2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)
Virtual, October 26-28, 2020

Profile Steering with Non-regular Time-Intervals

L.M. Bollen*, L.R. Heinsius*, M.L. Souilljee*, M. Boe*‡, G. Hoogsteen†, M.E.T. Gerards† J.L. Hurink†
*†*dept. of Electrical Engineering, Mathematics and Computer Science, University of Twente*
‡*dept. of Life Science & Technology, Saxion University of Applied Sciences*
Enschede, the Netherlands
*{l.m.bollen, l.r.heinsius, m.l.souilljee, m.boe}@student.utwente.nl
†{g.hoogsteen, m.e.t.gerards, j.l.hurink}@utwente.nl
‡m.boe@saxion.nl

***Abstract*—Demand Side Management (DSM) using the profile steering algorithm has succeeded to improve power quality and reduce energy losses. This paper presents an extension to profile steering called "IntervalMerge". IntervalMerge distinguishes itself from the original profile steering algorithm by using non-regular time interval lengths instead of static time interval lengths. This paper shows that by adjusting the profile steering algorithm, the execution time is reduced without significantly affecting the objective value. IntervalMerge is evaluated for different parameters and the achieved results show that by using different time interval lengths, the total execution time can be lowered up to 20%.**

***Index Terms*—Profile steering, smart grids, non-regular time intervals, optimization.**

I. Introduction

Existing electricity networks were designed when only few demanding loads and decentralized micro generators were present. The requirements for electricity networks changed in the last couple of years because of electrification and decentralized production. As a consequence, in electricity networks, peaks appear on different levels such as household, transformer and street level. To reduce these peaks, Demand Side Management (DSM) is used [1]. DSM is a broad research field for which different algorithms or incentives can be used based on the given application. Such an algorithm or incentive could, for example, be based on pricing signals [2].

Profile steering [3] is an iterative DSM algorithm that, given a desired profile (specified as power values), schedules usage of devices such that the aggregated power profile approaches the desired profile as closely as possible. These profiles consist of discrete time intervals that each represent the average power production/consumption over a given fixed amount of time. The distance between the desired profile and planned profile is expressed using the Euclidean distance, which is related to the losses in the electricity grid. To execute the profile steering algorithm, different parameters are of importance, e.g. the length of the time intervals used within the profile steering algorithm. Furthermore, to improve the quality of profile steering, a new planning should be made after a number of intervals have passed, i.e. a rolling horizon approach should be used.

To implement DSM, each participating household has to be equipped with a Home Energy Manager (HEM) which executes the DSM algorithm and controls the appliances. It is important that the HEM is a low power device to keep energy consumption low. Note, that this efficiency is important in the domain of DSM as potentially millions of devices need to be controlled. By reducing the computational time/complexity of the DSM algorithms, more applications of DSM using low-powered embedded systems become possible. In this context in [4] the effect of different window sizes for a rolling horizon planning algorithm are evaluated by breaking the planning into smaller pieces and increasing the frequency of re-planning. This resulted in a reduced computation time (between 10 to 100 times) without severely sacrificing the quality of the results.

The goal of this paper is to reduce the execution time of the profile steering algorithm by reducing the number of intervals. This is done by merging consecutive intervals into a single interval. The resulting reduction in computation time is measured using the execution time of the given simulation tool compared to the base case. The extension of the profile steering algorithm presented in this work is called IntervalMerge.

For executing the algorithms, in this research we use DEMKit, a platform for simulation and demonstrating control of smart grids [5]. The smart grid models consist of devices that model the behavior of physical devices. Devices can be controlled using an optional controller component, which contains algorithms for device-specific predictions and scheduling. Profile steering is one of the optimization algorithms already implemented as controller component.

To evaluate the results of IntervalMerge, a Pareto front is used to provide a trade-off between lowering the Euclidean distance versus reducing the execution time of the algorithm.

