} < 1$ for some $l \in \mathcal{L}$, $k \in \mathcal{K}$ and $t \in \mathcal{T}$ also correspond to valid and thus optimal solutions for the original problem. This is achieved by proving that although there are decision variables y_{lkt} assigned to continuous values, the model ensures that at each time period t only one activity set l is chosen, combined with either one linear segment or one breakpoint of the piecewise linear function f_t. It is an extension of the proof provided in [17]. Let $\widetilde{S}$ be a feasible solution of MILP (EFsos2): $\widetilde{x}_{it} \in \{0, 1\}^{|\mathcal{J}||\mathcal{T}|}$, $\widetilde{y}_{lkt} \in \mathbb{R}^{|\mathcal{L}||\mathcal{K}||\mathcal{T}|}$, $\widetilde{s}_t \in \mathbb{R}^{|\mathcal{T}|}$ and $\widetilde{m}_{kt} \in \mathbb{R}^{|\mathcal{K}||\mathcal{T}|}$. Given a time period $t^* \in \mathcal{T}$, let us denote:

- $\mathcal{LK}^{>0}$ the pairs "activity set,breakpoint" used at instant t^*. In other words $\mathcal{LK}^{>0} = \{l \in \mathcal{L}, k \in \mathcal{K} : \widetilde{y}_{lkt^*} > 0\}$.
- $J^{\mathcal{LK}^{>0}}$ the subset of activities that appear in at least one set l of $\mathcal{LK}^{>0}$. In other words $J^{\mathcal{LK}^{>0}} = \{i : \exists (l, k) \in \mathcal{LK}^{>0}$ that verifies $a_{il} = 1\}$.

- $\mathcal{LK}^{>0}(i)$ the subset of pairs "activity set,breakpoint" from $\mathcal{LK}^{>0}$ that contain activity i.

Since $\widetilde{y}_{lkt*} > 0, \forall (l,k) \in \mathcal{LK}^{>0}$ (by definition) and $\sum_{(l,k)\in\mathcal{LK}^{>0}} \widetilde{y}_{lkt*} = 1$ (from constraints (13) and (45)), we deduce Proposition 2 which can be proven valid because by definition $\forall i \in J^{\mathcal{LK}^{>0}}, \widetilde{x}_{it*} = 1$ and therefore constraints (47) imply that:

$$\sum_{(l,k)\in\mathcal{LK}^{>0}(i)} \widetilde{y}_{lkt*} = 1, \qquad \text{where } \mathcal{LK}^{>0}(i) \subseteq \mathcal{LK}^{>0}. \tag{49}$$

Proposition 2 *$\forall \widetilde{\mathcal{LK}} \subseteq \mathcal{LK}^{>0}$, if $\sum_{(l,k)\in\widetilde{\mathcal{LK}}} \widetilde{y}_{lkt*} = 1$ then $\widetilde{\mathcal{LK}} = \mathcal{LK}^{>0}$.*

Finally, combining Proposition 2 and Constraint (49) we can deduce:

$$\mathcal{LK}^{>0}(i) = \mathcal{LK}^{>0}, \forall i \in J^{\mathcal{L}^{>0}}. \tag{50}$$

Let $\mathcal{L}^{>0}$ be the set of activity sets that verify $\exists k : (l,k) \in \mathcal{LK}^{>0}$, and let $\mathcal{K}^{>0}$ be the set of breakpoints that verify $\exists l : (l,k) \in \mathcal{LK}^{>0}$. Constraints (50) imply that $|\mathcal{L}^{>0}| = 1$, and therefore only one activity set is chosen at each time period t^*. Constraints (12), (13) and (45) imply that either one breakpoint or two consecutive breakpoints are chosen, with a total weight of 1 at time period t^*.

Thus, solving (EFsos2) produces optimal solutions for the original problem (P). □

Lemma 2 *The extended formulation (EFsos2) dominates the compact formulation (CFsos2), i.e. the linear relaxation of (EFsos2) is stronger than the linear relaxation of (CFsos2).*

Proof: To prove that the linear relaxation of a formulation (A) is stronger than the linear relaxation of formulation (B), we can first prove that the linear relaxation of (A) at least as good as the linear relaxation of (B) by showing that any solution of the linear relaxation of (A) is also a solution of the linear relaxation of (B). Then we have to find an example for which the optimal solution of the relaxation of (A) has a strictly better objective function value than the optimal solution of the relaxation of (B).

