Planning of On/Off Devices with Minimum Run-times

Marco E. T. Gerards, Johann L. Hurink
Department of EEMCS, University of Twente
Enschede, the Netherlands

***Abstract*—To be able to reach objectives such as peak shaving or self-consumption within smart grids, it becomes increasingly important to control smart grid ready devices within households. This paper explores the planning of devices with minimal run-time constraints and constraints on total cumulative production (e.g., buffer level), such as heat pumps and electric vehicles. For such devices, a novel dynamic programming formulation is presented that makes it possible to find the optimal planning in $\mathcal{O}(N^2)$ time and space, where N is the number of time intervals. To evaluate the algorithm, we integrate it into an existing decentralized demand side management approach. This evaluation shows using simulations that a load curve within a neighborhood can be flattened this way.**

***Index Terms*—Optimal scheduling, smart grids.**

I. Introduction

Demand Side Management (DSM) projects typically assume that a Home Energy Management System (HEMS) is installed inside a customers home to control devices. Common objectives of these DSM projects are peak shaving to avoid losses and to defer grid investments, and increasing the amount of self-consumption of locally produced energy for economic and environmental reasons. To accomplish this, the HEMS steers devices within the house to meet the given objective. This paper considers planning-based DSM (see, e.g., [1]), a DSM approach wherein a planning for a forthcoming period (e.g., 24 hours ahead) is made for each device and used to steer the devices. The aim of this approach is to achieve an optimal trade off between the possible times where the flexibility of a device can be deployed. In Section II we give some more background and discuss related work on this topic, with a focus on approaches (such as [1]) that can be readily combined with the approach presented in this paper.

For planning-based DSM there are two options to plan the use of devices over a forthcoming period, namely using a centralized generic planning algorithm (e.g., a commercial MILP solver) for optimizing the planning for all devices in simultaneously, or using tailored algorithms for the planning of specific device classes and to use a decentralized approach to coordinate the planning of these devices. The first option of using a generic planning algorithm for general class of devices is commonly time consuming, and often not a viable option on embedded systems with low processing power such as the HEMS. In this paper we focus on the second option, and specifically we study the subclass of devices with a single mode of operation (i.e., on/off) and minimum run-time constraints. Furthermore, we consider constraints that specify, for each interval, the minimum and maximum time the device should have run up to this interval. This way we support, for example, heat producing devices (e.g., heat pumps) that have to guarantee a given comfort level inside the house, which can be translated to bounds on the fill level of an attached buffer. A detailed problem formulation of our optimization problem is given in Section III.

To solve this problem, we develop a dynamic programming approach that solves our problem with a time/space complexity of $\mathcal{O}(N^2)$ (see Section IV), while a straightforward DP formulation would yield $\mathcal{O}(N^3)$, where N is the number of time intervals. To ease the presentation, we use graphs to represent the state space of the dynamic program, and to visualize how the allowed state transitions enforce the minimum continuous run-time/off-time constraints, and ensure that the constraints on the total run-time are respected.

In Section V, we integrate our algorithm in an existing DSM approach. While our algorithm works for a single device, the profile steering approach is used to coordinate between many devices and is capable of using our algorithm as a subroutine. For a case of 121 houses together with a set of electric vehicles (we vary the number of vehicles) it is investigated how well the neighborhood load profile can be flattened. The results show that this combination of the DSM approach with the presented algorithm is a good fit and can be used to influence the load at the transformer over a complete day. Finally, Section VI presents the conclusions.

II. Background and related work

Decentralized DSM systems use steering signals to control the behavior of appliances within a house indirectly, opposed to Demand Response (DR) where the appliances are controlled by, for example, the Distribution System Operator (DSO) or an aggregator [2], [3]. Within DSM, two directions are visible: auction based systems and planning based systems. Where auction based systems focus on a balancing of supply and demand for the next time interval (an example is the PowerMatcher, see e.g., [4]), planning based DSM considers

This research was conducted within the DREAM project supported by STW (#11842) and Deutsche Bundesstiftung Umwelt (#30466).

the coordinated use of devices over several time intervals. As such for planning based DSM approaches planning algorithms for devices are needed. If one considers the joint optimization of all available devices, the resulting problems are in general (strongly) NP-hard [1], [5]. A common reaction to this is to formulate the problem as Mixed Integer Linear Programming (MILP) and use general solvers for this planning problem. However, such approaches are not suitable for the use on embedded device with low computational capabilities, and are therefore mainly used in simulations.

