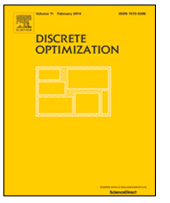
Discrete Optimization 39 (2021) 100627

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt



Greedy algorithm for local heating problem☆

Jiří Fink *, Johann L. Hurink

University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, P.O. Box 217, 7500 AE, Enschede, The Netherlands
Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University, Malostranske namesti 25, 118 00, Praha, Czech republic

ARTICLE INFO

Article history:
Received 6 December 2019
Accepted 14 January 2021
Available online 5 February 2021

Keywords:
Greedy algorithm
Union-find data structure
Smart Grids

ABSTRACT

This paper studies a planning problem for heating water. Hereby, boilers (e.g. gas or electric boilers, heat pumps or microCHPs) are used to heat the water and store it for domestic demands. We consider a simple boiler which is either turned on or turned off and has a buffer of limited capacity. The energy needed to run the boiler has to be bought on a day-ahead market, so we are interested in a planning which minimizes the cost to supply the boiler with energy. We present a greedy algorithm whose time complexity is $\mathcal{O}(T\alpha(T))$ where T is the number of time intervals and α is the inverse of Ackermann function.

1. Introduction

In modern society, a significant amount of energy is consumed for heating water [1]. Almost every building is connected to a district heating system or equipped with appliances for heating water locally. Typical appliances for heating water are electrical and gas heating systems, heat pumps and Combined Heat and Power units (microCHP). The heated water is stored in buffers to be prepared for the demands of the inhabitants of the building.

In this paper we consider a local heating systems which consist of

- a supply which represents some source of energy (electricity, gas),
- a converter which converts the energy into heat (hot water),
- a buffer which stores the heat for later usage and
- a demand which represents the (predicted) consumption profile of heat.

☆ This research is conducted within the iCare project (11854) supported by STW. The first author is also supported within the project Network Optimization (17-10090Y) supported by Czech Science Foundation.

* Corresponding author.
E-mail addresses: fink@ktiml.mff.cuni.cz (J. Fink), j.l.hurink@utwente.nl (J.L. Hurink).

https://doi.org/10.1016/j.disopt.2021.100627
1572-5286/

A more formal definition of the considered setting for local heating and the used parameters and variables is given in Section 1.1. The presented model can consider arbitrary types of energy but in this paper we use *electricity* and *heat* to distinguish consumed and produced energy. However, this simple model of a local heating system cannot only be applied for heating water but has many other applications, e.g. heating demand of houses, fridges and freezers and inventory management. More details about those applications are given also in Section 1.1.

The electricity used to heat the water has to be bought. Although these prices are nowadays mostly fixed for private costumers, the supply companies are faced with variable prices resulting e.g. from a day ahead market. This leads to the objective of minimizing the total cost of electricity consumed by the heating system during the planning period. Note that in cost or auction based control algorithms for Smart Grids, this objective is also used (see e.g. [2]).

1.1. Problem statement and results

In the following we present a mathematical description of the studied model and a summary of the results of this paper.

First of all, we consider a discrete time model for the considered problem, meaning that we split the planning period into T time intervals of the same length resulting in a set $\mathcal{T} = \{1, \dots, T\}$ of time intervals. In this paper, the letter t is always used as an index of time intervals.

For the heating system, we consider a simple converter which has only two states: In every time interval the converter is either turned on or turned off. The amount of produced heat during one time interval in which the converter is turned on is denoted by H. If the converter is turned off, then it consumes and produces no energy. Let $x_t \in \{0, 1\}$ be the variable indicating whether the converter is running in time interval $t \in \mathcal{T}$ or not. Furthermore, if the converter is running, then it consumes some amount of electricity which costs P_t in time interval $t \in \mathcal{T}$. In another words, P_t is the price for running the converter in time interval $t \in \mathcal{T}$. Summarizing, the objective of the planning problem is minimizing the cost for producing the heat, which is given by $\sum_{t \in \mathcal{T}} P_t x_t$.

Coupled to the heating system is a buffer. The state of charge of the buffer in the beginning of time interval $t \in \mathcal{T}$ is denoted by s_t and represents the amount of heat in the buffer. Note that s_{T+1} is the state of charge at the end of planning period. Based of the physical properties of the buffer, the state of charge s_t is limited by a lower bound L_t and an upper bound U_t. These two bounds are usually constant over time: the upper bound U_t is the capacity of buffer and the lower bound L_t is zero. But it may be useful to allow different values, e.g. a given initial state of charge can be modelled by setting L_1 and U_1 equal to the initial state. In this paper, we always assume that $L_1 = U_1$, meaning that the initial state of charge s_1 is fixed.

The (predicted) amount of consumed heat by the inhabitants of the house from the boiler during time interval $t \in \mathcal{T}$ is denoted by D_t. This amount is assumed to be given and is called the demand. In this paper, we study off-line version of the problem, so we assume that both the demands D_t and also the prices P_t are given for the whole planning period already at time the beginning of the planning period.

The variables x_t specifying the operation of the converter and the states of charge of the buffer s_t are restricted by the following constraints.

$$s_{t+1} = s_t + Hx_t - D_t \quad \text{for} \quad t \in \mathcal{T} \tag{1}$$

$$L_t \leq s_t \leq U_t \quad \text{for} \quad t \in \{1, \dots, T+1\} \tag{2}$$

$$x_t \in \{0, 1\} \quad \text{for} \quad t \in \mathcal{T} \tag{3}$$

Eq. (1) is the charging equation of the buffer. During time interval $t \in \mathcal{T}$, the state of charge s_t of the buffer is increased by the production of the converter which is Hx_t and it is decreased by demand D_t. Eqs. (2)

and (3) ensure that the domains of variables s_t and x_t, respectively, are taken into account. In this paper, the objective function is to minimize the cost for the electricity needed to produce the heat and this cost is given by the sum $\sum_{t\in\mathcal{T}} P_t x_t$.