Any solution of the linear relaxation of (EF) can be converted into a solution of the linear relaxation of (CF) by setting $x_{it}^{(\text{CFsos2})} = x_{it}^{(\text{EFsos2})}$, $w_t^{(\text{CFsos2})} = \sum_{k\in\mathcal{K}} b_k m_{kt}^{(\text{EFsos2})}$ and $s_t^{(\text{CFsos2})} = s_t^{(\text{EFsos2})}$. Therefore the linear relaxation of (EFsos2) is at least as strong as the linear relaxation of (CFsos2). It remains to be proven that there exist solutions of (EFsos2) that are not feasible for (CFsos2), in order to prove that the two linear relaxations are not just equivalent. To do so, we exhibit hereafter such a case, with $|\mathcal{J}| = 1, |\mathcal{T}| = 1, b_1 = 4, r_1 = 1, d_1 = 2, q_{\max} = 1.5, q_{\text{init}} = 0$ and the piecewise linear function f_1 illustrated on Figure 2. In this case, for the extended formulation (EFsos2), only one non-empty feasible set of tasks $l = \{1\}$ exists, therefore $\mathcal{L} = \{\varnothing, \{1\}\}$. Each column y_{lkt} corresponds to a pair "set of tasks l, breakpoint k" that verify $b_k - b_l \geqslant q_{\max}$ and $b_l - b_{k+1} \leqslant q_{\max}$. Therefore only four columns exist in (EFsos2): "$l = 0, k = 1$", "$l = 0, k = 2$", "$l = 1, k = 2$" and "$l = 1, k = 3$". The resulting linear relaxation is: $x_{1,1} = 1, y_{1,2,1} = y_{1,3,1} = m_{2,1} = m_{3,1} = 0.5$ and $y_{0,1,1} = y_{0,2,1} = m_{1,1} = m_{4,1} = s_1 = 0$. Its costs is equal to 9.5. Meanwhile, the linear relaxation of (CFsos2) produces the fractional solution $x_{1,1} = 1, w_1 = 4, m_{1,1} = m_{4,1} = 0.5$ and $m_{2,1} = m_{3,1} = 0$, which has a cost of 7.0, which is lower than the one of (EFsos2).

In conclusion, the linear relaxation of (EFsos2) is stronger than the linear relaxation of (CFsos2). This also proves that the pricing problem associated with formulation (EFsos2) is NP-hard. □

4.2 Column generation procedure

Formulation (EFsos2) has a polynomial number of constraints, a polynomial number of variables x_{it}, m_{kt}, s_t, but an exponential number of variables y_{lkt}. Solving its linear relaxation requires a column generation approach. The linear relaxation of (EFsos2) serves as the master problem (MP)

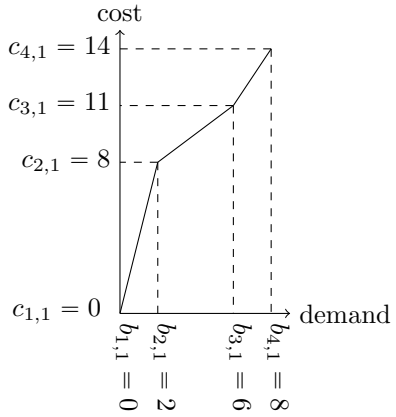


Figure 2: A piecewise linear cost function

of the Dantzig-Wolfe decomposition. Let $\mathcal{Y}$ be the set of variables y_{lkt}. At each iteration p of the column generation procedure we solve a restricted master problem (RMP_p) obtained by restricting $\mathcal{Y}$ to a subset $\widetilde{\mathcal{Y}} \subseteq \mathcal{Y}$. Then we try to identify one or several variables $y_{lkt} \in \mathcal{Y} \backslash \widetilde{\mathcal{Y}}$ having a negative reduced cost with regards to the dual variables values of the optimal solution of (RMP_p). Let α_{kt} denote the dual variable associated with constraints (45) and let β_{it} denote the dual variable associated with constraints (47). The dual constraint for variables y_{kit} for the restricted master problem can be written as

$$\alpha_{kt} + \sum_{i \in \mathcal{J}} a_{il} \beta_{it} \geqslant 0, \quad \forall t \in \mathcal{T}, \forall k \in \mathcal{K}_t, \forall l \in \widetilde{\mathcal{F}}_{kt} \tag{51}$$

The left-hand-side of inequality (51) is the reduced cost of the corresponding variable y_{lkt}. As a consequence, the pricing procedure consists in building a feasible task subset $\widetilde{l}$ and identifying a time period $\widetilde{t}$ and a breakpoint $\widetilde{k}$ that minimize the left-hand-side of inequality (51). If the value found is strictly negative, then the corresponding variable y_{lkt} is introduced into the model. Otherwise it can be disregarded and the current optimal solution of (RMP_p) is the optimal solution of the current linear relaxation of (EFsos2).

Let $u_i \in \{0, 1\}$ be a binary variable such that $u_i = 1$ if and only if task i is selected to compose the set $\widetilde{l}$. For a fixed $t \in \mathcal{F}_{kt}$ and a fixed $k \in \mathcal{K}_t$, searching for set $\widetilde{l}$ amounts to checking whether the 0-1 ILP $(\text{SP})_{k,t}$ below has an optimal value strictly lower than $-\alpha_{kt}$.

$$(\text{SP})_{k,t} \quad \min \sum_{i \in \mathcal{J}} \beta_{it} u_i \tag{52}$$

$$\sum_{i \in \mathcal{J}} b_i u_i \geqslant b_{k-1,t} - Q \tag{53}$$

$$\sum_{i \in \mathcal{J}} b_i u_i \leqslant b_{k+1,t} + Q \tag{54}$$

$$u_i = 0, \qquad \forall i \in \mathcal{J}, \forall t \in \mathcal{T} \backslash \{r_i, \cdots, d_i - 1\} \tag{55}$$

$$u_i \in \{0, 1\}, \qquad \forall i \in \mathcal{J} \tag{56}$$

$(\text{SP})_{k,t}$ is an integrated knapsack and covering problem. The constraints aim at eliminating a feasible set incompatible with breakpoints k and $k+1$ of f_t. Indeed, if $\sum_{i \in \mathcal{J}} b_i u_i + Q < b_{kt}$, there is no possible energy taken from reversible source such that the total consumption falls in the corresponding piece. Similarly if $\sum_{i \in \mathcal{J}} b_i u_i - Q > b_{k+1,t}$, the smallest possible consumption value falls beyond the considered piece for any energy level provided to the reversible source.