In the original profile steering algorithm, each planning step concerns a period of time divided into multiple intervals of constant time, e.g. a period of two days consisting of 192 intervals that each represent 15 minutes. In each profile steering iteration, each device creates a new schedule and a resulting power profile based on the given desired profile. We show that by reducing the number of intervals it is possible to significantly reduce the time required to execute one planning iteration, similar to [4].

The main contributions of this paper are as follows:
- The IntervalMerge algorithm using non-regular time intervals.
- Analysis of IntervalMerge for different parameters.

978-1-7281-7100-5/20/$31.00 ©2020 IEEE

The remainder of this paper is organized as follows: Section II describes the IntervalMerge algorithm. Followed by a description of device level optimization in Section III. The results achieved with the IntervalMerge algorithm are described in Section IV and in Section V conclusions are drawn, finally in Section VI future work is presented.

II. Algorithm

This section gives a brief introduction on the profile steering algorithm [3] and the changes made to the algorithm. Profile steering is an algorithm used in DSM to find a planning for all available controllable appliances such that the resulting load of all devices matches a desired profile $\vec{p}$ (in our case a flat profile). Profile steering uses time intervals of the same length. The difference between the original algorithm and the IntervalMerge algorithm presented in this work is mainly that IntervalMerge reduces the size of the desired profile and the consumption profile by using non-regular time intervals.

In profile steering each time interval represents a constant time t_{base} (for this research $t_{base} = 900s$). In contrast to this, each time interval represents t_{base} or a multiple of t_{base} in IntervalMerge. For this, a new parameter vector $\vec{\alpha}$, is introduced to describe the merging method. Every element α_n in $\vec{\alpha}$ represents the number of consecutive intervals to be merged into one interval.

This vector $\vec{\alpha}$ is used to reduce the size of the desired profile $\vec{p}_m$ resulting in reduced desired profile $\vec{q}_m$. The reduced desired profile is then used as input for the appliances that now also return as a result the reduced profile $\vec{\hat{w}}_m$. Afterward, IntervalMerge reconstructs based on $\vec{\hat{w}}_m$ a vector of the original size $\vec{\hat{x}}_m$ using the interval vector $\vec{\alpha}$.

In order to build the original size result, the operation is inverted, using the same $\vec{\alpha}$ and $\vec{\hat{x}}_m$. I.e. the power values are taken from the intervals of the reduced profiles to which the given intervals of the original vector belongs. A description of the IntervalMerge algorithm is presented in Algorithm 1. The grey part shows the changes with respect the original profile steering algorithm.

Regular Time-intervals

15 minutes	15 minutes	15 minutes	15 minutes	15 minutes
15 minutes	30 minutes		30 minutes	

Non-regular Time-intervals

Fig. 1. Example of using non-regular time-intervals versus regular time-intervals.

The values $q_{m,c}$ of q_m and values $w_{m,c}$ of w_m are calculated using (1) and (2) respectively. As example, given the desired profile $\vec{p}_m = \begin{bmatrix}321 & 230 & 258 & 240 & 566\end{bmatrix}$ consisting of five intervals and $\vec{\alpha} = \begin{bmatrix}1 & 2 & 2\end{bmatrix}$, the desired profile can be reduced to $\vec{q}_m = \begin{bmatrix}321 & \frac{230+258}{2} & \frac{240+566}{2}\end{bmatrix} = \begin{bmatrix}321 & 244 & 403\end{bmatrix}$ consisting of three intervals (see Fig. 1).

$$I(c) = 1 + \sum_{n=1}^{c-1} \alpha_n \tag{1}$$

Algorithm 1: IntervalMerge algorithm.