In a previous paper [3], we presented an algorithm for the problem of minimizing cost for the local heating which is based on dynamic programming and it has the time complexity $\mathcal{O}(T^2)$. In Section 4 we prove that the optimal solution also can be found using a greedy algorithm. This greedy algorithm first sorts all time intervals by their prices P_t, and then it processes all time intervals one-by-one. In the basic version of the algorithm, the necessary updates in each step take time $\mathcal{O}(T)$, so the total time complexity of the algorithm is $\mathcal{O}(T^2)$, which is the same as the dynamic programming algorithm. In Section 5, we use disjoint-set data structure of the union-find algorithm (see e.g. Cormen et al. [4]) to obtain a complexity of $\mathcal{O}(T\alpha(T))$ where α is the inverse of Ackermann function. Hereby, we ignore the complexity of sorting the time intervals since the order may be a part of the input or a be found using a bucket sort algorithm (see e.g. Cormen et al. [4]).

2. Related works and applications

In the following we present related literature and give some possible applications of this model.

Some related works can be found in the inventory management and lotsizing literature (see e.g. [5,6] for reviews). In inventory control problems (see [7]) a buffer may represent an inventory of items, whereby a converter represent the production of items and demand represent the ordered quantities. As our problem consists of only one commodity, the single item lot sizing problem is related (see [8] for a review). Wagner and Whitin [9] presented an $\mathcal{O}(T^2)$ algorithm for the uncapacitated lot-sizing problem which was improved by Federgruen and Tzur [10] to $\mathcal{O}(T \log T)$. On the other hand, Florian, Lenstra and Rinnooy [11] proved that the lot-sizing problem with upper bounds on production and order quantities is NP-complete. Computational complexity of the capacited lot sizing problems is studied in [12]. Our problem is a special case of capacited single item lot sizing problem which does not seem to be considered in the literature.

One other related area is vehicle routing and scheduling (see e.g. [13] for an overview of this area). For example, Lin, Gertsch and Russell [14] studied optimal vehicle refuelling policies. In their model, a refuelling station can provide an arbitrary amount of gas while our converter is restricted to two possible states of heat generation. Other papers on vehicle refuelling policies are more distant from our research since they consider that a car is routed on a graph (see e.g. [15,16]).

In the following we give some possible applications of the model presented in this paper.

Hot water: Converter and buffer can be seen as a model of a simple electrical or gas boiler. Hereby, demand represents the consumption of hot water in a house.

House Heating: The model may be used to express a very simple model for house heating. The converter represents a simple heater. The capacity of the buffer corresponds to thermal capacity of the heating system (e.g. hot water buffer or thermal capacity of concrete floors and walls) and the state of charge of the buffer is related to the temperature inside the house. Heat losses of the house may be modelled using the demand if we assume that the temperature difference inside the house does not have significant influence on the losses. More details about using thermal mass as a buffer is presented in [17] and computing heat demands is explained in [18].

Fridges and freezers: A fridge essentially works in the opposite way than heating, so it may be modelled similarly. However, we have be careful with the correct interpretation of all parameters. The state of charge of the buffer again represents the temperature inside the fridge, but a higher state of charge means a lower temperature. The converter does not produce heat to the fridge but it decreases the temperature inside the fridge, so the converter increases the state of charge of the buffer (fridge). The demand decreases the state of charge of the fridge due to thermal loss and usage of the fridge by humans.

3. Reformulation of the problem

In [3] a reformulation of the problem presented in Section 1.1 is given, which enables a better presentation and analysis of our algorithm. For sake of completeness, we give this reformulation in this section. We show that conditions (1) and (2) can be replaced by one condition (8).

First, we expand the recurrence formula (1) into an explicit equation

$$s_{t+1} = s_1 + \sum_{i=1}^{t} Hx_i - \sum_{i=1}^{t} D_i.$$

Since we assume that the initial state of charge satisfies $s_1 = L_1 = U_1$, we can replace s_1 by L_1 and substitute this into inequalities (2), leading to

$$L_{t+1} \leq L_1 + H\sum_{i=1}^{t} x_i - \sum_{i=1}^{t} D_i \leq U_{t+1}$$

which can be rewritten as

$$\frac{L_{t+1} - L_1 + \sum_{i=1}^{t} D_i}{H} \leq \sum_{i=1}^{t} x_i \leq \frac{U_{t+1} - L_1 + \sum_{i=1}^{t} D_i}{H}.$$

Since the sum $\sum_{i=1}^{t} x_i$ is an integer between 0 and t we obtain the following simple constrains for this sums

$$A'_t \leq \sum_{i=1}^{t} x_i \leq B'_t \text{ for } t \in \mathcal{T} \tag{4}$$

where

$$A'_t = \max\left\{0, \left\lceil \frac{L_{t+1} - L_1 + \sum_{i=1}^{t} D_i}{H} \right\rceil\right\}, \tag{5}$$

$$B'_t = \min\left\{t, \left\lfloor \frac{U_{t+1} - L_1 + \sum_{i=1}^{t} D_i}{H} \right\rfloor\right\}. \tag{6}$$

The values of A'_t and B'_t for every $t \in \mathcal{T}$ can easily be computed in time $\mathcal{O}(T)$. In the rest of this section we study properties of sequences $A'_1, \ldots, A'_T$ and $B'_1, \ldots, B'_T$ which are shortly denoted by $(A'_t)_t$ and $(B'_t)_t$, respectively.

Observe that the sequence of partial sums $\sum_{i=1}^{t} x_i$ for $t = 1, \ldots, T$ is non-decreasing and the difference of two consecutive elements is at most 1. We say that a sequence $(Z_t)_t$ of $T+1$ integers $Z_0, Z_1, \ldots, Z_T$ satisfies (7) if

$$\begin{aligned} &Z_0 = 0, \\ &Z_{t-1} \leq Z_t \leq Z_{t-1} + 1 \text{ for all } t \in \mathcal{T}. \end{aligned} \tag{7}$$

We show that we can replace parameters A'_t and B'_t by parameters A_t and B_t such that sequences $(A_t)_t$ and $(B_t)_t$ satisfy (7) and the binary variables x_t satisfy (4) if and only if they satisfy

$$A_t \leq \sum_{i=1}^{t} x_i \leq B_t \text{ for } t \in \mathcal{T}. \tag{8}$$

The idea of the replacement is presented in Fig. 1. It is easy to see that for this example every solution $(x_t)_t$ that satisfies (4) also satisfies (8) and vice versa. In [3] an algorithm to obtain required sequences $(A_t)_t$ and $(B_t)_t$ is given.