We integrate the search for the breakpoint $\widetilde{k}$ in the formulation, by adding variable $v_k \in \{0,1\}$ that takes value 1 if and only 1 piece $k, k+1$ is selected. We obtain subproblem $(\text{SP})_t$ below for each time period t. The pricing procedure therefore consists in solving $(\text{SP})_t$ for each time period $t \in \mathcal{T}$, identifying the solutions with a negative objective-function value, and adding the corresponding variables $y_{\widetilde{l}\widetilde{k}t}$ to the restricted master problem.

$$
\begin{aligned}
(\text{SP})_t \quad \min \quad & \alpha_{kt} v_k + \sum_{i \in \mathcal{J}} \beta_{it} u_i && && (57)\\
& \sum_{i \in \mathcal{J}} b_i u_i \geqslant \sum_{k \in \mathcal{K}_t} b_{k-1,t} v_k - Q && && (58)\\
& \sum_{i \in \mathcal{J}} b_i u_i \leqslant \sum_{k \in \mathcal{K}_t} b_{k+1,t} v_k + Q && && (59)\\
& u_i = 0, && \forall i \in \mathcal{J}, \forall t \in \mathcal{T} \backslash \{r_i, \cdots, d_i - 1\} && (60)\\
& u_i \in \{0,1\}, && \forall i \in \mathcal{J} && (61)\\
& v_k \in \{0,1\}, && \forall k \in \mathcal{K}_t && (62)
\end{aligned}
$$

4.3 Branch-and-Price

Solving (EFsos2) requires a branch-and-price to ensure the integrity of binary variables x_{it} and the satisfaction of sos2 constraints. Two variants of column generation or branch-and-price algorithms can be obtained depending on the column adding policy in the pricing procedure:

- Variant 1: for each time period t, add the activity set $\widetilde{l}$ at breakpoint $\widetilde{k}$ only
- Variant 2: for each time period t, add the activity set $\widetilde{l}$ at all compatible breakpoints

As for the branching strategy, the only binary variables are the x_{it} variables. We let the solver (SCIP) apply its default branching strategy on these variables and on the SOS2 Constraints.

5 Computational Experiments

In this section, we present computational results [2] for random and adapted instances from the literature. The general purpose of these experiments is to answer the following questions

- Is it worthwhile to tackle the problem with the decomposition approach that solves iteratively the scheduling subproblem and the energy dispatch (lot-sizing) subproblem, both in terms of upper bound quality and computational requirements?
- As a complement, can we provide lower bounding schemes to assess the quality of the upper bounds and what is the quality of the bound based on the extended formulation, compared to the one based on the compact formulation?

All experiments were run in a single thread mode on 3 cluster nodes, each with 28 Intel Xeon CPU E5-2695 v4 2.30GHz cores running Linux Ubuntu 16.04.4.

[2] Detailed results including lower bounds, upper bounds and cpu times are available at http://homepages.laas.fr/sungueve/Data/SWRSFullResultsLBUBcpu.csv

5.1 Lot-sizing solution: dynamic programming vs MILP solver

In this section we analyze the benefits and drawbacks of solving the lot-sizing subproblem (CEF) with the pseudo-polynomial time dynamic programming algorithm of [1] or with a general-purpose MILP solver. This analysis aims at selecting the best approach to solve the lot-sizing problem so as to answer properly our first question on the merits of the decomposition approach. Basically, we are looking in turn for the answers to two other questions: (i) which of the two is faster? (ii) given that the application of the dynamic programming requires a preliminary rounding of the input data to the next higher whole number, how far is its solution in comparison to the optimal (CEF) solution? The dynamic programming algorithm was coded in C++. The lot-sizing model (CEF) was solved with IBM CPLEX 12.7.1. We generated two sets of random instances with a maximal demand of 100 and 2000 units, and with the following parameters settings $T \in \{20, 100, 500\}, \overline{q} \in \{4, 10\}, \overline{Q} \in \{100, 1000\}$. Ten random instances were generated for each combination of parameters, leading to a total of 120 instances per set. Tables 1 and 2 present the results obtained with the following headers: computing time in seconds (average, minimum, maximum), upper bounds (=solution costs), lower bound (from the MILP solver), #opt is the number of instances solved to optimality, gap is the average distance between the costs of solutions provided by the dynamic programming in comparison to the optimal ones from the MILP solver when available.