Determine interval vector $\vec{\alpha}$.
Request each appliance $m \in \{1, \ldots, M\}$ to minimize $||\vec{x}_m||_2$.
$\vec{x} := \sum_{m=1}^{M} \vec{x}_m$ {Total household consumption}
repeat
  $\vec{d} := \vec{x} - \vec{p}$ {Difference vector}
  for $m \in \{1, \ldots, M\}$ **do**
    $\vec{p}_m := \vec{x}_m - \vec{d}$
    Reduce desired profile $\vec{p}_m$ to $\vec{q}_m$and consumption profile $\vec{x}_m$ to $\vec{w}_m$ using $\vec{\alpha}$ (see (1), (2) and (3)).
    For appliance m, find a planning $\vec{\hat{w}}_m$ that minimizes $||\vec{\hat{w}}_m - \vec{q}_m||_2$.
    Build $\vec{\hat{x}}_m$ from $\vec{\hat{w}}_m$ using $\vec{\alpha}$ (inverse of (3)).
    $e_m = ||\vec{x}_m - \vec{p}_m||_2 - ||\vec{\hat{x}}_m - \vec{p}_m||_2$ {Relevant flexibility of appliance m}
  end for
  Find appliance m with the highest contribution e_m
  $\vec{x} := \vec{x} - \vec{x}_m + \vec{\hat{x}}_m$ {Update the total consumption}
  $\vec{x}_m = \vec{\hat{x}}_m$ {Update the profile of the appliance m}
until $e_m < \epsilon$ {Repeat as long as there is sufficient progress}

$$\vec{q}_{m,c} = \frac{1}{\alpha_c} \sum_{n=I(c)}^{I(c)+\alpha_c-1} p_{m,n} \tag{2}$$

$$\vec{w}_{m,c} = \frac{1}{\alpha_c} \sum_{n=I(c)}^{I(c)+\alpha_c-1} x_{m,n} \tag{3}$$

III. Device Level Optimization

Each device optimizes its own energy profile based on the received desired profile from the profile steering algorithm. The objective here is to minimize the Euclidean distance between the communicated desired profile and its own power profile.

With the IntervalMerge variant of profile steering, also the device level optimization algorithms need to be revised to create an optimal schedule under the modified desired profiles. For the class of static devices, merging the profile (either a prediction or the actual input data) is sufficient. To this end, we take the average power consumption over the merged intervals, similar to how we merge intervals in the IntervalMerge variant of profile steering. However, the other device classes require more modifications. In this work we restrict the modifications to the optimization of batteries, or more generically speaking: buffers.

A. Buffer optimization

As a basis, the discrete buffer optimization algorithm by van der Klauw [6, Algorithm 5.2] is used. For each given time interval, this algorithm splits the possible charging/discharging steps for the buffer up in linear pieces for each discrete power option z that the buffer supports,

e.g. $z = [-2000 \ -1000 \ 0 \ 1000 \ 2000]$. The cost of selecting power option z^{j+1} over z^j in interval t is given by piece s_t^j (4). In our case, the cost of selecting the next possible power option is related to the change in the objective value and is calculated for the original profile steering algorithm.

$$s_t^j := \frac{||p_{m,t} - z_t^{j+1}||_2 - ||p_{m,t} - z_t^j||_2}{z_t^{j+1} - z_t^j} \tag{4}$$

Initially, all pieces s_t^1 are added to a set S and sorted non-decreasingly and the planned power $w_{m,t}$ for a device m is set to z^1. The charging/discharging is determined in an iterative process by greedily selecting the piece with the lowest cost, i.e. the first element in S. The corresponding s_t^j is removed from S. The planned power for t is updated by setting $w_{m,t} = z_t^{j+1}$. In each stage, for each interval, the next possible (feasible) piece s_t^{j+1} (increase in power consumption) is inserted into S and sorted non-decreasingly again.

Additional bookkeeping of the state of charge V_t for each interval is needed (see [6]) to ensure that choosing s_t^j does not result in an infeasible solution. This mean that in case of a buffer, the resulting state of charge may not become negative or surpass the capacity C of the buffer. Furthermore, V_t depends on initial state of charge C_0. For a given interval t, the state of charge is given as $V_t = C_0 + \sum_{t'=1}^{t} w_{m,t'}$.