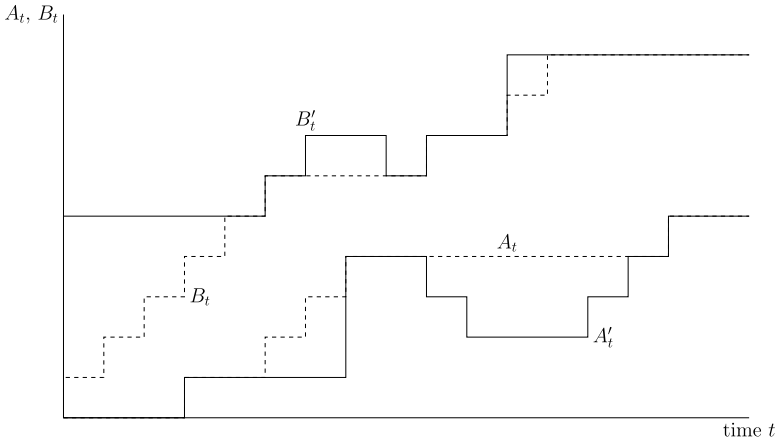


Fig. 1. Precomputation of lower and upper bounds on the sequence of partial sums $\sum_{i=1}^{t} x_i$ according to Section 3. Full lines are the original bounds $(A'_t)_t$ and $(B'_t)_t$ defined by formulas (5) and (6) and dashed lines are the precomputed bounds $(A_t)_t$ and $(B_t)_t$.

4. Greedy algorithm

In this section we present a greedy algorithm to the problem of fulfilling the heat demand with minimal cost which is based on solution that can be formulated as

$$\begin{aligned} \text{Minimize} \quad & \sum_{t\in\mathcal{T}} P_t x_t \\ \text{such that } A_t \le & \sum_{i=1}^{t} x_i \le B_t \quad \text{for } t \in \mathcal{T} \\ & x_t \in \{0,1\} \qquad\quad \text{for } t \in \mathcal{T} \end{aligned} \tag{9}$$

For the following, we assume that the bounds A_t and B_t are precomputed, meaning that the sequences $(A_t)_t$ and $(B_t)_t$ satisfy (7).

The first natural question is under which conditions problem (9) has a feasible solution. An obvious necessary condition for the existence of a feasible solution x_t is that $A_t \le B_t$ for every $t \in \mathcal{T}$. This condition is also sufficient, since in this case $x_t = A_t - A_{t-1}$ for $t \in \mathcal{T}$ gives a feasible solution. Summarizing, we get the following lemma.

Lemma 4.1. *The problem (9) has a feasible solution if and only if*

$$A_t \le B_t \text{ for every } t \in \mathcal{T}. \tag{10}$$

Since the condition (10) can easily be evaluated in linear time, we assume in the remainder of the paper that the problem (9) has a feasible solution. To solve the problem, we use the classical greedy approach. First, the time intervals are sorted by prices P_t. Then, time intervals are processed in order of increasing prices and the converter is turned on in time interval $t^\star \in \mathcal{T}$, if there exists a feasible solution satisfying $x_{t^\star} = 1$. Such a feasible solution implies

$$A_{t^\star-1} \le \sum_{i=1}^{t^\star-1} x_i < \sum_{i=1}^{t^\star} x_i \le B_{t^\star}$$

which implies the following lemma.

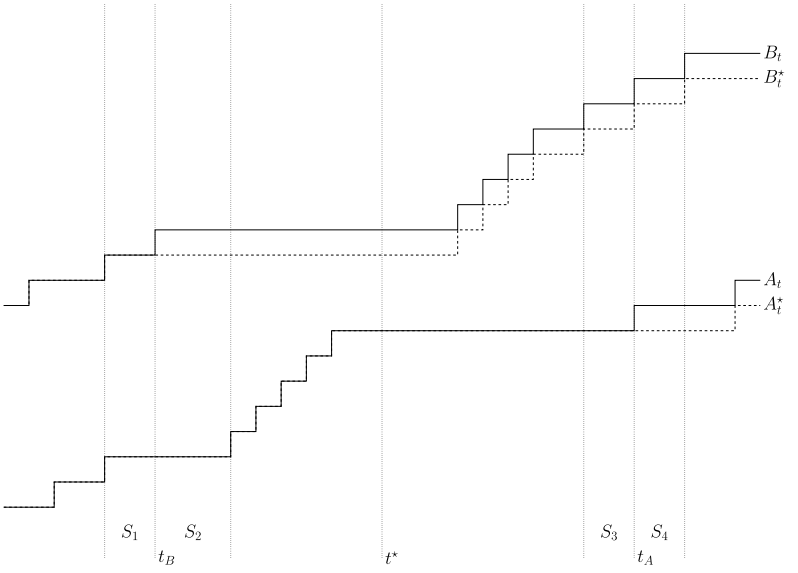


Fig. 2. Sequences $(A_t)_t$, $(B_t)_t$, $(A_t^\star)_t$ and $(B_t^\star)_t$ according to Lemma 4.3.

Lemma 4.2. *If the problem (9) has a feasible solution $(x_t)_t$ satisfying $x_{t^\star} = 1$ for a given $t^\star \in \mathcal{T}$, then the inequality $A_{t^\star-1} < B_{t^\star}$ holds.*

Note that the condition $A_{t^\star-1} < B_{t^\star}$ in the lemma is well-defined also for $t^\star = 1$ since we consider that $A_0 = 0$. The condition $A_{t^\star-1} < B_{t^\star}$ in Lemma 4.2 is also sufficient (if (10) is satisfied) and is the base for the presented approach. At this point, we do not give a formal proof of this implication, but the proof follows from the further considerations and lemmas (e.g. 4.3 or 4.5).

The greedy algorithm starts with the (infeasible) solution $x_t = 0$ for every $t \in T$. It finds the cheapest time interval $t^\star$ satisfying $A_{t^\star-1} < B_{t^\star}$ and it sets $x_{t^\star} := 1$. After choosing $t^\star$ and setting $x_{t^\star} = 1$, the condition $A_{t-1} < B_t$ do no longer imply that x_t can be set to 1. We first have to adopt the values of sequences $(A_t)_t$ and $(B_t)_t$ to incorporate the choice $x_{t^\star} = 1$. The following lemma specifies how the values of $(A_t)_t$ and $(B_t)_t$ have to be updated in every step and shows that this update is correct.