			Dynamic Programming					CPLEX						
T	$\overline{q}$	$\overline{Q}$	Computing time (s)			Bound	#opt	Computing time (s)			Bound		#opt	gap
			avg	min	max	upper	(/10)	avg	min	max	upper	lower	(/10)	(%)
20	4	100	**0.0003**	0.0002	0.0005	2502.13	10	0.1224	0.0019	0.2710	2483.14	2483.01	10	0.8
20	10	100	**0.0007**	0.0006	0.0010	2788.15	10	1.6372	0.0232	8.8190	2766.15	2765.94	10	0.8
20	4	1000	**0.0019**	0.0015	0.0028	2159.08	10	0.2177	0.0015	1.4725	2141.12	2141.1	10	0.9
20	10	1000	**0.0039**	0.0037	0.0047	2445.73	10	5.8101	0.0016	56.6005	2426.26	2426.19	10	0.8
100	4	100	**0.0011**	0.0010	0.0019	9730.84	10	271.8140	12.1239	305.8810	9662.17	9396.98	1	0.6
100	10	100	**0.0048**	0.0025	0.0237	11478.90	10	240.3810	0.2242	300.0970	11366.9	11195.7	2	1.0
100	4	1000	**0.0127**	0.0073	0.0562	7728.18	10	78.7309	0.0578	300.0000	7651.76	7649.19	8	1.0
100	10	1000	**0.0202**	0.0193	0.0215	10198.00	10	151.7000	0.0275	300.0200	10086.2	10076.4	5	1.1
500	4	100	**0.0071**	0.0060	0.0114	61676.30	10	300.0120	300.0000	300.0520	61501.4	60551.3	0	-
500	10	100	**0.0140**	0.0136	0.0145	56813.10	10	300.0060	300.0000	300.0230	56632.2	55604.1	0	-
500	4	1000	**0.0426**	0.0379	0.0560	58571.80	10	210.0460	0.0986	300.0150	58057.9	57986	3	0.9
500	10	1000	**0.1072**	0.0932	0.1997	52692.70	10	270.0890	0.8423	300.0300	52193.8	52087.5	1	0.9
		Total					120						60	

Table 1: Comparison between dynamic programming and CPLEX (maximal demand = 100)

			Dynamic Programming					CPLEX						
T	$\overline{q}$	$\overline{Q}$	Computing time (in seconds)			Bound	#opt	Computing time (in seconds)			Bound		#opt	gap
			avg	min	max	upper	(/10)	avg	min	max	upper	lower	(/10)	(%)
20	4	100	**0.0002**	0.0002	0.0003	46346.4	10	0.0026	0.0010	0.0059	46322.6	46322.6	10	0.1
20	10	100	**0.0006**	0.0004	0.0007	51630.6	10	0.0032	0.0018	0.0046	51604.3	51604.3	10	0.1
20	4	1000	**0.0020**	0.0014	0.0024	40991.9	10	0.0406	0.0023	0.3443	40969.9	40969.1	10	0.1
20	10	1000	**0.0032**	0.0027	0.0049	46945.6	10	0.1269	0.0124	0.2820	46920.4	46917.1	10	0.1
100	4	100	**0.0007**	0.0007	0.0008	308410	10	0.0086	0.0042	0.0124	308272	308266	10	0.0
100	10	100	**0.0017**	0.0016	0.0021	274699	10	0.0115	0.0094	0.0138	274566	274564	10	0.0
100	4	1000	**0.0060**	0.0056	0.0065	273788	10	252.9610	3.9349	300.2350	274049	270483	2	0.1
100	10	1000	**0.0164**	0.0129	0.0434	244329	10	300.0100	300.0000	300.1040	244269	242273	0	–
500	4	100	**0.0042**	0.0041	0.0044	1279010	10	0.0471	0.0398	0.0571	1278420	1278410	10	0.0
500	10	100	**0.0086**	0.0083	0.0089	1350720	10	0.0722	0.0600	0.0964	1350090	1350020	10	0.0
500	4	1000	**0.0302**	0.0290	0.0310	1124550	10	240.0110	0.0238	300.0580	1127720	1110030	2	0.3
500	10	1000	**0.0688**	0.0675	0.0723	1231380	10	300.0090	300.0000	300.0350	1234930	1213730	0	–
		Total					120						84	

Table 2: Comparison between dynamic programming and CPLEX (maximal demand = 2000)

Computational results show that the dynamic programming is orders of magnitudes faster than CPLEX, with a computing time in general less than a tenth of a second, whereas CPLEX does not solve to optimality 40% of the instances. Also the gap between optimal dynamic programming solution cost and optimal MILP solutions costs (when available) is in general lower than 1%. In conclusion, even

though input data needs to be rounded up as a preprocessing, the lot-sizing subproblem (CEF) is better solved with the dynamic programming algorithm than with the general-purpose MILP solver. And because of the negligible computing time of dynamic programming, when used in the matheuristic, the computing time of steps II become negligible and all of the computing time available can be assigned to the step I.