With the IntervalMerge algorithm, the original algorithm [6, Algorithm 5.2] needs to be modified to accommodate the number intervals which get merged for the new interval t, denoted by α_t. The state of charge at interval t therefore becomes $V_t = C_0 + \sum_{t'=1}^{t} w_{m,t'}\alpha_{t'}$. Furthermore, the maximum feasible power option σ, which ensures that a chosen option is feasible, needs to be adapted by dividing it through α_t. Lastly, the calculation of the cost of selecting the next power value must be rescaled (5).

$$s_t^j := \alpha_t \frac{||p_{m,t} - z_t^{j+1}||_2 - ||p_{m,t} - z_t^j||_2}{z_t^{j+1} - z_t^j} \tag{5}$$

The complexity of the algorithm, $O(TM)$, remains the same, in which T now denotes the number of time intervals and M the number of pieces. However, note that IntervalMerge reduces the number of time intervals T and therefore it can be expected that the computation time reduces linearly with the number of intervals merged. For more details about the algorithm we refer the reader to [6, Algorithm 5.2].

IV. Evaluation

This section evaluates the performance of the IntervalMerge algorithm presented in Section II. For this IntervalMerge has been implemented in DEMKit [5] and two tests have been performed. The first test evaluates speedup, while ignoring possible loss in quality of the results and the performance loss during upscaling. The second test studies the performance loss versus processing time gain for a static scenario with different interval merge parameters.

All test cases are performed on a household level and include the following devices: batteries, baseload and photovoltaic (PV) panel(s). The values for the baseload and PV are increased respectively with an increasing number of batteries. To remove external influences on the computational times as much as possible it is vital to create a constant and representable execution time. Therefore DEMKit is modified by:

- Removing unnecessary printing to the terminal.
- Removing logging to external programs such as InfluxDB.

All considered execution times in the evaluation are average execution of 50 times where all outliers are removed beforehand. Note that a rolling horizon is used which schedules 48 hours ahead and a new schedule is made each 24 house.

A. Scalability of IntervalMerge

The scalability of IntervalMerge is evaluated by applying four different parameter settings to one test case. These settings are:

- Original algorithm: the original algorithm presented in [3] consisting of 192 time-intervals.
- IntervalMerge, 161/192: the intervalMerge algorithm reduced to 161 time-intervals
- IntervalMerge, 129/192: the intervalMerge algorithm reduced to 129 time-intervals
- IntervalMerge, 97/192: the intervalMerge algorithm reduced to 97 time-intervals

Fig. 2 shows a performance comparison for the original profile steering algorithm and IntervalMerge with three different settings and with increasing number of batteries. Notice that the execution time increases approximately linearly with the number of batteries.

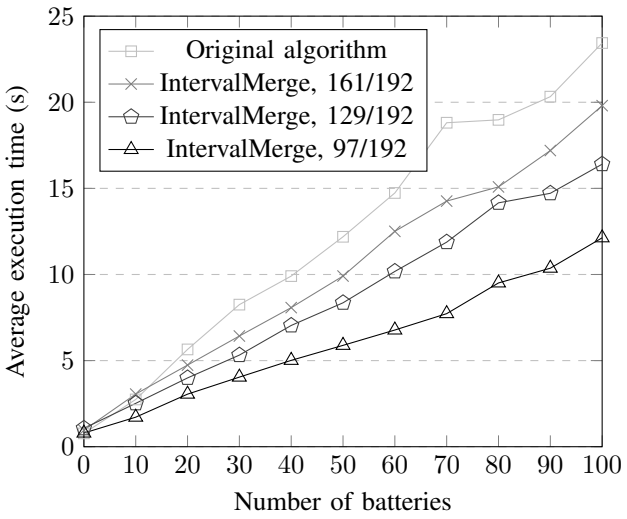

Fig. 2. DEMKit execution time original algorithm vs IntervalMerge

From this figure we can conclude that when the simulation is up-scaled (more houses or a higher level hierarchy), the time saved by using IntervalMerge can be estimated to be linearly in the number of used intervals. This is also in agreement with Section III.

