On the Scalability of Decentralized Energy Management using Profile Steering

Gerwin Hoogsteen, Marco E. T. Gerards, Johann L. Hurink
University of Twente, Department of EEMCS
Enschede, the Netherlands
{g.hoogsteen, m.e.t.gerards, j.l.hurink}@utwente.nl

***Abstract*—Optimizing the use of flexibility, provided by e.g. batteries and electric vehicles, provides opportunities for various stakeholders. Examples are aggregators acting on energy markets, or energy cooperations willing to maximize their self-consumption. However, with large numbers of devices that need to be scheduled, the underlying optimization problem becomes difficult. This paper investigates the scalability of a smart grid optimization approach called Profile Steering. This approach uses a hierarchical structure to perform distributed optimization. In this paper, the approach is extended with methods to accept multiple profiles at once and the possibility to prune children with little flexibility. Simulation studies with almost 20,000 households are carried out to evaluate the scalability of Profile Steering. The results show that, with the presented improvements, the required optimization time of Profile Steering scales linearly with the number of children and a speedup factor of 56 is achieved with 1000 households. Furthermore, the approach scales well across multiple computing processes.**

***Index Terms*—decentralized control, smart grids, model predictive control, optimization**

I. Introduction

Due to the depletion of fossil fuels and their impact on global warming, more renewable energy sources (RES) are present in distribution grids. Most RES produce electricity, the share of electricity in the total energy demand is also expected to rise [9], for which the existing infrastructure may not provide enough distribution capacity. Another challenge is that most renewable electricity is generated using intermittent sources, resulting in increasing difficulties to balance supply and demand. Decentralized energy management (DEM) can exploit the locality of energy generation to relieve the stress on the electricity network [7]. Furthermore, flexible loads (e.g. batteries and electric vehicles) can be integrated in such a system to match the demand with the supply.

To fully benefit from the potential of devices that can offer flexibility to the distribution grid, a scalable (many devices need to be controlled) and versatile (the cluster of devices is heterogeneous) optimization and control method is required. Centralized optimization methods in general do not scale well with an increasing number of controllable devices. Iacovella et al. [8] use a few tracer devices to reduce the complexity, which accurately describe the flexibility of a larger group of devices. Alternatively, Claessens et al. [1] use reinforced learning to create an abstract model of the flexibility, which is used to optimize the power profile of the connected devices. On the other hand, transactive control and coordination [10] embraces (distributed) local intelligent controllers in a hierarchical tree. With a transactive energy methodology, such as double-sided auctions, the processing and computation time scales with the height of the tree instead of the number of devices. The same degree of scalability is also applicable to distributed optimization algorithms, such as Profile Steering [3], and alternating direction of multipliers method (ADMM) [4], [14].

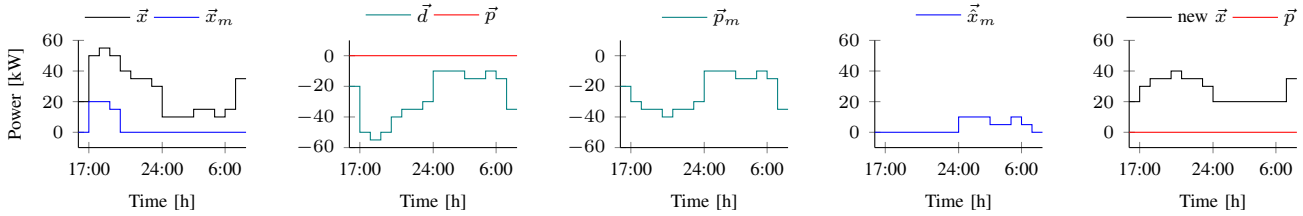
978-1-5386-4505-5/18/$31.00 © 2018 IEEE

Scalability of DEM optimizations algorithms is becoming increasingly important for e.g. aggregators to act on energy markets. Hansen et al. [5] investigated this and their method requires 113 minutes to solve a problem with 5555 customers using a genetic algorithm based centralized optimization method. Rivera et al. [11] have evaluated the performance of ADMM under various circumstances. Significant reductions in required computation time and memory usage are achieved with a parallelized implementation compared to a sequential implementation. Lastly, Toersche [12] shows signs that the computation time with Profile Steering can be reduced significantly by accepting multiple profiles per iteration.