5.2 Comparison of the upper bounding procedures

In this section we evaluate the matheuristic in comparison to applying a MILP solver on the formulations (CF) and (CFSOS2). The scheduling instances with $|\mathcal{J}| \in \{30, 60\}$ and the adapted continuous piecewise linear cost function for the non-reversible source are from [17]. We consider a reversible source capacity of $Q = 2$ with $s_0 = s_T = 0$. Among the 288 resulting instances, 181 could be solved to optimality or within a 1% optimality gap by a general purpose MILP solver applied on compact formulation (CF) in 600 seconds or less. The test bench used in this section is composed of the remaining 107 instances. Algorithms were coded in C++ with the framework SCIP 5.0.1. The compact formulation (CF) was solved with IBM CPLEX 12.7.1 using its build-in piecewise linear functions (ILOPWL). The compact formulation (CFSOS2) was solved with IBM CPLEX 12.7.1 and with SCIP 5.0.1. For the column-generation-based or the branch-and-price-based algorithms, each master problem was solved with SOPLEX 3.1.1 whereas each subproblem was solved with IBM CPLEX 12.7.1.

A time limit of 600 s was given to all solvers and algorithms. Different variants of the matheuristic can be obtained, depending on the solution method applied on step I or step II. In the previous section 5.1 it is shown that dynamic programming outperforms the MILP solver for solving (CEF), therefore steps II will only be solved with dynamic programming. Since the resulting computation time for steps II is negligible, then the time limit of 600 s can be assigned directly to solving scheduling subproblems, therefore a time limit of $600s/maxit$ is assigned to each individual scheduling subproblem. The following three main matheuristic variants were applied:

(cpct) Step I solved by applying the MILP solver on (CF) with a time limit and obtaining the best feasible solution

- 3 options: 10x60 s (i.e $maxit = 10$, 60 s time limit per sub-problem); 6x100 s and 3x200 s

(bp) Step I solved by applying the Branch-and-Price from [17]

- 3 options: 10x60 s; 6x100 s and 3x200 s

(h) Step I solved with the tabu search heuristic

- 1 option: 10x60 s, since the heuristic takes less than a second per call
- heuristic parameters used:
  - $nbs = 3$
  - tabu list size: 2 options, short ($tls = 7$) or long ($tls = 14$). The first time the matheuristic calls upon the heuristic, both options are used and the one that produces the best solution is selected and will be the only one applied during the remainder of the calls from the matheuristic.
  - stopping criterion: predefined number of movements without improvement of the best known solution (=200)

Tables 3, 4 and 5 report the computational results of the MILP solver and the matheuristic with the following headings: (i) UB gap: upper bound gap, equal to $100\frac{\text{UB}-\text{UB}^+}{\text{UB}^+}$ where UB is the upper bound obtained and UB^+ is the best known upper bound from any of the algorithms or solvers; (ii) cpu: average computing time; (iii) #it: average number of iterations performed by the matheuristic; (iv) #feas: number of feasible solutions obtained (out of the 107 instances of the benchmark).

5.2.1 Matheuristic vs MILP solver

Table 3 show that the general-purpose MILP solvers are unable to produce feasible solutions on at least 10% of the instances. It also shows that the quality of the instances obtained is significantly improved when using the build-in piecewise linear constraints instead of the classical SOS2. Previous studies have already established that SOS2 modeling can be greatly improved, for example by valid inequalities [10].

		gap	cpu	#feas (/107)
CPLEX	with build-in (ILOPWL) constraints	2.58 %	600 s	96
	with SOS2 constraints	76.37 %	600 s	95
SCIP	with SOS2 constraints	1.98 %	600 s	3

Table 3: Computational results from the MILP solvers applied on (CF) and (CFSOS2)

Table 4 shows that the matheuristic outperforms the general-purpose MILP solver on all three criteria (solution quality, computing time, number of feasible solutions) when the branch-and-price or the heuristic are used to solve step I. The average number of iterations shows that only a few number of iterations are needed to converge. Note that no warmstart mechanism has been implemented in the branch-and-price, which could reduce the computing time further while improving the solutions quality from one iteration to another. This is however the underlying principle of the incremental procedure presented in the following section.

		gap	cpu	#it	#feas (/107)
cpct	10 x 60	2.13 %	512 s	8.53	83
	3 x 200	2.82 %	582 s	2.91	87
	6 x 100	2.31 %	542 s	5.42	84
bp	10 x 60	1.25 %	236 s	4.90	101
	3 x 200	1.03 %	391 s	2.89	106
	6 x 100	0.83 %	350 s	4.68	103
h	10 x 60	2.20 %	4 s	3.51	107

Table 4: Computational results from the matheuristic

5.2.2 Focus on the matheuristic: tabu-search-based incremental step I solution

The previous subsection 5.2.1 showed that the matheuristic outperforms the MILP solver, in particular when branch-and-price is used to solve step I. However, the computing times for solving step I are significant: 350s on average for the best variant. One reason is that each method used for step I (either cpct, bp or h) is relaunched from scratch at each iteration. To reduce the computing times while preserving the solution quality, one idea is to favor incrementality by repairing and improving the solution found after the step III of the previous iteration, instead of relaunching the cpct, bp or h method. The repair and improvement procedure is based on the tabu search procedure previously used inside the heuristic and described in Section 3.2. The resulting tabu-search-based variant of the matheuristic follows the same 3-steps principles described in Section 3.1. Its distinguishing feature resides in the procedure used to compute the solution of (CSF) in step I. Let q be the current iteration of the matheuristic and let $(x^{(\mathrm{CSF})}, s^{(\mathrm{CEF})}, w^{(\mathrm{CEF})})_{q-1}$ be the solution obtained after step III of the iteration $q-1$. The step I of the tabu-search-based matheuristic can be summarized as follows:

if $q = 1$, then

solve (CSF) with the dedicated solution method, i.e MILP solver (cpct-ts), branch-and-price (bp-ts), heuristic (h-ts).