The lemma finds the time interval t_A where the sequence $(A_t)_t$ has the first increasing value after the time interval $t^\star$, and the time interval t_B where the sequence $(B_t)_t$ has the last increasing value before the time interval $t^\star$. These increases at time intervals t_A and t_B are removed from sequences $(A_t)_t$ and $(B_t)_t$, respectively; that is values of sequences are decreased by one after these time intervals (see Fig. 2).

Lemma 4.3. *Let $t^\star \in \mathcal{T}$ satisfy $A_{t^\star-1} < B_{t^\star}$, and let*

$$t_A = 1 + \max\{t \in \mathcal{T};\ A_t = A_{t^\star-1}\} \qquad t_B = \min\{t \in \mathcal{T};\ B_t = B_{t^\star}\}$$

$$A_t^\star = \begin{cases} A_t & \text{if } t < t_A \\ A_t - 1 & \text{if } t \geq t_A \end{cases} \qquad B_t^\star = \begin{cases} B_t & \text{if } t < t_B \\ B_t - 1 & \text{if } t \geq t_B. \end{cases}$$

Then, sequences $(A_t^\star)_t$ and $(B_t^\star)_t$ satisfy (7) and for every 0–1 sequence $(x_t)_t$ with $x_{t^\star} = 1$ the condition (9) holds if and only if it holds

$$A_t^\star \leq \sum_{\substack{i=1 \\ i \neq t^\star}}^{t} x_i \leq B_t^\star \text{ for every } t \in \mathcal{T}. \tag{11}$$

Before we prove Lemma 4.3 we first explain why we need the equivalence in the lemma. The idea is that sequences $(A^\star_t)_t$ and $(B^\star_t)_t$ form another instance of the problem (9) where the time interval $t^\star$ is skipped. When we find an optimal solution $(x_t)_t$ for the instance of the problem 4 with $(A^\star_t)_t$ and $(B^\star_t)_t$ where $t \in \mathcal{T} \setminus \{t^\star\}$, then we obtain an optimal solution for the original instance with $(A_t)_t$ and $(B_t)_t$ where $t \in \mathcal{T}$ by setting $x_{t^\star} := 1$. Note that Lemma 4.3 does not guarantee the optimality but only the feasibility and that time interval $t^\star$ needs not to be removed in the greedy algorithm since the greedy algorithm process every time interval at most once.

Proof of Lemma 4.3. Note that if $A_{t^\star-1} = A_T$, then $t_A = T + 1$ and the sequences $(A_t)_t$ and $(A^\star_t)_t$ are the same. Otherwise, if $A_{t^\star-1} < A_T$, then $A_{t_A-1} = A_{t_A} - 1$. Furthermore, it holds that $B_{t_B-1} = B_{t_B} - 1$. Therefore, the sequences $(A^\star_t)_t$ and $(B^\star_t)_t$ differ from sequences $(A_t)_t$ and $(B_t)_t$ by removing (at most) one step of the step function. This implies that they satisfy (7).

In order to prove the second part, let $(x_t)_t$ be a 0–1 sequence with $x_{t^\star} = 1$. For such a sequence, the condition (9) is equivalent to

$$\begin{aligned} A_t \le \sum_{\substack{i=1\\ i\neq t^\star}}^{t} x_i \le B_t \qquad & \text{for every } t < t^\star \text{ and} \\ A_t - 1 \le \sum_{\substack{i=1\\ i\neq t^\star}}^{t} x_i \le B_t - 1 \qquad & \text{for every } t \ge t^\star. \end{aligned} \tag{12}$$

Thus, it remains to prove that conditions (11) and (12) are equivalent.

First, we consider the lower bounds. Observe that the lower bounds of (11) and (12) only differ for time intervals $t \in \mathcal{T}$ with $t^\star \le t < t_A$. For such t it holds that $A^\star_t = A_t$ and, thus, we only have to prove that (12) implies (11) since the lower bound in (11) is stronger. However, the implication follows directly from

$$A^\star_t = A_t = A_{t^\star-1} \le \sum_{i=1}^{t^\star-1} x_i \le \sum_{\substack{i=1\\ i\neq t^\star}}^{t} x_i.$$

The upper bounds can be considered similar. Observe that the upper bounds of (11) and (12) differ only for time intervals $t \in \mathcal{T}$ satisfying $t_B \le t < t^\star$. For such t it holds that $B^\star_t = B_t - 1$ and we only have to prove that (12) implies (11) since the upper bound in (11) is stronger. The implication follows from

$$\sum_{\substack{i=1\\ i\neq t^\star}}^{t} x_i \le \sum_{\substack{i=1\\ i\neq t^\star}}^{t^\star} x_i \le B_{t^\star} - 1 = B_t - 1 = B^\star_t. \quad \square$$

In practice, the price of electricity is usually positive. However, we present a greedy algorithm which also works if the price P_t is negative for some $t \in \mathcal{T}$. If all prices are non-negative, then without loss of generality we can assume that $A_T = B_T$ since there is an optimal solution which turns the converter on only A_T-times (that is, there exists an optimal solution with $\sum_{t\in\mathcal{T}} x_t = A_T$). In the general case where prices can be negative, the value of the objective function may be improved by turning the converter on more often. In the later case, we need to extend the condition $A_{t^\star-1} < B_{t^\star}$ of the greedy algorithm to a condition which also considers negative prices. The new condition is

$$A_{t^\star-1} < B_{t^\star} \text{ and } (A_{t^\star-1} < A_T \text{ or } P_{t^\star} < 0). \tag{13}$$

Note, that the condition $A_{t^\star-1} < B_{t^\star}$ is already necessary for setting $x_{t^\star} = 1$ by Lemma 4.2. If $A_{t^\star-1} = A_T$ then the total minimal number of runs of the converter has to be reached already before

Algorithm 4.1: Greedy algorithm for minimizing cost.

Input: Sequences $(A_t)_t$ and $(B_t)_t$ satisfying (7) and (10)
Output: Optimal solution $(x_t)_t$ the problem (9)

initialization: $x_t := 0$ for all $t \in \mathcal{T}$;
for $t^\star \in \mathcal{T}$ *sorted by prices* $(P_t)_t$ **do**
  if *the condition* (13) *is satisfied* **then**
    $x_{t^\star} := 1$;
    Apply Lemma 4.3

return *Optimal solution* $(x_t)_t$

the time interval $t^\star$, so the lower bound $(A_t)_t$ does not force the converter on in the time interval $t^\star$. In this case, it is obvious that an optimal solution satisfies $x_{t^\star} = 0$ unless the price $P_{t^\star}$ is negative.