In this paper, we further investigate the scalability of Profile Steering using the DEMKit optimization and simulation framework developed at the University of Twente [7]. Within this tool, complete (micro)grids can be modelled in detail with device, infrastructure and control objects. This paper presents and evaluates solutions to significantly reduce the required computation time of the Profile Steering heuristic. Furthermore, inherent parallelism is exploited and the scalability of distributed optimization is evaluated by extending DEMKit with multiprocessing support. The main contributions of this paper to Profile Steering are:

- solutions to increase the computation efficiency,
- distributed energy optimization by exploiting parallelism,
- a scalability analysis with up to 19,200 households.

The remainder of this paper is organized as follows. A brief introduction to Profile Steering and its current shortcomings is given in Section II. Solutions to increase the computational efficiency are presented and evaluated in Section III, whereas scalability using parallelism is analyzed in Section IV. Conclusions and directions for further speed and search-space improvements are addressed in Section V.



(a) Initial profiles (b) Incentive (c) EV desired (d) EV candidate (e) Parent controller

Fig. 1. Visual example of Profile Steering with an EV m arriving at 17:00 and leaving at 7:00 the next day and energy demand of 55 kWh. (a) The initial power profile of the parent ($\vec{x}$) and greedy planning of the EV ($\vec{x}_m$). (b) The parent controller sends out an incentive ($\vec{d}$), based on the desired profile ($\vec{p}$) to the EV controller. (c) The EV controller produces a local desired profile ($\vec{p}_m$) by adding $\vec{x}_m$. (d) The EV optimizes its profile based on the desired profile, resulting in a (selected) candidate power profile ($\vec{\hat{x}}_m$). (e) Finally the profile at the parent is updated (new $\vec{x}$), where the profile is flattened compared to (a).

II. Profile Steering

This section gives a brief overview of Profile Steering as presented in [3], followed by an analysis to identify the major bottlenecks of this algorithm.

A. Algorithm

The Profile Steering algorithm (an example is given in Fig. 1) is a heuristic to schedule the energy profile of a cluster of devices for an upcoming period of N time intervals using a hierarchical structure of controllers. It does so by sending an explicit *desired profile* ($\vec{p} = [p_1, \ldots, p_N]^T$) to lower level controllers, which we refer to as *children*. These profiles express exactly the energy profile that a cluster of children should attain according to their higher level controller, which we refer to as *parent*. The objective for the children is to follow this steering signal by minimizing the distance between this desired profile and the aggregated *power profile* $\vec{x} = [x_1, \ldots, x_N]^T$, according to some vector norm. In our case, we use the Euclidean distance (i.e., minimize $||\vec{x} - \vec{p}||_2$). A desired profile could be a zero-profile (i.e. $[0, \ldots, 0]^T$) which expresses that balance of production and consumption of energy is desired.

Initially, each child $m \in M$ receives this desired profile $\vec{p}$ and minimizes the distance between this desired profile and its own power profile $\vec{x}_m = [x_{m,1}, \ldots, x_{m,N}]^T$. The parent controller receives all power profiles from connected children, resulting in an initial aggregated power profile $\vec{x} = \sum_{m \in M} \vec{x}_m$. Then, iteratively, the parent controller sends out a *difference profile* $\vec{d} = \vec{p} - \vec{x}$. Each child obtains a local desired profile $\vec{p}_m = \vec{d} + \vec{x}_m$ to find a new *candidate power profile* $\vec{\hat{x}}_m$ that minimizes $||\vec{x}_m - \vec{p}_m||_2$. Each child calculates its improvement e_m that is achieved by replacing the current power profile $\vec{x}_m$ by $\vec{\hat{x}}_m$: $e_m = ||\vec{x}_m - \vec{p}_m||_2 - ||\vec{\hat{x}}_m - \vec{p}_m||_2$, and communicates the obtained improvement e_m back to the parent controller. The parent controller selects the child with the largest improvement, and may couple a (monetary) reward to this improvement to attract participants. The chosen device m commits its own candidate profile (i.e., $\vec{x}_m$ becomes $\vec{\hat{x}}_m$) and communicates its changes, such that the parent can update $\vec{x}$. With this method, most information stays local to minimize communication requirements. Subsequently, a new difference profile $\vec{d}$ is obtained and the process is repeated until a maximum number of iterations is reached or no significant improvement is made, i.e. $e_m < e_{\min}$ where $e_{\min}$ is a positive value denoting the minimum required improvement. The result of the algorithm is a schedule $\vec{x}_m = [x_{m,1}, \ldots, x_{m,N}]^T$ for each child m.