otherwise

1. initialize $\widetilde{x} \leftarrow (x^{(\text{CSF})})_{q-1}$, then compute the new cost of $\widetilde{x}$ using the updated piecewise linear costs functions
2. apply the tabu search on $\widetilde{x}$ and find a new best solution denoted x^*
3. return $x^{(\text{CSF})} \leftarrow x^*$ as the solution of (CSF) obtained at the end of step I

Results are summarized in Table 5, where it can be observed that the best results are obtained with the new variant bp-ts in less than 80s. Remarkably, the "pure" heuristic h-ts (without any call to a general-purpose solver) obtains better results than all standalone MILP solvers on formulations (CF) and (CFsos2) in only 1 s.

		gap	cpu	#it	#feas (/107)
cpct-ts	10 x 60	29.40 %	61 s	3.76	89
	3 x 200	14.80 %	200 s	2.71	90
	6 x 100	25.18 %	101 s	3.53	89
bp-ts	10 x 60	1.82 %	48 s	3.23	103
	3 x 200	1.57 %	131 s	2.60	107
	6 x 100	1.65 %	72 s	2.90	105
h-ts	10 x 60	2.49 %	1 s	2.99	107

Table 5: Computational results from the tabu-search-based matheuristic

5.3 Comparison of the lower bounding procedures: branch-and-price vs MILP solver

In this section we investigate five lower bounding schemes that can be derived from the compact formulations (CF), (CFsos2) and from the extended formulation (EFsos2). The goal is to assess the quality of the upper bounds obtained in Section 5.2 and to identify the best lower bounding scheme among the following approaches:

- CPLEX ilopwl: the compact formulation (CF) is solved with general-purpose solver IBM CPLEX 12.7.1, using its build-in piecewise linear functions modeling constraints (ilopwl)
- CPLEX sos2: the compact formulation (CFsos2) is solved with general-purpose solver IBM CPLEX 12.7.1
- SCIP sos2: the compact formulation (CFsos2) is solved with general-purpose solver SCIP 5.0.1 that uses SOPLEX 3.1.1 as the underlying linear solver.
- Column generation / Branch-and-Price variant 1: the extended formulation (EF) is modeled and solved with the framework SCIP 5.0.1 where every master problem is solved with SOPLEX 3.1.1 and every subproblem is solved with IBM CPLEX 12.7.1. under the variant 1 of the Branch-and-Price procedure explained in Section 4.3
- Column generation / Branch-and-Price variant 2: same as variant 1 with variant 2 of the Branch-and-Price procedure (see Section 4.3).

The 107 instances that constitute the benchmark are grouped into four classes depending on the number of tasks and length of the time horizon. These classes contain respectively 27, 27, 19 and 55 instances. A time limit of 600 s was given to all solvers and algorithms. Tables 6 and 7 report the computational results of the lower bounding schemes with the following headings: (i) LB ratio: average lower bound ratio, equal to $100\frac{\text{LB}}{\text{UB}^+}$ where LB is the lower bound obtained and UB^+ is the best known upper bound from any of the algorithms or solvers from section 5.2.1; (ii) # time out (only for Table 6): number of instances for which the linear relaxation could not be solved to optimality by the LP solvers or the column generation algorithms under the time limit. Note that the average LB ratio is computed over all instances for which a lower bound is produced even if the corresponding linear relaxation was not solved to optimality; (iii) # bounds: number of instances for which a lower bound was produced; (iv) # nodes: number of nodes processed by the MILP solvers or branch-and-price.

Table 6 focuses on the solution of the linear relaxations of the formulations. It shows that for three classes of instances out of four, the column generation algorithm produces on average the best lower bounds. This behaviour is in line with what was expected since it was proven (Lemma 2) that formulation (EFsos2) dominates (CFsos2). However, there is one class of instances, $|\mathcal{J}| = 60$ and $|\mathcal{T}| \geqslant 700$, for which column generation did not produce the best average lower bound ratio. This is certainly related to the high number of time out occurrences: less than 45% of instances of that class could be solved to optimality under the time limit, therefore the lower bounds produced were lower than what should have been the optimal solution value of the linear relaxation of (EFsos2).