An alternative is to use specialized algorithms for the planning of specific devices or device classes and to use a heuristic algorithm to coordinate these individual plannings to meet a certain overall goal such as peak shaving for the group of devices. A simple way to do this is by iteratively adapting the steering signal that is sent to houses or (groups of) appliances (see, e.g., [1], [3]). The appliances in such an approach receive this steering signal and use specialized planning algorithms that use the received steering signal while respecting both the physical constraints of this device, and the user constraints regarding this device. An example of such a specialized algorithm is presented in [6] where an electric vehicle is scheduled to minimize costs (such as price and deviation from a received target profile) while respecting constraints such as the required state of charge, bounds on the charging power, arrival time of the car, and the charging deadline. This algorithm was used as added subroutine in the evaluation of the *profile steering* approach [1] to schedule a fleet of electric vehicles according to a received steering signal. Note, that for practical use of the sketched planning based DSM approach it is crucial that the planning algorithms for the devices are efficient and fast enough to be used in an embedded device with low computational capabilities.

The algorithms we propose in this paper fulfill the mentioned requirements and as such can be used as a planning subroutine in planning based DSM approaches. As mentioned, we focus on the optimal planning of on/off devices with minimum run-time constraints. In our evaluation in Section V, we use smart plugs (i.e., on/off devices) to control the charging of a fleet of electric vehicles with minimum continuous run-time constraints, applied within profile steering.

Another application of the approach developed in this paper is dispatching on/off devices with minimum continuous run-time constraints in an economically optimal way. This has been done in, e.g., [7], [8], to schedule the charging of (plug-in) electric vehicles to attain minimal charging costs. Our algorithm supports such monetary cost functions and can be applied in similar settings.

Optimization problems with minimum run-time constraints are commonly modeled as a MILP problem (see, e.g., [9]), which typically results in a high computational load when solved using general solvers. Scheduling of on/off devices with minimum continuous run-time constraints are also mentioned in [9], wherein a group of on/off devices is scheduled. Our approach can be applied more generally, in the sense that it can be combined with DSM approaches such as [1], [3]. In this way, the on/off devices can easily be combined with other types of devices, whereas the approach in [9] is tailored for on/off devices.

In [10], also the implementation of an iterative DSM approach is discussed, together with a dynamic programming approach for devices with minimum continuous run-time/off-time constraints. Opposed to our approach, their approach gives an approximation and does not guarantee optimality.

The Unit Commitment Problem (e.g., [11]–[13]) is in some respects similar to the considered problem, namely it also considers minimum continuous run-time/off-time constraints. However, in addition our problem imposes a minimum and maximum total on/off time constraint for each individual interval (e.g., to keep a buffer fill level within certain bounds). Furthermore, in most cases where the unit commitment problem is considered, a calculation time of several minutes on a regular computer is required, and therefore MILP formulations using commercial solvers may be used. In contrast, in our setting the developed algorithm has to be executed on a HEMS, a small embedded system or microcontroller for which less to no MILP solvers are available, and the calculation must be done in microseconds instead of minutes. This makes it impossible to directly use the existing research on unit commitment to solve the problem that we study. Since we need a planning for just a single device, opposed to many in the unit commitment problem, we can use a tailored algorithm that finds the solution efficiently.

Finally, we want to mention that the developed dynamic programming formulation is inspired by the approach in [14], where a problem is treated from another problem domain (green computing), which has some similarities to the problem considered in this paper.

III. Problem statement

In planning-based demand side management systems, devices are planned such that some global objective is met. In this planning problem, the planning horizon is subdivided into time intervals. In this paper, we denote the number of such time intervals with N, and we assume that all intervals have the same length. For each interval, we need to make a decision and for the devices considered in this paper, a device is either turned on or off in an interval $n \in \{1, \dots, N\}$, denoted by $x(n) = 1$ and $x(n) = 0$ respectively.