Note that the parent controller is agnostic about the details of its children, as long as they communicate as described above. Thus, a child could optimize the operation of a device (for optimization algorithms under the aforementioned steering signals see e.g. [13]). But, a child can also be another node running Profile Steering, making it possible to form a hierarchical (tree) structure, where each child performs several iterations before returning the candidate profile. This does require additional bookkeeping (see Chapter 4 of [7]).

B. Computational Efficiency

The time required to finish one iteration of Profile Steering depends on the time it takes for the children to obtain a candidate profile and whether the children optimize their profile sequentially or in parallel. In the sequential case the total computational time is $D = D_{\text{ps}} + \sum_{m \in M} D_m$, where D_{ps} is the time required by the Profile Steering algorithm to perform calculations and bookkeeping, and D_m is the computation time of child $m \in M$. In the parallel case, the computation time depends on the worst case execution time among the children. Hence $D = D_{\text{ps}} + \max(D_0, D_1, \ldots, D_m)$.

Assuming that all used processors are equal, D_m and D_{ps} can be translated directly into an amount of processing power. In this case, for an iteration i, with $D_{\bar{m}}$ being the selected child, we can calculate which fraction of the computations is used to improve the profile. The candidate profiles obtained by all not selected children are discarded and hence their effort remains unused, resulting in a computation efficiency $\eta_i < 1$. If we now consider a situation where the children recursively obtain a profile D_m by steering a group of *grandchilden*, the computation efficiency to obtain such a profile is $\eta_{\bar{m}} < 1$. Thus, the efficiency in terms of computation time is found using:

$$\eta_i = \frac{D_{\text{ps}} + D_{\bar{m}}\eta_{\bar{m}}}{D_{\text{ps}} + \sum_{m \in M} D_m}. \tag{1}$$

Assume that the number of iterations grows linearly in M, we get a quadratic increase in computation time in a sequential implementation. In the parallel case, the computation time and

number of processing units both increase linear with the number of children. The resulting efficiency decreases considerably if more layers are added to the control tree.

It is hard to give an exact figure for the total computation time and efficiency as this heavily depends on the situation, target profile and foremost the amount and type of considered flexibility. From experience, with low $e_{\min} \leq 1$, we found that the number of children with sufficient flexibility approximates the number of required iterations. The results in Section III show that this is also the case in this paper.

III. Improvements to Profile Steering

The main bottleneck identified in Section II is the resulting low efficiency due to accepting only one profile per iteration. Furthermore, not all children are (always) able to provide flexibility to the system, but still require computational time and communication bandwidth. This section presents and evaluates improvements (pseudocode of this approach is summarized in Algorithm 1) to these problems.

A. Accepting Multiple Profiles

To reduce computation time, the number of accepted profiles can be increased. Toersche [12] experimented with accepting profiles by first sorting the children based on e_m. Then, candidate profiles $\hat{\vec{x}}_m$ are accepted as long as they improve the overall objective value. This method resulted in a decrease from 380 to 65 iterations in a case with 400 households.

Instead of checking the improvement at the parent level to avoid overshoots, we propose to distribute only a fraction of the master problem to each child. This fraction depends on the number of simultaneous commits (accepted candidate profiles) in an iteration, denoted as μ. Thus in an iteration, each child receives $\vec{d} = [\frac{d_1}{\mu}, \frac{d_2}{\mu}, \ldots, \frac{d_N}{\mu}]^T$ as a steering signal. This strategy allows a number of children to contribute to the solution simultaneously with a reduced risk of (large) overshoots. Now, equation (1) can be rewritten into:

$$\eta_i = \frac{D_{\text{ps}} + \sum_{\bar{m} \in \bar{M}} D_{\bar{m}} \eta_{\bar{m}}}{D_{\text{ps}} + \sum_{m \in M} D_m}, \tag{2}$$

where $\bar{M}$ is the set of selected children in an iteration. With $\mu = |M|$ all computational power is used (assuming $\eta_{\bar{m}} = 1$).

In each iteration, all children communicate their improvement e_m to the parent controller. The parent sorts the children in descending order based on their respective e_m value and subsequently selects the top μ children to commit their profile. As a large μ also includes a risk of overshoots, we propose to reduce μ after each iteration, i.e. by dividing μ by a factor $\beta > 1$. The number of simultaneous commits in iteration $i \geq 1$ is now given by $\mu_i = \max(1, \lfloor \mu_{i-1}/\beta \rfloor)$.