Instances			Results with the linear relaxations				
			CPLEX		SCIP	Column generation	
Characteristics	#		ilopwl	sos2	sos2	variant 1	variant 2
$\|\mathcal{J}\| = 30$ and $\|\mathcal{T}\| \leqslant 700$	27	LB ratio	92.77 %	91.82 %	93.75 %	**96.25 %**	96.13 %
		# time out	0	0	0	1	1
		# bounds	27	27	27	27	27
$\|\mathcal{J}\| = 30$ and $\|\mathcal{T}\| \geqslant 700$	27	LB ratio	91.41 %	87.22 %	93.11 %	**94.12 %**	93.5 %
		# time out	0	0	0	19	19
		# bounds	27	27	27	27	27
$\|\mathcal{J}\| = 60$ and $\|\mathcal{T}\| \leqslant 700$	19	LB ratio	93.96 %	92.36 %	93.91 %	**96.04 %**	**96.05 %**
		# time out	0	0	0	2	3
		# bounds	19	19	19	19	19
$\|\mathcal{J}\| = 60$ and $\|\mathcal{T}\| \geqslant 700$	55	LB ratio	**92.67 %**	89.06 %	92.28 %	91.00 %	91.73 %
		# time out	0	0	0	31	30
		# bounds	55	55	55	54	52

Table 6: Computational results obtained with the linear relaxations and with 600 s time limit

Table 7 reports the results of the lower bounding schemes with no nodes limits, i.e branch-and-bound for the MILP solvers and branch-and-price for the (EFsos2) solution method. Three main observations can be made. First, the extended formulation (EFsos2) solved with SCIP-based branch-and-price outperforms the compact formulation (CFsos2) solved with the same SCIP solver. Second, CPLEX sos2 performs better than SCIP sos2, which means that there may be valid inequalities, preprocessing or more efficient branching schemes in CPLEX in comparison to SCIP. In spite of that (EFsos2) solved with the SCIP-based branch-and-price outperforms (CFsos2) solved with CPLEX. Thirdly, the build-in ilopwl-based formulation performs better on instances with a high number of time periods. Finally, we observe that on average among all instances the ratio is larger than 95%, which illustrate the quality of the proposed upper bound.

Instances			Results with a 600 s time limit (no nodes limit)				
			CPLEX		SCIP	Branch-and-Price	
Characteristics	#		ILOPWL	SOS2	SOS2	variant 1	variant 2
$\|\mathcal{J}\| = 30$ and $\|\mathcal{T}\| \leqslant 700$	27	LB ratio	95.64 %	94.32 %	93.11 %	**96.27 %**	96.17 %
		# nodes	655385	633850	323925	1474	1445
		# bounds	27	26	27	27	27
$\|\mathcal{J}\| = 30$ and $\|\mathcal{T}\| \geqslant 700$	27	LB ratio	**94.79 %**	93.69 %	93.75 %	94.14 %	93.51 %
		# nodes	426890	446445	323924	815	799
		# bounds	27	23	27	27	27
$\|\mathcal{J}\| = 60$ and $\|\mathcal{T}\| \leqslant 700$	19	LB ratio	95.90 %	94.48 %	94.15 %	**96.04 %**	**96.06 %**
		# nodes	579335	597733	300378	1205	1160
		# bounds	18	19	19	19	19
$\|\mathcal{J}\| = 60$ and $\|\mathcal{T}\| \geqslant 700$	55	LB ratio	**95.36 %**	93.94 %	92.46 %	91.05 %	91.65 %
		# nodes	277454	290565	115629	426	409
		# bounds	55	53	55	54	52

Table 7: Computational results obtained with a 600 s time limit and no nodes limit

6 Concluding remarks

This paper presents mathematical models and decomposition schemes for computing upper and lower bounds for a strongly NP-hard scheduling problem with reversible and non-reversible energy sources and time-dependent piecewise linear energy costs. The problem allows to represent practical situations (such as smart buildings, data centers, manufacturing) where the energy load is a consequence of discrete scheduling decisions and has to be dispatched among external energy sources (such as Photovoltaïc panels and the grid) with the support of storage resources (such as batteries). The proposed decomposition matheuristic solves iteratively with dedicated methods a scheduling subproblem with no storage and a lot-sizing subproblem for energy dispatch. The time-dependent piecewise linear functions used in the scheduling subproblem are updated after each lot-sizing solution. Computational results, on randomly generated and adapted instances from the literature, show that the matheuristic produces fastly high quality solution compared to a compact MILP formulation. To assess the quality of the obtained upper bounds, several lower bounding schemes are designed and compared, among which various ways of solving the compact MILP formulation and a new extended formulation solved by column generation and branch-and-price. The LP relaxation of the extended formulation is shown to strictly dominate the LP relaxation of the compact formulation. On the considered instances, the lower bounding scheme based on branch-and-price outperforms the one based on the compact formulation with standard SOS2 modeling of piecewise linear constraints and is competitive with CPLEX using the built-in piecewise linear constraints. Finally, the gap between the best lower bound and upper bound is less that 5% on average, which validates the interest of solving iteratively the scheduling subproblem and the energy dispatch problem for practical applications.

Future work will seek on the one hand to integrate more realistic characteristics of the energy resources or of the scheduling part, mainly considering additional resources such as machines and manpower. On the other hand, an exact Benders-like decomposition method could be derived from the matheuristic. The main issue is that the (combinatorial) Benders cut may yield a too hard-to-solve scheduling problem while the heuristic scheme allowed to pass information only via the modification of the piecewise linear functions without changing the scheduling subproblem structure. The branch-and-price approach is also not competitive yet with the compact formulation with built-in piecewise linear contraints on a significant number of cases. Whether it is feasible to overcome this issue is also an open research direction.

Acknowledgements

This research benefited from the support of the FMJH "Program Gaspard Monge for optimization and operations research and their interactions with data science", and from the support from EDF and/or Thales. This work was also supported by the ANR Project PICOMODALITE.