The planning objective is commonly described using a cost function, which expresses how well a schedule meets the given objective (i.e., costs do not have to be monetary). For example, if the energy price in time interval n is A_n (costs per kWh), and if $p_n(x_n)$ gives the power consumption (in watt) for the off/on state x_n, the costs may be specified by the function $f_n^{\text{price}}(x_n) = cA_n p_n(x_n)$, where c is used as a constant for conversion of the units, taking into account the chosen interval length. In a similar way cost functions can be defined to express, e.g., the self-consumption of a household (i.e., a household prefers its own solar energy above import from the grid), profile flattening/peak shaving (e.g., $f_n(x_n) = p_n(x_n)^2$), or minimizing the distance to some target profile $\vec{y} = [y_1, \dots, y_N]$ that is defined by a DSM system (i.e.,

$f_n(x_n) = (y_n - p_n(x_n))^2$. Note, that in these cases costs are not monetary. To obtain the costs for all N intervals together we need a sum (i.e., a separable function). Our goal is to find a planning for the device such that the total costs are minimized, i.e., to find a vector $\vec{x} = [x_1, \dots, x_N]$ that indicates for each time interval if the device is on or off and that minimizes the separable (global) cost function:

$$\mathcal{C}(\vec{x}) = \sum_{n=1}^{N} f_n(x_n).$$

At this point we want to stress that our solution method does not impose any restrictions on the functions $f_1, \dots, f_N : \{0,1\} \to \mathbb{R}$.

However, the device cannot be turned on/off in an arbitrary way to minimize the costs $\mathcal{C}(\vec{x})$, but device constraints must be respected. In this paper, we consider two types of device constraints. The first type is a total run-time constraint: before the end of interval N, the device must have been on for (in total) at least M_N intervals. Typically, devices that fill a buffer (heat pump, or EV battery charging) can be modeled this way.

In this situation, an optimal solution can be found easily by turning the device on in the M_N cheapest intervals. This results in a time complexity of $\mathcal{O}(N)$ [15]. A more general set of constraints results if we introduce a total run-time constraint M_n for each intermediate interval n, which is used to express that in interval n the buffer has to be filled according to some demand that is expected for interval n, or to model a finite buffer capacity. This case is treated later.

The second type of constraint we consider is a minimum continuous run-time/off-time constraint. This constraint enforces that when the device is turned on, it must remain on for at least R_1 intervals, and when it is turned off it must remain off for at least R_0 intervals. Such constraints are, for example, enforced by several thermal devices [9]. To ease the notation, we define the set of solutions that respect the minimum continuous run-time constraints:

$$\mathcal{X}_1 := \{\vec{x} \in \{0,1\}^N \mid \forall n \in \{1, \dots, N\} : x_n = 1 \Rightarrow \exists i : i \leq n \leq i + R_1 - 1 \wedge x_i = \dots = x_{i+R_1-1} = 1\},$$

which expresses that whenever $x_n = 1$, it is contained in a consecutive sequence of "on intervals" of at least length R_1. In a similar fashion, a set of solutions that respect minimum continuous off-time constraints is defined:

$$\mathcal{X}_0 := \{\vec{x} \in \{0,1\}^N \mid \forall n \in \{1, \dots, N\} : x_n = 0 \Rightarrow \exists i : i \leq n \leq i + R_0 - 1 \wedge x_i = \dots = x_{i+R_0-1} = 0\}.$$

If we combine the above constraints with the objective function, we get the following optimization problem, which we call the Simple Minimum Run-Time (SMR) problem.

$$\min_{\vec{x} \in \mathcal{X}_0 \cap \mathcal{X}_1} \sum_{n=1}^{N} f_n(x_n), \quad \text{subject to} \quad \sum_{n=1}^{N} x_n = M_N. \tag{SMR}$$

This problem is solved in Section IV. In the same section, we extend our approach to take into account a total run-time constraint for each interval. This optimization problem, called Minimum Run-Time (MR), is formulated as:

$$\min_{x_1,\dots,x_N \in \{0,1\}} \sum_{n=1}^{N} f_n(x_n), \quad \text{subject to} \quad L_k \leq \sum_{n=1}^{k} x_n \leq U_k, \quad \text{for } k \in \{1, \dots, N\}, \tag{MR}$$

where L_k and U_k place bounds on the cumulative on time of the device.