With a fraction of the difference profile, the contribution e_m that a child can make is reduced. Hence, there is a risk that the Profile Steering algorithm might stop iterating too soon if $e_{\min}$ is not adapted to reflect this change. We propose to make $e_{\min}$ depend on μ_i and the number of controlled children M, i.e.:

Algorithm 1 Extended Profile Steering algorithm.

```
Request each child m ∈ M to minimize ||x⃗_m − p⃗||_2
x⃗ := Σ_{m=1}^{M} x⃗_m                          ▷ Total cluster consumption
k := 1
repeat
    d⃗ := (p⃗ − x⃗)/μ                            ▷ Difference profile
    for m ∈ M do                                ▷ For each child M
        p⃗_m := d⃗ + x⃗_m
        Find a candidate x̂⃗_m minimizing ||x⃗_m − p⃗_m||_2
        Communicate e_m := ||x⃗_m − p⃗_m||_2 − ||x̂⃗_m − p⃗_m||_2
    end for

    Sort M descending based on e_m
    j := 1
    while j ≤ μ do
        x⃗ := x⃗ + x̂⃗_j − x⃗_j                    ▷ Update the total consumption
        x⃗_j := x̂⃗_j                             ▷ Update the profile of child j
        j := j + 1
    end while
    e_{min,i} := (M e_min)/√μ

    if Use child pruning then
        Sort M ascending based on e_m
        j := 1
        while j ≤ μ ∧ e_j < e_min/√μ do
            Remove j from M
            j := j + 1
        end while
    end if
    μ := min(|M|, max(1, ⌊μ/β⌋))
    k := k + 1
until ∀e_m < e_{min,i} ∨ k > k_max ∨ |M| = 0
```

$$e_{\min,i} = \frac{M e_{\min}}{\sqrt{\mu_i}}. \tag{3}$$

In terms of fairness, choosing $\mu_0 = M$ ensures that candidate profiles of all children are accepted at least once. From a technical point of view this approach spreads the problem better over the children, especially in the first iterations. This generally results in reduced local peaks and an improved power quality [7]. From a computational complexity point of view, the addition of sorting the children does increase the complexity. However the number of children is generally low, limiting the size of the sorting problem.

B. Pruning Inflexible Children

Another observation is that some devices lack (significant) flexibility, as surveyed in [2]. For these devices the probability that they provide sufficient contribution to the problem (i.e. a significant e_m) to be selected is small. The approach is extended with the option to automatically prune the pool of considered children based on the last contribution. A child is removed from the pool of children for the current optimization process if the improvement $e_m < \frac{e_{\min}}{\sqrt{\mu_i}}$.

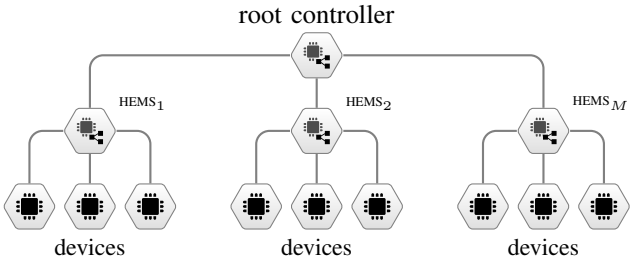
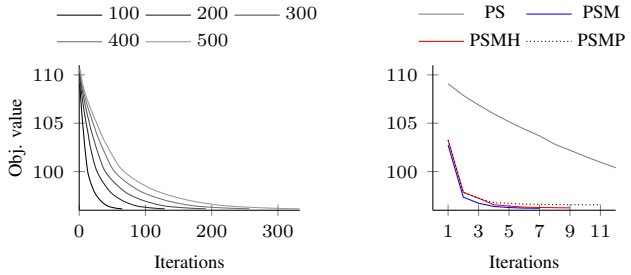


Fig. 2. Overview of the structure with M Profile Steering instances (HEMS), communicating with one root controller and household devices.



(a) Multiple problem sizes for PS (b) improved PS*

Fig. 3. Convergence of the original Profile Steering algorithm for different number of houses (a), and for 100 households with the presented improvements (b). Note that the objective value is based on 192 intervals.