As already mentioned above, we can assume that $A_T = B_T$ if all prices are non-negative. In this case the condition $A_{t^\star-1} < B_{t^\star}$ implies $A_{t^\star-1} < A_T$, so the condition (13) reduces to the condition $A_{t^\star-1} < B_{t^\star}$.

The greedy algorithm is summarized in Algorithm 4.1. In the following, mathematical induction is used to prove that this greedy algorithm finds an optimal solution. The following lemma provides the base of the induction.

Lemma 4.4. *If $A_T = 0$ and there is no $t^\star \in \mathcal{T}$ such that $B_{t^\star} > 0$ and $P_{t^\star} < 0$, then $x_t = 0$ for all $t \in \mathcal{T}$ is an optimal solution.*

Proof. Since $A_T = 0$ it follows that $A_t = 0$ for all $t \in \mathcal{T}$ and thus, the trivial solution $x_t = 0$ for all $t \in \mathcal{T}$ is feasible. Let $\bar{t} = \max\{x \in \mathcal{T};\ B_t = 0\}$ and let $(\bar{x}_t)_t$ be an arbitrary feasible solution. Observe that $\bar{x}_t = 0$ for $t \leq \bar{t}$; and $P_t \geq 0$ for $t > \bar{t}$. Hence,

$$\sum_{t\in\mathcal{T}} P_t\bar{x}_t = \sum_{t>\bar{t}} P_t\bar{x}_t \geq 0 = \sum_{t\in\mathcal{T}} P_t x_t$$

which implies that the solution $(x_t)_t$ is optimal. □

From Lemma 4.4 it follows that the condition $A_T = 0$ and $P_{t^\star} \geq 0$ can be used as termination condition in Algorithm 4.1. Next, we prove the induction step of the proof that Algorithm 4.1 finds an optimal solution. The following lemma proves that for a time interval $t^\star \in \mathcal{T}$ with minimal prize $P_{t^\star}$ among all intervals satisfying (13), there exists an optimal solution $(x_t)_t$ with $x_{t^\star} = 1$. The proof essentially processes by a contradiction, so it is assumed that there exist an optimal solution $(\bar{x}_t)_t$ but $\bar{x}_t = 0$; and a solution $(\hat{x}_t)_t$ is constructed such that $\hat{x}_t = 1$ and $\sum_{t\in\mathcal{T}} P_t\hat{x}_t \leq \sum_{t\in\mathcal{T}} P_t\bar{x}_t$. The construction finds a time interval $\hat{t}$ which is either the maximal $\hat{t} < t^\star$ or the minimal $\hat{t} > t^\star$ such that $x_{\hat{t}} = 1$; and the running of the converter is rescheduled from the time interval $\hat{t}$ to $t^\star$ without increasing the objective. Two cases result from the following distention: If $\sum_{i=1}^{t^\star-1} \bar{x}_i = A_{t^\star-1}$, then the minimal $\hat{t} > t^\star$ need to be used and if $\sum_{i=1}^{t^\star} \bar{x}_i = B_{t^\star}$, then the maximal $\hat{t} < t^\star$ need to be used.

Lemma 4.5. *Assuming that there exists a time interval $t^\star \in \mathcal{T}$ satisfying (13), let $t^\star$ be the time interval satisfying (13) with the minimal price $P_{t^\star}$. Then, there exists an optimal solution $(x_t)_t$ such that $x_{t^\star} = 1$.*

Proof. We prove the lemma indirectly by proving that for every feasible solution $(\bar{x}_t)_t$ there exists a feasible solution $(\hat{x}_t)_t$ such that $\hat{x}_{t^\star} = 1$ and $\sum_{t\in\mathcal{T}} P_t\hat{x}_t \leq \sum_{t\in\mathcal{T}} P_t\bar{x}_t$. Since we assume that there always exists a feasible solution, the lemma follows from this observation.

Let $(\bar{x}_t)_t$ be a feasible solution. If $\bar{x}_t = 1$, we are done. Thus, we assume that $\bar{x}_t = 0$, and we consider two cases.

Case 1 $\sum_{i=1}^{t^\star-1} \bar{x}_i > A_{t^\star-1}$: Let $\hat{t} = \max\{t < t^\star\,;\, \bar{x}_t = 1\}$ which is well-defined since $\sum_{i=1}^{t^\star-1} \bar{x}_i > A_{t^\star-1} \geq 0$. The new solution now is defined by $\hat{x}_{\hat{t}} = 0$ and $\hat{x}_{t^\star} = 1$ and $\hat{x}_t = \bar{x}_t$ for $t \in \mathcal{T} \setminus \{\hat{t}, t^\star\}$. In order to prove that $(\hat{x}_t)_t$ fulfils the mentioned conditions, we first prove that it is feasible. The equality $\sum_{i=1}^{t} \hat{x}_t = \sum_{i=1}^{t} \bar{x}_t$ does not hold only for time intervals t with $\hat{t} \leq t < t^\star$. However, for such t it holds that

$$A_t \leq A_{t^\star-1} \leq \sum_{i=1}^{t^\star-1} \bar{x}_i - 1 = \sum_{i=1}^{t} \bar{x}_i - 1 = \sum_{i=1}^{t} \hat{x}_i < \sum_{i=1}^{t} \bar{x}_i \leq B_t.$$

Hence, $(\hat{x}_t)_t$ is feasible.

Next, for sake of contradiction we assume $\sum_{t\in\mathcal{T}} P_t\hat{x}_t > \sum_{t\in\mathcal{T}} P_t\bar{x}_t$, implying that $P_{\hat{t}} < P_{t^\star}$. If $\hat{t}$ satisfies (13), we have a contradiction with the definition for $t^\star$, so $\hat{t}$ does not satisfy (13). From Lemma 4.2 applied on $\bar{x}_{\hat{t}} = 1$ it holds that $A_{\hat{t}-1} < B_{\hat{t}}$ which implies that $A_{\hat{t}-1} = A_T$ and $P_{\hat{t}} \geq 0$. Since $\hat{t} < t^\star$ we have $A_{\hat{t}-1} = A_{t^\star-1} = A_T$. Furthermore, we have $P_{t^\star} > P_{\hat{t}} \geq 0$, meaning that $t^\star$ does not satisfy (13) which is a contradiction. Thus, $\sum_{t\in\mathcal{T}} P_t\hat{x}_t \leq \sum_{t\in\mathcal{T}} P_t\bar{x}_t$.