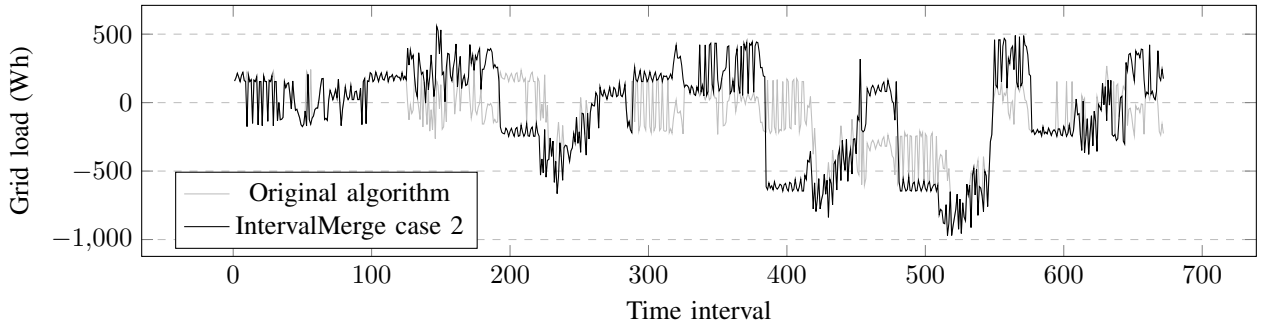

Fig. 3. DEMKit grid load original algorithm vs IntervalMerge case 2

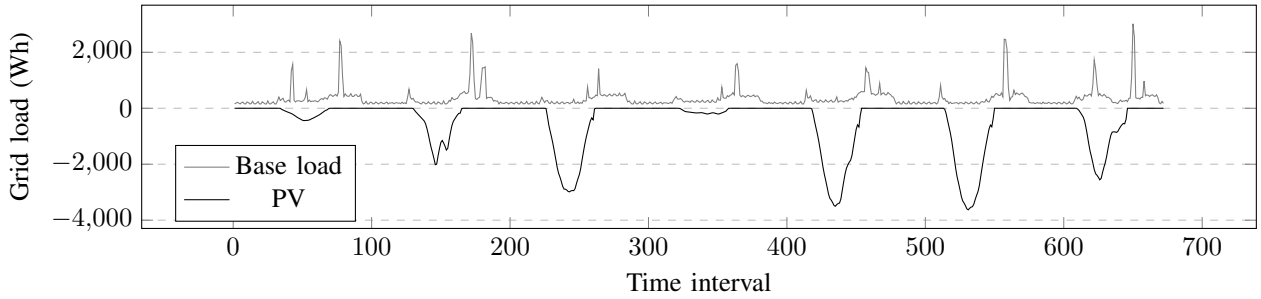

Fig. 4. DEMKit Base load and PV

B. Performance loss versus time gain

To evaluate the performance of the IntervalMerge algorithm, different numbers of time-intervals have been used and tested in the simulation software DEMKit. The data retrieved from these tests contains the execution time in seconds, the state of charge of the batteries over time and the resulting load of the house on the grid. We use this load to calculate the Euclidean distance between this load profile and the desired profile. These results give insight into the performance of using non-regular time intervals in the profile steering algorithm.

To collect the results, we implemented a logging algorithm. The performance of the algorithm is calculated by accumulating the square of the Euclidean distance of the simulated load on the network. The results of the nine test cases are presented in Table I. In order to visualize the gain and loss, Fig. 6 shows the Pareto front where each case number references the case given in Table I. Hereby, test case 1 is the original profile steering algorithm [3]. The *eu-n* in Table I stands for the Euclidean distance which is given as a dimensionless number and the percentage it deviates from the original profile steering algorithm. Hereby the Euclidean norm is the distance between the desired profile (zero profile) and the generated power profile. The performance loss (deviation from the Euclidean norm) between the original algorithm and IntervalMerge is shown in Fig. 3 and Fig. 4.