However, note that the above decision is made based on the last improvement, which depends on the received difference profile $\vec{d}$. As the difference profiles change, children with little flexibility may contribute significant improvements in later iterations. With the proposed method of pruning, these children may be pruned too early, which limits the solution that can be found. All children are added to the pool again at the start of a new run of the Profile Steering algorithm.

C. Evaluation Method

To evaluate the proposed improvements to Profile Steering, a synthetic model of a neighbourhood with 100 houses is generated using the profile generator from [6]. In this neighbourhood, 50% of the households are equipped with solar panels and all households have a smart dishwasher. Significant flexibility in this neighbourhood is provided by batteries (25% penetration, 2, 5 or 12 kWh capacity, maximum power of 3700 W), Plug-In Hybrid Electric Vehicles (25% penetration, 12 kWh capacity, maximum power of 3700 W), and Full Electric Vehicles (25% penetration, 60 kWh capacity, maximum power of 7400 W). In total, 61 of the 100 households have significant flexibility to offer to the system. In all simulations, the objective of the root controller is:

$$\text{minimize } ||\vec{p} - \vec{x}||_2,$$

where $\vec{p} = [0, 0, \ldots, 0]^T$. The interval length is set to 15 minutes and a total of two days is optimized (192 intervals). Within the case studies, we compare the results of the first day (96 intervals), where the optimization of a second day is used to prevent the algorithm from exhausting too much flexibility (which would result in worse results the next day). The other parameters are $e_{\text{min}} = 1$ and $\beta = 2$.

The configuration of these households is loaded into the DEMKit optimization and simulation software (written in Python) [7]. The optimizations and simulations are executed on an Intel Core-i7 5820K clocked at 4 GHz (6 cores, 12 threads) with 16 GB RAM. Multiple simulations are run to evaluate the performance and scalability of the proposed improvements. Each household in the model is equipped with a Home Energy Management System (HEMS) that runs the Profile Steering algorithm. The devices of one household are linked to their respective HEMS as children. Furthermore, there is one root controller, also running the Profile Steering algorithm, to optimize the cluster of households (see Fig. 2). Four variants of Profile Steering are compared:

- *PS*: The original Profile Steering concept,
- *PSM*: Profile Steering with multiple commits (III-A),
- *PSMP*: PSM with pruning (III-B) enabled on each level,
- *PSMH*: PSM with pruning (III-B) enabled at the HEMS.

These approaches are all simulated using a single thread and multiple cluster sizes, ranging from 100 to 1000 households in steps of 100 households. To achieve a large number of households, we duplicate the original cluster of 100 households. To compare between simulations with different cluster sizes, we scale down all results to a per household average.

D. Sequential Simulation Results

Numerical results of the simulation studies with the sequential implementations are given in Table I. From these numbers it is clear that the original Profile Steering approach results in a quadratically increasing optimization time as the number of households (M) increases, resulting in an optimization time of almost 52 minutes for 1000 households. With the presented improvements, linear scaling is obtained with an average optimization time of 56 ms per household. For a case of 1000 households, this results in a speedup factor of almost 56. The effect of pruning is negligible in most situations w.r.t. optimization time. However, pruning (PSMP) does effect the obtained average objective value per household ($||\frac{\vec{p}-\vec{x}}{|M|}||_2$), which is similar across the other (PS, PSM and PSMH) cases.

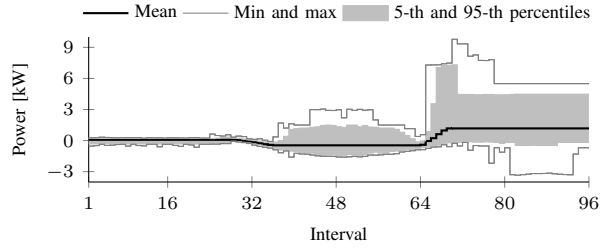
The addition of accepting multiple candidate profiles per iteration (PSM) significantly improves the convergence speed of the algorithm as shown in Fig. 3. The original approach requires approximately one iteration per household with significant flexibility. Accepting multiple profiles results in a reduced and steady number of iterations which is unaffected by the number of households. On the other hand, the number of accepted candidate profiles is increased by a factor of 2.3 on average, increasing the amount of required communication. Lastly, we also observe the effect of more flattened local profiles across the cluster of households. Fig. 4 shows that the PSM approach reduces the peaks of individual households in comparison with the PS case.