Case 2 $\sum_{i=1}^{t^\star-1} \bar{x}_i = A_{t^\star-1}$: Unlikely, we now need two subcases depending on whether there exists $t \geq t^\star$ such that $\bar{x}_t = 1$.

Case 2.1 $\hat{t} = \min\{t \geq t^\star\,;\, \bar{x}_t = 1\} \in \mathcal{T}$ is well-defined: The new solution is defined by $\hat{x}_{\hat{t}} = 0$ and $\hat{x}_{t^\star} = 1$ and $\hat{x}_t = \bar{x}_t$ for $t \in \mathcal{T} \setminus \{\hat{t}, t^\star\}$. In order to prove that $(\hat{x}_t)_t$ fulfils the mentioned conditions, we first prove that it is feasible. The equality $\sum_{i=1}^{t} \hat{x}_t = \sum_{i=1}^{t} \bar{x}_t$ do not holds only for time intervals t with $t^\star \leq t < \hat{t}$. However, for such t holds that

$$A_t \leq \sum_{i=1}^{t} \bar{x}_i < \sum_{i=1}^{t} \hat{x}_i = \sum_{i=1}^{t} \bar{x}_i + 1 = \sum_{i=1}^{t^\star-1} \bar{x}_i < A_{t^\star-1} + 1 \leq B_{t^\star} \leq B_t.$$

Hence, $(\hat{x}_t)_t$ is feasible.

Next, for sake of contradiction we assume $\sum_{t\in\mathcal{T}} P_t\hat{x}_t > \sum_{t\in\mathcal{T}} P_t\bar{x}_t$, implying that $P_{\hat{t}} < P_{t^\star}$. If $\hat{t}$ satisfies (13), we have a contradiction with the definition of $t^\star$, so $\hat{t}$ does not satisfy (13). From Lemma 4.2 applied for $\bar{x}_{\hat{t}} = 1$ it holds that $A_{\hat{t}-1} < B_{\hat{t}}$ which implies that $A_{\hat{t}-1} = A_T$ and $P_{\hat{t}} \geq 0$. Since $A_{t^\star-1} = \sum_{i=1}^{t^\star-1} \bar{x}_i = \sum_{i=1}^{t^\star-1} \hat{x}_i = A_{\hat{t}-1} = A_T$, we have $P_{t^\star} < 0$ by definition of $t^\star$. This implies $0 > P_{t^\star} > P_{\hat{t}} \geq 0$ which is a contradiction. Thus, $\sum_{t\in\mathcal{T}} P_t\hat{x}_t \leq \sum_{t\in\mathcal{T}} P_t\bar{x}_t$.

Case 2.2 $\bar{x}_t = 0$ for all $t \geq t^\star$: Let $\hat{x}_{t^\star} = 1$ and $\hat{x}_t = \bar{x}_t$ for $t \in \mathcal{T} \setminus \{t^\star\}$. It follows from $\sum_{i=1}^{t^\star} \hat{x}_i = A_{t^\star-1} + 1 \leq B_{t^\star}$ that the solution $(\hat{x}_t)_t$ is feasible. If we assume $A_{t^\star-1} < A_T$, then $\sum_{i=1}^{T} \bar{x}_i = \sum_{i=1}^{t^\star-1} \bar{x}_i = A_{t^\star-1} < A_T$, which implies that $(\bar{x}_t)_t$ is infeasible. Hence, $P_{t^\star} < 0$. But since $\sum_{t\in\mathcal{T}} P_t\hat{x}_t < \sum_{t\in\mathcal{T}} P_t\bar{x}_t$, it follows that the solution $(\hat{x}_t)_t$ satisfies all requirements. □

Theorem 4.6. *Algorithm 4.1 finds an optimal solution for the local heating problem in time* $\mathcal{O}(T^2)$.

Proof. Algorithm 4.1 assumes that the pre-computed sequences $(A_t)_t$ and $(B_t)_t$ satisfy (7) and (10). This initialization can easily be computed in linear time.

We use induction on the number of updates according to Lemma 4.3 to prove that Algorithm 4.1 finds an optimal solution. As base of the induction, we assume that Algorithm 4.1 newer applies Lemma 4.3. In this case, there is no time interval $t^\star$ which satisfies (13). Hence, $A_T = 0$ and Lemma 4.4 implies that the trivial solution $x_t = 0$ for all time intervals $t \in \mathcal{T}$ is an optimal solution.

For the induction step, assume that Algorithm 4.1 finds a time interval $t^\star$ satisfying (13) with the minimal price $P_{t^\star}$ and that Lemma 4.3 is applied with this $t^\star$. By the induction hypothesis, Algorithm 4.1 finds an

optimal solution $(x_t)_t$ for the instance with sequences $(A^\star_t)$ and $(B^\star_t)_t$ where $t \in \mathcal{T} \setminus \{t^\star\}$. Algorithm 4.1 now extends this solution by setting $x_{t^\star} = 1$. This solution $(x_t)_t$ is feasible for $(A_t)_t$ and $(B_t)_t$ by Lemma 4.3.

By Lemma 4.5 there exists an optimal solution $(\bar{x}_t)_t$ satisfying $\bar{x}_{t^\star} = 1$. By Lemma 4.3 the solution $(\bar{x}_t)_t$ is feasible for the instance with sequences $(A^\star_t)$ and $(B^\star_t)_t$ where $t \in \mathcal{T} \setminus \{t^\star\}$. From induction hypothesis it follows that $\sum_{t\in\mathcal{T}\setminus\{t^\star\}} P_t x_t \leq \sum_{t\in\mathcal{T}\setminus\{t^\star\}} P_t \bar{x}_t$. Hence, $\sum_{t\in\mathcal{T}} P_t x_t \leq \sum_{t\in\mathcal{T}} P_t \bar{x}_t$ which implies that $(x_t)_t$ is an optimal solution.