Although on a first view, it seems that there is a large deviation between the original algorithm and the IntervalMerge in the distance shown in Fig. 3, e.g. at time interval 400. A closer look shows that the extra load on the grid is a result of the battery discharging on the grid. E.g. at the beginning

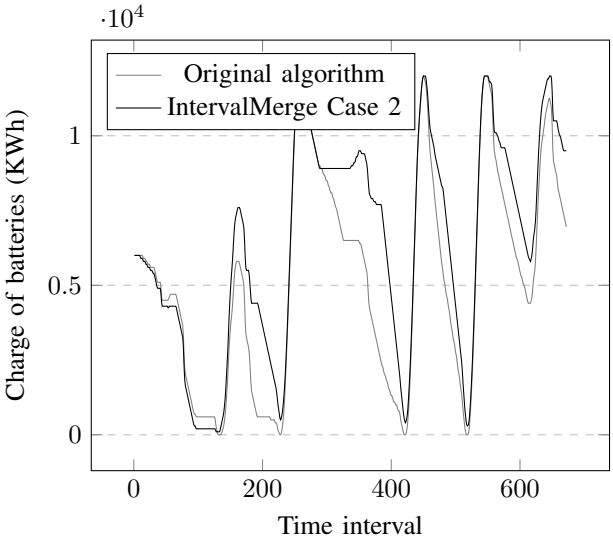

Fig. 5. DEMKit SOC original algorithm vs IntervalMerge case 2

of a sunny day, it would be optimal to have a low SoC so that the battery can charge itself with excess energy from the solar panels. If the SoC is relatively high, the profile steering algorithm lowers it by discharging the battery on the grid as can be observed in Fig. 5. The execution time of the different cases is given in Table I in seconds(ex_t(s)) and as a percentage it deviates from the original profile steering algorithm(ex_t(%)). Important to note is that the execution time includes the time for initializing the simulation.

TABLE I
RESULTS

case	α	eu-d(-)	Ex-t(s)	eu-n(%)	Ex-t(%)
1	$< \alpha_1, ..\alpha_{192} = 1 >$	2047.9	6.0	0%	0%
2	$< \alpha_1, ..\alpha_{96} = 1, \alpha_{97}, ..\alpha_{144} = 2 >$	2216.2	4.8	8.2%	-20.2%
3	$< \alpha_1, ..\alpha_{96} = 1, \alpha_{97}, ..\alpha_{120} = 4 >$	2874.5	4.7	40.4%	-22.2%
4	$< \alpha_1, ..\alpha_{96} = 1, \alpha_{97}, ..\alpha_{108} = 8 >$	3326.6	4.5	62.4%	-25.1%
5	$< \alpha_1, ..\alpha_{96} = 1, \alpha_{97}, ..\alpha_{104} = 12 >$	3653.9	4.1	78.4%	-31.5%
6	$< \alpha_1, ..\alpha_{96} = 1, \alpha_{97}, ..\alpha_{100} = 24 >$	3886.8	4.6	89.8%	-24.1%
7	$< \alpha_1, ..\alpha_{96} = 1, \alpha_{97} = 96 >$	4077.8	4.0	99.1%	-32.9%
8	$< \alpha_1, ..\alpha_2 = 12, \alpha_3, ..\alpha_{74} = 1, \alpha_{75}, ..\alpha_{82} = 12 >$	4284.4	4.0	109.2%	-32.6%
9	$< \alpha_1 = 24, \alpha_2, ..\alpha_{73} = 1, \alpha_{74}, ..\alpha_{77} = 24 >$	5210.1	4.3	154.4%	-28.7%

V. CONCLUSION

The presented results show that it is possible to significantly lower the total execution time of the profile steering algorithm by reducing the number of input intervals. Our aim was to investigate if it is possible to reduce the execution time of the profile steering algorithm without affecting performance using non-regular time intervals. The result of case 2 in Table I, shows that a 20% processing time gain is achieved but at the cost of an 8% performance loss. This means that any improvement in execution time seems to result in a loss of performance.