IV. Solution approach

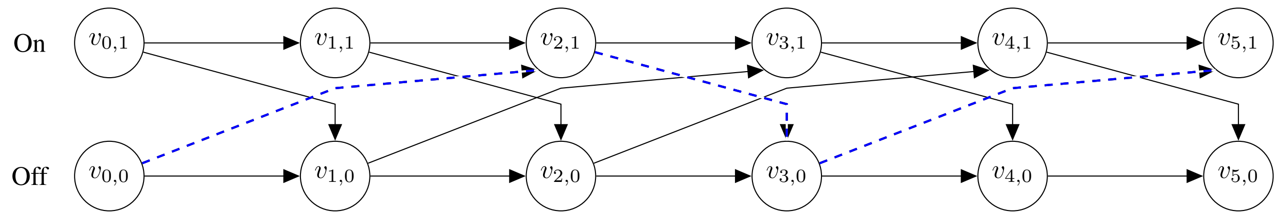
To solve both (SMR) and (MR), we use dynamic programming. The dynamic program determines for each possible state the cheapest way to get in this state. More precisely, we represent states as a triple $(n, a, m) \in \{1, \dots, N\} \times \{0,1\} \times \{1, \dots, N\}$, where n is the interval, a is the on/off state (0/1) of the device in this interval, and m is the total on time of the device up to this interval. For each state (n, a, m), we define a mapping $T'(n, a, m) = (n', a', m', c')$, where (n', a', m') is the preceding state (i.e., $n' < n$) on the shortest path from the initial state to (n, a, m) and c' is the lowest cost. Hence, this (implicitly) gives the path that leads to the lowest costs, by backtracking from the desired end state.

To ease the presentation and analysis, we give a graph representation of this DP wherein the tuples of intervals and on/off state are mapped on vertices of the graph (i.e., the tuples $(n, a) \in \{1, \dots, N\} \times \{0, 1\}$). Section IV-A discusses this graph and how a path in this graph corresponds to a solution. To ensure that the minimum continuous run-time/off-time constraints are met, we construct the graph such that all paths in the graph correspond to a solution that respects these constraints (Section IV-B). More precisely, in Section IV-C we construct a mapping $T(v, m) = (v', c)$ that implicitly gives the lowest cost (c) path (through v') with a total on-time m to a vertex v. By backtracking from the desired state (i.e., run-time M_N in interval N), we obtain the optimal solution with a complexity of $\mathcal{O}(N^2)$ for both space and time. To respect the additional constraints of (MR), Section IV-D encodes the minimum and maximum run-time constraints (commonly user or capacity constraints) by assigning infinite costs to paths that do not respect these constraints.

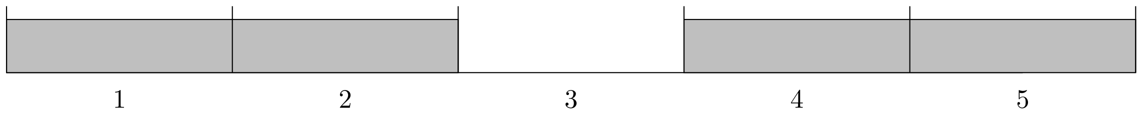
A. Graph representation

To make it easier to reason about the (SMR) optimization problem, we introduce a graph that depicts the decisions that can be made. Let $V = \{v_{0,1}, \dots, v_{0,N}, v_{1,1}, \dots, v_{1,N}\}$ be the set of vertices in this graph. Before we discuss how the other ingredients of these graphs are constructed, we describe how a path from $v_{0,1}$ to either $v_{0,N}$ or $v_{1,N}$ within this graph is related to a solution $\vec{x}$. Consider a path p with L vertices starting at vertex $v_{0,1}$ and ending at $v_{0,N}$ or $v_{1,N}$:

$$p = v_{a_0,k_0} v_{a_1,k_1} \dots v_{a_N,k_L},$$



(a) Graph depicting the minimum continuous run-time/off-time constraints.



(b) On/off schedule of device corresponding with the dashed blue path from Fig. 1a.

Fig. 1: Example of a graph with the minimum continuous run-time/off-time constraint, with the corresponding schedule for a single path. The graph nodes from Fig. 1a are aligned to the intervals from Fig. 1b.

with $0 = k_0 < k_1 < k_2 < \cdots < k_L = N$. In the above path, the indices k_i correspond to intervals up to where the device is switched on/off as is indicated by a_i. More precisely, the solution $\vec{x}$ corresponding to a path p can be constructed using the following transformation. For each edge $v_{a_j,k_j} v_{a_{j+1},k_{j+1}}$ we set $x_{k_j+1} = \cdots = x_{k_{j+1}} = a_{j+1}$. The following example illustrates this translation.