Since Lemma 4.3 is called at most T-times and every step is evaluated in time $\mathcal{O}(T)$, the total time complexity is $\mathcal{O}(T^2)$. □

Algorithm 4.1 has quadratic complexity because the updates of sequences $(A_t)_t$ and $(B_t)_t$ take linear time. This complexity easily can be improved using a binary tree. The basic idea is that values of $(A_t)_t$ and $(B_t)_t$ are handled independently by two separate balanced binary trees (see e.g. Cormen et al. [4]). In the following, we only describe the tree for the sequence $(A_t)_t$ since the tree for sequence $(B_t)_t$ can be handled analogously. Let $A^d_t = A_t - A_{t-1}$. The leaves of the tree store the value A^d_t for the time intervals $t \in \mathcal{T}$. Time intervals are assigned to leaves in a sorted way where the left subtree of every inner vertex contains earlier time intervals than the right subtree. Every inner vertex of the tree stores the sum of values of A^d_t for all leaves t in the subtree. Since the tree is constructed to be balanced, the length of every path from the root to a leaf is $\log_2(T) + \mathcal{O}(1)$. This binary tree is constructed in time $\mathcal{O}(T)$. It is a simple exercise to find out how values A_t and B_t are determined and how both trees are updated when Lemma 4.3 is applied. Both operations are performed in logarithmic time, so these binary trees improve the time complexity of Algorithm 4.1 to $\mathcal{O}(T \log T)$. We skip more details because next section presents even faster data structure.

5. Union-find

In this section we use the disjoint-set data structure of the union-find algorithm (see e.g. Cormen et al. [4]) to store and update values of sequences $(A_t)_t$ and $(B_t)_t$ to reduce the time complexity of the presented algorithm.

A disjoint-set data structure is a data structure that keeps track of a set of elements partitioned into a number of disjoint (non-overlapping) subsets. A union-find algorithm is an algorithm that performs two useful operations on such a data structure:

Find: Determine which subset a particular element is in. This can be used for determining if two elements are in the same subset.

Union: Join two subsets into a single subset.

Using a technique called path compression, both operation have amortized complexity $\mathcal{O}(\alpha(n))$ where α is the inverse of Ackermann function and n is the number of elements.

When Algorithm 4.1 applies Lemma 4.3, the two time intervals t_A and t_B need to be determined for a given $t^\star$. Since, the time intervals t_A are equal for all time intervals t having the same value of A_{t-1}, we use the disjoint-set data structures which partitions the set of time intervals $\mathcal{T}$ according to values A_{t-1}. Analogously, in order to determine time interval t_B for given time interval t, we use a similar disjoint-set data structures to partition the set of time intervals $\mathcal{T}$ according to values B_t. Note that values A_t and B_t are not stored in these data structures. These disjoint-set data structures only store the partitioning and time intervals t_A and t_B in every subset. However, this is sufficient information to determine whether $A_{t_1} = A_{t_2}$ and whether $B_{t_1} = B_{t_2}$ for given two time intervals t_1 and t_2.

In order to simplify further notation, let $\sim$ be a relation on the set of time intervals $\mathcal{T}$ such that $t_1 \sim t_2$ if $A_{t_1-1} = A_{t_2-1}$ and $B_{t_1} = B_{t_2}$ where $t_1, t_2 \in \mathcal{T}$. Observe that $\sim$ is an equivalence on $\mathcal{T}$ in which every

factor class contains a set of consecutive time intervals. Factor classes of the equivalence $\sim$ are called B-A-sets. During the algorithm, the relation $\sim$ is changing, but the only change in the relation $\sim$ is that some B-A-sets are united. In fact, one application of Lemma 4.3 leads to at most two unions: B-A-sets containing time intervals $t_A - 1$ and t_A may be united and B-A-sets containing time intervals $t_B - 1$ and t_B may be united. This naturally leads to a third usage of the disjoint-set data structure which is discussed below in more details. Since values of A_t for all time intervals t of one B-A-set S are equal, we denote this value A_S. Similarly, B_S denotes the B_t value of all for any time intervals t of B-A-set S.

A straightforward way to determine whether $A_{t^\star-1} < B_{t^\star}$ required in condition (13) would be to store the difference $D_t = B_t - A_{t-1}$ for every time interval $t \in \mathcal{T}$ but updating these differences after every application of Lemma 4.3 is too time consuming. Since all time intervals t in one B-A-set have the same value of the difference D_t, this difference D_t can be stored in every B-A-set. This approach is already insufficient to reach the desired time complexity. Indeed, updating the difference D_t becomes expensive when there are many B-A-sets $S_1, S_2, \ldots, S_n$ between time intervals t_B and $t^\star$ since differences of all those B-A-sets need to be decreased by one. Observe from the definition of t_B that $A_{S_1} < A_{S_2} < \cdots < A_{S_n}$ and $B_{S_1} = B_{S_2} = \cdots = B_{S_n}$ in this case; see Fig. 2. Similarly, there can be many steps in the sequence $(B_t)_t$ without any step in the sequence $(A_t)_t$ for t between $t^\star$ and t_A. Note that the difference D_t is not used in Algorithm 4.1, since it is only necessary to determine whether $D_t = 0$ or $D_t > 0$. Therefore, the difference D_t is actually stored only in some selected factor classes that form some kind of local minimals of the sequence $(D_t)_t$ which is formally explained below.

Let us consider one B-A-set S and let S^- and S^+ be the preceding and the following B-A-set, respectively. Observe that if $A_{S^-} = A_S$, then $B_{S^-} + 1 = B_S$ and therefore $D_S = D_{S^-} + 1$. Similarly, if $B_S = B_{S^+}$, then $A_S + 1 = A_{S^+}$ and therefore $D_S = D_{S^+} + 1$. In both cases it is unnecessary to store the value D_S since the facts that $D_{S^-} \geq 0$ and $D_{S^+} \geq 0$ imply that D_S is positive. The difference D_S is stored in a B-A-set S if and only if

$$A_{S^-} < A_S \text{ and } B_S < B_{S^+}. \tag{14}$$

In summary, we use three disjoint-set data structures. In order to determine whether (13) is satisfied, it suffices to find B-A-set storing time interval $t^\star$. The condition $A_{t^\star-1} < B_{t^\star}$ holds if and only if the difference $D_{t^\star}$ is unsaved or the stored value is positive. It remains to show how those data structures are updated when Lemma 4.3 is applied. Disjoint-set data structures factorizing by A_t and B_t are directly updated by the operation union.