The contribution of Félix Goupil, Janik Rannou and Omar Saadi in preliminary internships is gratefully acknowledged.

References

[1] Absi, N., Artigues, C., Kedad-Sidhoum, S., Ngueveu, S.U., Saadi, O.: Lot-sizing models for energy management. In: International Workshop on Lot Sizing - IWLS'2017, pp. 45–48. Glasgow (2017)

[2] Bambagini, M., Marinoni, M., Aydin, H., Buttazzo, G.: Energy-aware scheduling for real-time systems: A survey. ACM Transactions on Embedded Computing Systems (TECS) **15**(1), 1–34 (2016)

[3] Beale, E.M.L., Forrest, J.J.H.: Global optimization using special ordered sets. Mathematical Programming **10**(1), 52–69 (1976)

[4] Beldiceanu, N., Poder, E.: A continuous multi-resources cumulative constraint with positive-negative resource consumption-production. In: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming, pp. 214–228. Springer (2007)

[5] Brahimi, N., Absi, N., Dauzère-Pérès, S., Nordli, A.: Single-item dynamic lot-sizing problems: An updated survey. European Journal of Operational Research **263**(3), 838–863 (2017)

[6] Carlier, J., Moukrim, A.: Storage resources. In: Handbook on Project Management and Scheduling Vol. 1, pp. 177–189. Springer (2015)

[7] Carlier, J., Moukrim, A., Sahli, A.: Lower bounds for the event scheduling problem with consumption and production of resources. Discrete Applied Mathematics **234**, 178–194 (2018)

[8] Carlier, J., Moukrim, A., Xu, H.: The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm. Discrete Applied Mathematics **157**(17), 3631–3642 (2009)

[9] Ji, K., Chi, C., Marahatta, A., Zhang, F., Liu, Z.: Energy efficient scheduling based on marginal cost and task grouping in data centers. In: Proceedings of the Eleventh ACM International Conference on Future Energy Systems, pp. 482–488 (2020)

[10] Keha, A.B., de Farias Jr, I.R., Nemhauser, G.L.: A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. Operations research **54**(5), 847–858 (2006)

[11] Koné, O., Artigues, C., Lopez, P., Mongeau, M.: Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. Flexible Services and Manufacturing Journal **25**(1-2), 25–47 (2013)

[12] Laborie, P.: Algorithms for propagating resource constraints in ai planning and scheduling: Existing approaches and new results. Artificial Intelligence **143**(2), 151–188 (2003)

[13] Meng, L., Sanseverino, E.R., Luna, A., Dragicevic, T., Vasquez, J.C., Guerrero, J.M.: Microgrid supervisory controllers and energy management systems: A literature review. Renewable and Sustainable Energy Reviews **60**, 1263–1273 (2016)

[14] Merkert, L., Harjunkoski, I., Isaksson, A., Säynevirta, S., Saarela, A., Sand, G.: Scheduling and energy–industrial challenges and opportunities. Computers & Chemical Engineering **72**, 183–198 (2015)

[15] Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L.: An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Management science **44**(5), 714–729 (1998)

[16] Neumann, K., Schwindt, C.: Project scheduling with inventory constraints. Mathematical Methods of Operations Research **56**(3), 513–533 (2003)

[17] Ngueveu, S.U., Artigues, C., Lopez, P.: Scheduling under a non-reversible energy source: An application of piecewise linear bounding of non-linear demand/cost functions. Discrete Applied Mathematics **208**, 98–113 (2016)

[18] Olatomiwa, L., Mekhilef, S., Ismail, M.S., Moghavvemi, M.: Energy management strategies in hybrid renewable energy systems: A review. Renewable and Sustainable Energy Reviews **62**, 821–835 (2016)

[19] Parisio, A., Rikos, E., Glielmo, L.: A model predictive control approach to microgrid operation optimization. IEEE Transactions on Control Systems Technology **22**(5), 1813–1827 (2014)

[20] Sahli, A., Carlier, J., Moukrim, A.: Comparison of mixed integer linear programming models for the event scheduling problem with consumption and production of resources. IFAC-PapersOnLine **49**(12), 1044–1049 (2016)

[21] Schutt, A., Stuckey, P.J.: Explaining producer/consumer constraints. In: International Conference on Principles and Practice of Constraint Programming, pp. 438–454. Springer (2016)

[22] Shaw, D.X., Wagelmans, A.P.: An algorithm for single-item capacitated economic lot sizing with piecewise linear production costs and general holding costs. Management Science **44**(6), 831–838 (1998)

[23] Sourd, F., Rogerie, J.: Continuous filling and emptying of storage systems in constraint-based scheduling. European journal of operational research **165**(2), 510–524 (2005)

[24] Wei, W., Liu, F., Mei, S.: Energy pricing and dispatch for smart grid retailers under demand response and market price uncertainty. IEEE transactions on smart grid **6**(3), 1364–1374 (2014)

[25] Zia, M.F., Elbouchikhi, E., Benbouzid, M.: Microgrids energy management systems: A critical review on methods, solutions, and prospects. Applied energy **222**, 1033–1055 (2018)