Example 1 (Correspondence between a path and a feasible solution)**.** *Fig. 1a shows an example of a graph that corresponds to* 4 *intervals. The vertices* $v_{0,0}$ *and* $v_{0,1}$ *are dummy nodes that ease the notation.*

The path $p = v_{0,0}v_{2,1}v_{3,0}v_{5,1}$ *is the dashed blue path indicated in the graph of Fig. 1a. Using the transformation described above we find the corresponding solution, which turns on the device in the first two intervals, turns the device off in interval three and turns the device on in intervals four and five (depicted in Fig. 1b).* □

B. Feasible paths

In the following, we describe how to construct a graph such that any path from vertex $v_{0,0}$ to either vertex $v_{N,0}$ or vertex $v_{N,1}$ corresponds (using the procedure discussed above) to a vector $\vec{x} \in \mathcal{X}_0 \cap \mathcal{X}_1$, i.e., only solutions that respect the minimum continuous run-time/off-time constraints correspond to paths in our graph.

To construct such a graph, we first consider a transition $v_{k_i,a}v_{k_{i+1},a}$, which means that when the device is in some state, it remains in this state. Such transitions always respect the minimum continuous run-time/off-time constraints, if we assume that the path up to $v_{k_i,a}$ was feasible. Hence, we can add all edges of the form $v_{k_i,a}v_{k_{i+1},a}$ without problems.

To switch from on to off, or from off to on, we (only) add edges that guarantee to respect the minimum continuous run-time constraints. This leads to edges in the form of $v_{k_i,0}v_{k_{i+R_1},1}$, which describe a transition from the off state to the on state, and guarantees that the device remains on for R_1 intervals. Since such an on interval can be extended using the edge in the form $v_{k_i,1}v_{k_{i+1},1}$, any feasible on period can be modeled. Using the same line of thought, we can also model off to on transitions using the edges that respect the minimum continuous off-time constraints, namely those of the form $v_{k_i,1}v_{k_{i+R_0},0}$.

Summarizing, the set of edges E contains all edges that describe remaining in the same state for one extra time interval, a transition from off to on and remaining on for R_1 intervals, and a transition from on to off and remaining off for R_0 intervals. Using these edges, any path from vertex $v_{0,0}$ to either vertex $v_{0,N}$ or vertex $v_{1,N}$ corresponds (using the transformation from Section IV-A) to a vector $\vec{x} \in \mathcal{X}_0 \cap \mathcal{X}_1$ and vice versa.

Example 2 (Minimum continuous run-time/off-time constraints (continued from Example 1))**.** *For* $N = 5$*, a minimum continuous run-time constraint of two intervals (*$R_1 = 2$*), and a minimum continuous off-time constraint of one interval (*$R_0 = 1$*) the graph with all feasible paths is given in Fig. 1a. By construction all paths from* v_{0,a_0} *to* v_{N,a_N} *(with* $a_0, a_N \in \{0, 1\}$*) correspond to a feasible schedule (e.g., the schedule in Fig. 1b that corresponds to the dashed blue path is feasible).* □

The next section describes how costs can be added to the graph and how cost minimization can be combined with the constraint that the total on time of a device is within a certain range for each interval.

C. Cost minimization

Each edge $e \in E$ represents a decision that is made for a set of consecutive intervals. For example, an edge $e = v_{k,0}v_{\ell,1}$ models $x_{k+1} = \cdots = x_\ell = 1$. Based on this, we introduce a cost function $F(e)$ defined as

$$F(v_{k,1-a}v_{\ell,a}) = \sum_{i=k+1}^{\ell} f_i(a),$$

where $a \in \{0, 1\}$.

To make sure that we only consider paths that correspond to feasible solutions of (SMR), we construct a representation of all feasible (sub)paths starting in $v_{0,0}$. In this step we already consider the costs, such that we can prune inefficient paths. For each node $v \in V$ and total run-time of m time intervals, we have a mapping $T(v, m) = (v', c) \in V \times \mathbb{R}$ that (implicitly) describes a path from vertex $v_{0,0}$ to v with a total on-time of m time intervals and minimal costs whenever such a path exists (otherwise we set $c = \infty$). Here, c specifies the costs of the path, and v' is the preceding vertex on this path. It is easy to see that using this mapping, the path we get by backtracking from either $(v_{N,0}, M_N)$ or $(v_{N,1}, M_N)$—the one with the lowest c—corresponds to the optimal solution to (SMR) using the transformation from Section IV-A.