Let S_1, S_2, S_3, S_4 and $S^\star$ be B-A-sets containing time intervals $t_B - 1$, t_B, $t_A - 1$, t_A and $t^\star$; respectively. Note that S_2, S_3 and $S^\star$ may be the same and observe that the difference D_t is changed only for time intervals t with $t_A \leq t < t_B$, where it decreases by one. Therefore, the evaluation of the condition (14) may change only for sets S_1, S_2, S_3 and S_4. Furthermore, $S^\star$ is the only B-A-set which can satisfy (14) and for which the stored difference $D_{S^\star}$ can change. The only possible changes in partitioning of time intervals $\mathcal{T}$ into B-A-sets are uniting S_1 and S_2 and uniting S_3 and S_4.

In the following, we only discuss updates of B-A-sets S_1 and S_2 since updates of B-A-sets S_3 and S_4 are analogous. Note that $B_{S_1} + 1 = B_{S_2}$ and new values according to Lemma 4.3 satisfy $B^\star_{S_1} = B^\star_{S_2} = B_{S_1}$. If $A_{S_1} < A_{S_2}$, then B-A-sets S_1 and S_2 are not united and the value of the difference D_{S_1} is deleted (if it has been stored). If $A_{S_1} = A_{S_2}$, then B-A-sets S_1 and S_2 are united. Observe that if the united set satisfies (14), then S_1 has satisfied (14) and the difference of the united set is the difference of S_1. All updates of the disjoint-set data structure of B-A-sets are summarized in Algorithm 5.1.

The time complexity of Algorithm 4.1 if it uses the disjoint-set data structures is $\mathcal{O}(T\,\alpha(T))$ where α is the inverse of Ackermann function since the number of operations union and find on the disjoint-set data structures in every iteration is upper bounded by a constant.

Algorithm 5.1: Update of the disjoint-set data structure for B-A-sets

Input: Time interval $t^\star$

Find time intervals t_A and t_B;
Find B-A-sets S_1, S_2, S_3, S_4 and $S^\star$ containing $t_B - 1$, t_B, $t_A - 1$, t_A and $t^\star$ respectively;
if $A_{S_1} = A_{S_2}$ **then**
    Union of sets S_1 and S_2 into a set S_{12};
    if S_{12} *satisfies* (14) **then**
        Set the difference $D_{S_{12}}$ to be the difference D_{S_1} before the last uniting;
if $B_{S_3} = B_{S_4}$ **then**
    Union of sets S_3 and S_4 into sets S_{34};
    if S_{34} *satisfies* (14) **then**
        Set the difference $D_{S_{34}}$ to be the difference D_{S_4} before the last uniting;
if $S^\star \neq S_2$ *and* $S^\star \neq S_3$ **then**
    Decrease the difference $D_{S^\star}$ by one;

6. Conclusion

This paper presents a $\mathcal{O}(T\,\alpha(T))$ algorithm for local heating problem.

In some practical cases, a valve can be used to control the heat flow to a buffer (e.g. district heating). This mathematically means that the constrain (3) is replaced by $0 \leq x_t \leq 1$. It should be possible to adopt our algorithm to this case, although some parts may become more technical.

References

[1] C. Aguilar, D.J. White, D.L. Ryan, Domestic water heating and water heater energy, consumption in Canada, Can. Build. Energy End-use Data Anal. Cent. 2 (2005).
[2] A. Molderink, V. Bakker, M.G.C. Bosman, J.L. Hurink, G.J.M. Smit, Management and control of domestic smart grid technology, IEEE Trans. Smart Grid 1 (2) (2010) 109–119.
[3] J. Fink, J.L. Hurink, Minimizing costs is easier than minimizing peaks when supplying the heat demand of a group of houses, European J. Oper. Res. 242 (2015) 644–650.
[4] T.H. Cormen, C.E. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2001.
[5] A. Drexl, A. Kimms, Lot sizing and scheduling-survey and extensions, European J. Oper. Res. 99 (2) (1997) 221–235.
[6] B. Karimi, S.M.T. Fatemi Ghomi, J.M. Wilson, The capacitated lot sizing problem: A review of models and algorithms, Omega 31 (5) (2003) 365–378.
[7] A. Sven, Inventory control, in: International Series in Operations Research and Management Science, vol. 90, Springer, 2006.
[8] N. Brahimi, S. Dauzere-Peres, N.M. Najid, A. Nordli, Single item lot sizing problems, European J. Oper. Res. 168 (1) (2006) 1–16.
[9] H.M. Wagner, T.M. Whitin, Dynamic version of the economic lot size model, Manage. Sci. 5 (1) (1958) 89–96.
[10] A. Federgruen, M. Tzur, A simple forward algorithm to solve general dynamic lot sizing models with n periods in 0 (n log n) or 0 (n) time, Manage. Sci. 37 (8) (1991) 909–925.
[11] M. Florian, J.K. Lenstra, A.H.G. Rinnooy Kan, Deterministic production planning: Algorithms and complexity, Manage. Sci. 26 (7) (1980) 669–679.
[12] G.R. Bitran, H.H. Yanasse, Computational complexity of the capacitated lot size problem, Manage. Sci. 28 (10) (1982) 1174–1186.
[13] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, European J. Oper. Res. 59 (3) (1992) 345–358.
[14] S.-H. Lin, N. Gertsch, J.R. Russell, A linear-time algorithm for finding optimal vehicle refueling policies, Oper. Res. Lett. 35 (3) (2007) 290–296.
[15] T.M. Sweda, D. Klabjan, Finding minimum-cost paths for electric vehicles, in: Electric Vehicle Conference (IEVC), 2012 IEEE International, 2012, pp. 1–4.
[16] S.-H. Lin, Finding optimal refueling policies: A dynamic programming approach, J. Comput. Sci. Colleges 23 (6) (2008) 272–279.

[17] R.P. van Leeuwen, J. Fink, J.B. de Wit, G.J.M. Smit, Thermal storage in a heat pump heated living room floor for urban district power balancing, effects on thermal comfort, energy loss and costs for residents, in: Smartgreens 2014, 2014.
[18] J. Fink, R.P. van Leeuwen, J.L. Hurink, G.J.M. Smit, Linear programming control of a group of heat pumps, Energy Sustainability Soc. 5 (33) (2015).