However as explained in Section IV, merging intervals that contain peaks will average them out, resulting in an underestimation of the load for both the baseload and PV panels. The underestimation leads to a difference in the state of charge of the batteries. This difference affects the rolling horizon output by drawing or feeding energy from and to the network to compensate, increasing the overall load on the network.

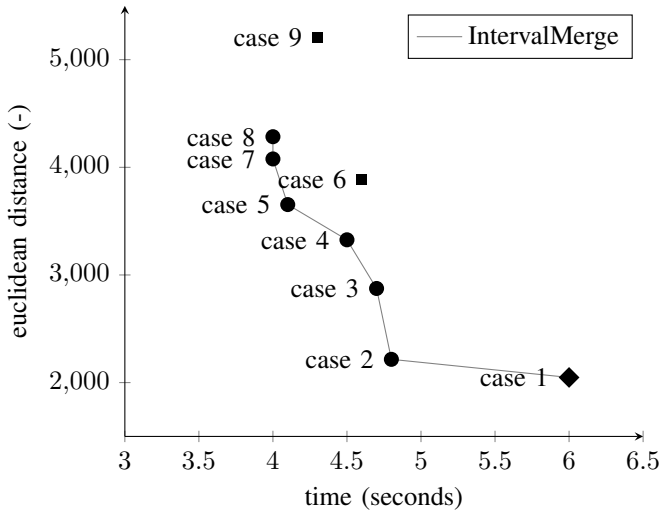

Fig. 6. Pareto front (square = non pareto optimal, circle = perato optimal, diamond = base case)

VI. FUTURE WORK

The evaluation in Section IV is based on the results of the IntervalMerge algorithm on the household level in the control hierarchy. Future work should consider the applicability of merging intervals at higher points in the control hierarchy, e.g. at the neighborhood level.

In the current implementation the time intervals are chosen statically. This may cause problems because a chosen set of non-regular time interval may be sufficient in certain scenarios but be insufficient in others. Therefore, dynamically setting the non-regular time intervals could be added to improve the overall performance of the algorithm.

Table I shows that in test numbers six and nine the euclidean distance increases and so does the execution time. The increase in execution time is unexpected but may be caused by an increase in the number of planning iterations. This relation should be investigated in future work.

Extended testing of the IntervalMerge algorithm is beneficial for strengthening the practical application and therefore should be performed in future work. This should be done in combination with more theoretical research into the algorithm in combination with more references.

Currently IntervalMerge is only implemented for batteries, in future work the IntervalMerge algorithm should be implemented for other devices. As many devices, e.g. electric vehicles (EV) make use of the same underlying approach, this is expected to be relatively simple.

REFERENCES

[1] P. Siano, "Demand response and smart grids—a survey," *Renewable and sustainable energy reviews*, vol. 30, pp. 461–478, 2014.

[2] M. H. Christensen, D. C. Nozal, I. Kavadakis, and P. Pinson, "Data-driven learning from dynamic pricing data-classification and forecasting," in *2019 IEEE Milan PowerTech*. IEEE, 2019, 6 pages.

[3] 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 *2015 IEEE Eindhoven PowerTech*, 2015, 6 pages.

[4] J. F. Marquant, R. Evins, and J. Carmeliet, "Reducing computation time with a rolling horizon approach applied to a milp formulation of multiple urban energy hub system," *Procedia Computer Science*, vol. 51, pp. 2137–2146, 2015.

[5] G. Hoogsteen, J. L. Hurink, and G. J. M. Smit, "Demkit: a decentralized energy management simulation and demonstration toolkit," in *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, 2019, 5 pages.

[6] T. van der Klauw, "Decentralized energy management with profile steering - resource allocation problems in energy management," Ph.D. dissertation, University of Twente, Enschede, The Netherlands, May 2017, CTIT Ph.D. thesis Series No. 17-424.