The only thing that remains is the construction of the mapping $T(v, m) = (v', c)$ that supports this backtracking approach. This can be described using a table that can be constructed by a simple forward scan over all possible combinations of v and m, and evaluating all possible edges leaving this state. The resulting method is summarized in Algorithm 1.

Summarizing, to find the optimal solution to (SMR) we can proceed as follows:

1) Use Algorithm 1 to construct the table T.
2) Backtrack from $(v_{N,0}, M_n)$ or $(v_{N,1}, M_n)$ and reconstruct the cheapest path.
3) Translate this path to an optimal solution $\vec{x}$ using the transformation from Section IV-A.

Since the table contains $\mathcal{O}(N^2)$ entries, the space complexity of this approach is $\mathcal{O}(N^2)$. The time complexity is $\mathcal{O}(N^2)$, since each table entry can be calculated in $\mathcal{O}(1)$ time. In practical implementations, calculation time may be saved for many problem instances by calculating the values in T for required values only, e.g., by using a recursive function. A recursive approach prunes the unreachable states, e.g., the states that cannot be reached because M_N is very low or high.

Algorithm 1 Create the table T

```
$T(v, m) := (v, \infty)$ for all $(v, m) \in V \times \{1, \dots, M\}$
for $n = 0$ to $N - 1$ do
  for $m = 0$ to $n$ do
    ▷ Consider transitions from the state $(v_{n,0}, m)$
    $(v', c') = T(v_{n,0}, m)$
    if $n+R_1 \leq N$ and $c'+F(v_{n,0}v_{n+R_1,1}) < F(v_{n+R_1,1})$ then
      $T(v_{n+R_1,1}, m+R_1) := (v_{n,0}, c' + F(v_{n,0}v_{n+R_1,1}))$
    end if
    if $c' + F(v_{n,0}v_{n+1,0}) < F(v_{n+1,0})$ then
      $T(v_{n+1,0}, m) := (v_{n,0}, c' + F(v_{n,0}v_{n+1,0}))$
    end if
    ▷ Consider transitions from the state $(v_{n,1}, m)$
    $(v', c') = T(v_{n,1}, m)$
    if $n+R_0 \leq N$ and $c'+F(v_{n,1}v_{n+R_0,0}) < F(v_{n+R_0,0})$ then
      $T(v_{n+R_0,0}, m) := (v_{n,1}, c' + F(v_{n,1}v_{n+R_0,0}))$
    end if
    if $c' + F(v_{n,1}v_{n+1,1}) < F(v_{n+1,1})$ then
      $T(v_{n+1,1}, m+1) := (v_{n,1}, c' + F(v_{n,1}v_{n+1,1}))$
    end if
  end for
end for
```

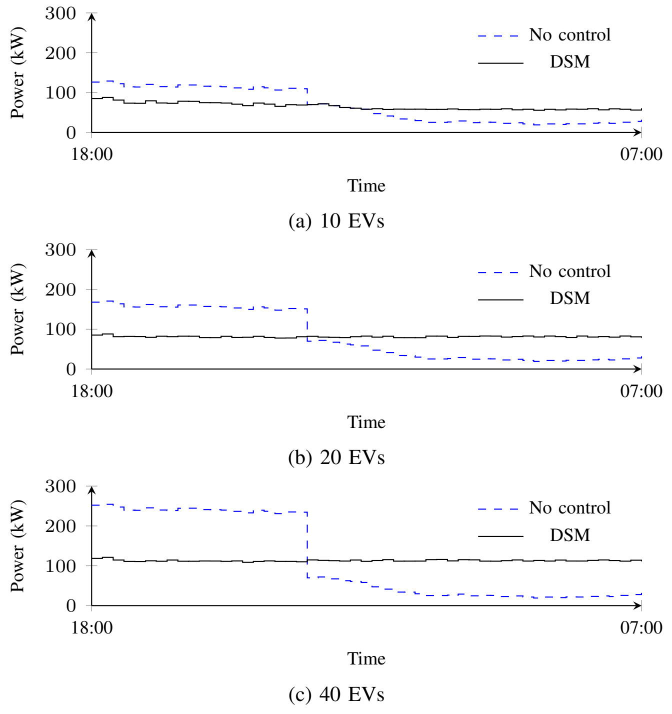


Fig. 2: Total power at transformer (incl. losses)

D. Generalization

To solve the Minimum Run-Time problem (MR), only a minor extension to Algorithm 1 is required. The algorithm calculates for each vertex $v_{n,a}$ the cheapest path to this vertex, and stores the costs and predecessor vertex in this path. As for an interval k we have to fulfill the constraint $L_k \leq \sum_{n=1}^{k} x_n \leq U_k$, we only need to consider the states $(v_{k,a}, L_k), \dots, (v_{k,a}, U_k)$ (for $a = \{0, 1\}$). This implies that the inner 'for'-loop of Algorithm 1 should be skipped when $m < L_n$ or $m > U_n$. Hence, such additional restrictions on the permitted state space may even speed up the algorithm.

V. Evaluation

The device planning algorithm developed in this paper was designed with integration into existing DSM methodologies in mind. Each HEM executes the device planning algorithm from this paper to function as subroutine within such a methodology. In this evaluation, we combine our algorithm with the *profile steering* approach from [1] to study if the algorithm can be applied in the setting it was designed for.

The profile steering approach iteratively requests improvements in the form of desired power profiles and aims to converge toward a given profile at the transformer; in our case a flat profile. The algorithm from Section IV can be used to minimize the distance toward such desired profiles, and be used to create a planning based on such requests, hence it can be directly used within the approach from [1].

In this evaluation we use detailed grid information of a Dutch LV grid that consists of 121 houses combined with household measurement data that is representative for the region. We use this to calculate the power (incl. losses) at the transformer. This given grid is extended with appliances that are controlled by our algorithm, and we use the profile steering algorithm to create steering signals such that the power profile as observed at the transformer becomes as flat as possible. Here, the power profile contains the controllable device as well as the given load profile of the 121 houses.

In our evaluation we use Electric Vehicles (EVs) as controllable appliances. Where in experiments with fast chargers often the possibility of adapting the charging power is used, for home charging installations that are currently in the field, the controllability of the charging is often only in switching on or off the charging (e.g., in a joint project with industry the control of charging the EV was realized by a smart plug that can handle such loads; see [16] for a description of the project). In order to reduce the stress on the equipment and to prevent damage, a minimum continuous run-time is desirable and in this evaluation it is set to 60 minutes. While in a practical setting a lower minimum run-time may be acceptable, a rather long minimum run-time restricts the planning freedom and demonstrates the capabilities of the algorithms since it creates more difficult circumstances. Note, that also the restriction to on/off control, increases the difficulty for the control because of a relatively low flexibility.

In this evaluation, we randomly distribute K EVs over the 121 houses, whereby the values $K \in \{10, 20, 40\}$ are used. To study the operation under extreme load, we schedule all EVs to arrive simultaneously at 18:00, and set the deadline to 07:00 the next morning. We use time intervals of 15 minutes each, hence there are in total 52 charging intervals. When the switch is turned on, each car charges at 4 kW, and in total it should charge 20 kWh within the charging horizon (i.e., it charges 20 intervals of the in total 52 intervals).

Fig. 2c shows that greedy charging upon arrival creates a peak of approximately 250 kW when the neighborhood contains 40 EVs. In contrast, the profile steering DSM approach in combination with our planning algorithm removes the peak by flattening the load profile. Fig. 2a demonstrates the case with 10 EVs, and shows that a DSM implementation can apply our algorithm for on/off devices to flatten the load profile. When more EVs are used there is more freedom, and as a result the load profile becomes even flatter (see Fig. 2b and Fig. 2c). Hence, the profile steering algorithm, using the presented algorithm as a subroutine, avoids charging at the peak and schedules the EVs to flatten the load curve, while taking into account the minimum run-time constraint.

VI. Conclusions

Many DSM and economic dispatch approaches control devices by some steering signal. This paper shows how to plan on/off devices optimally for any separable steering signal. More precisely, it presents a dynamic programming algorithm that optimally schedules on/off devices with minimum continuous run-times/off-times, and total run-time constraints for each individual time interval. In our algorithm, the permitted state space is constructed and can be explicitly represented in $\mathcal{O}(N^2)$ space. Subsequently, the optimal solution can be found by backtracking from the desired state in the final interval. This yields an approach that runs in $\mathcal{O}(N^2)$ time, and execution time improvements can be achieved using a recursive approach that constructs the state space lazily.

The algorithm is evaluated in a decentralized DSM context, wherein it is used as a subroutine to control the charging of electric vehicles. Our evaluation shows that the algorithm naturally supplements the existing DSM approaches, and can be used to influence the load curve.

In future work, we plan to extend this approach to take uncertainty in both the steering signal (e.g., matching unknown prices or profiles) and constraints (e.g., producing enough to meet unknown heat consumption) into account in a robust way.

References

[1] M. E. T. Gerards, H. A. Toersche, G. Hoogsteen, T. van der Klauw, J. L. Hurink, and G. J. M. Smit, "Demand side management using profile steering," in *PowerTech, 2015 IEEE Eindhoven*, June 2015, pp. 457 759:1–457 759:6.

[2] P. Siano, "Demand response and smart grids–a survey," *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 461–478, 2014.

[3] A. Molderink, V. Bakker, J. L. Hurink, and G. J. M. Smit, "On indirect controlled cost function based DSM strategies," in *Proceedings of IEEE PowerTech (POWERTECH), Grenoble*, June 2013, pp. 2236–2241.

[4] K. Kok, "The PowerMatcher: Smart coordination for the smart electricity grid," Ph.D. dissertation, Vrije Universiteit Amsterdam and TNO, 2013.

[5] M. G. C. Bosman, V. Bakker, A. Molderink, J. L. Hurink, and G. J. M. Smit, "On the microCHP scheduling problem," in *Proceedings of the 3rd Global Conference on Power Control and Optimization PCO, 2010, Gold Coast, Australia*, ser. AIP Conference Proceedings, vol. 1239. Australia: PCO, February 2010, pp. 367–374.

[6] T. van der Klauw, M. E. T. Gerards, G. J. M. Smit, and J. L. Hurink, "Optimal scheduling of electrical vehicle charging under two types of steering signals," in *IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Istanbul, Turkey*, October 2014, pp. 122:1–122:6.

[7] M. Sarker, M. Ortega-Vazquez, and D. Kirschen, "Optimal coordination and scheduling of demand response via monetary incentives," *Smart Grid, IEEE Transactions on*, vol. 6, no. 3, pp. 1341–1352, May 2015.

[8] D. Wu, D. Aliprantis, and L. Ying, "Load scheduling and dispatch for aggregators of plug-in electric vehicles," *Smart Grid, IEEE Transactions on*, vol. 3, no. 1, pp. 368–376, March 2012.

[9] B. Biegel, P. Andersen, T. Pedersen, K. Nielsen, J. Stoustrup, and L. Hansen, "Smart grid dispatch strategy for ON/OFF demand-side devices," in *Control Conference (ECC), 2013 European*, July 2013, pp. 2541–2548.

[10] H. Toersche, J. Hurink, and M. Konsman, "Energy management with TRIANA on FPAI," in *PowerTech, 2015 IEEE Eindhoven*, June 2015, pp. 462 751:1–462 751:6.

[11] M. Carrion and J. M. Arroyo, "A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1371–1378, Aug 2006.

[12] J. M. Arroyo and A. J. Conejo, "Optimal response of a thermal unit to an electricity spot market," *IEEE Transactions on Power Systems*, vol. 15, no. 3, pp. 1098–1104, Aug 2000.

[13] J. Ostrowski, M. F. Anjos, and A. Vannelli, "Tight mixed integer linear programming formulations for the unit commitment problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 39–46, Feb 2012.

[14] M. E. T. Gerards and J. Kuper, "Optimal DPM and DVFS for frame-based real-time systems," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 4, pp. 41:1–41:23, jan 2013.

[15] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection," *Journal of Computer and System Sciences*, vol. 7, no. 4, pp. 448–461, 1973.

[16] V. Bakker, A. Molderink, J. Hurink, G. Smit, S. Nykamp, and J. Reinelt, "Controlling and optimizing of energy streams in local buildings in a field test," in *Electricity Distribution (CIRED 2013), 22nd International Conference and Exhibition on*, June 2013, pp. 506:1–506:4.