TABLE I
NUMERICAL RESULTS OF THE SEQUENTIAL PROFILE STEERING IMPLEMENTATION FOR 96 OF 192 INTERVALS

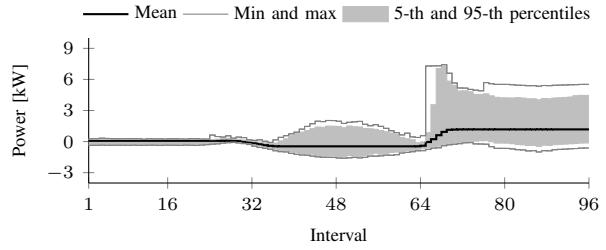
	iterations				accepted profiles				objective value				optimization time [s]			
M	PS	PSM	PSMH	PSMP	PS	PSM	PSMH	PSMP	PS	PSM	PSMH	PSMP	PS	PSM	PSMH	PSMP
100	66	7	9	11	66	156	158	160	46.61	46.59	46.63	46.84	28	6	6	5
200	131	7	8	9	131	314	315	316	46.61	46.59	46.64	46.86	111	12	11	10
300	193	7	8	9	193	470	472	473	46.61	46.59	46.64	46.86	238	18	18	15
400	262	7	8	9	262	629	632	633	46.61	46.61	46.64	46.86	451	23	23	22
500	330	7	7	8	330	785	785	788	46.61	46.61	46.65	46.85	753	29	27	26
600	417	7	7	8	417	943	943	947	46.63	46.61	46.65	46.85	1150	36	33	32
700	491	6	7	8	491	1089	1099	1104	46.61	46.62	46.65	46.85	1553	38	39	37
800	558	6	7	8	558	1247	1259	1265	46.63	46.62	46.65	46.86	2077	43	45	44
900	636	6	7	8	636	1402	1416	1423	46.61	46.62	46.65	46.85	2609	49	52	48
1000	706	6	7	8	706	1558	1573	1580	46.62	46.62	46.65	46.86	3177	57	57	57

IV. EXPLOITING PARALLELISM

The Profile Steering approach is developed with decentralized control in mind. The hierarchical setup allows the distribution of the difference profile over multiple processes to work on the optimization problem in parallel. In this case, the parent communicates the difference profile to all children and waits for all the children to return their obtained improvement e_m. To test the performance of a distributed Profile Steering algorithm, we extended DEMKit with network communication and a communication protocol. The Zero Messaging Queue networking (ZMQ) library is used to make these processes communicate with each other. The developed communication protocol allows multiple DEMKit instances, running in separate processes, to interact with each other in a similar fashion as used in the single threaded implementation. This results in a modular structure where DEMKit configurations can be decomposed into multiple smaller configurations. Hence, servers, workstations or a cluster of low-power embedded systems can be targeted.



(a) PS



(b) PSM

Fig. 4. Optimized profile with PS (a) and PSM (b), indicating that PSM reduces the extreme peaks. Where Mean indicated the scheduled average household consumption, Min and Max indicate the minimum and maximum power consumption of one house respectively, and the 5-th and 95-th percentiles indicate the spread for the majority of households.

A. Multiprocessing Results

To test the scalability of Profile Steering using a multiprocessing setup we use different numbers of clusters. In this case there is one root controller, running in its own process, that controls a number of M clusters. Each of these clusters run in a separate DEMKit process and contains a group of 100 households, including a cluster controller running Profile Steering that is connected with the root controller (see Fig. 5). For this simulation we use the same system, settings and households as presented in Section III-C, except for $e_{\min} = 100$. Each cluster has a random selection of households from a list containing copies of the households previously introduced. Two simulations are run, one with the PSM approach from Section III-C, and a variation PSM* which has a fixed number of $k_{\max} = \lceil \sqrt{M} \rceil$ iterations. The latter approach is used to compare the objective value with more iterations.

Table II presents numerical results of the multiprocessing simulation. The PSM approach requires 2 iterations before the stopping conditions are met, except for $M = 4$ and $M = 6$ which required 3 iterations. The low number of iterations required at the root controller is explained by the fact that the flexibility in each cluster of 100 households can be expected to be heterogeneous, which already results in a near optimal solution after the first iteration. Hence, high in the tree, only few iterations are required to share some flexibility between clusters. Using more iterations (PSM*) does not significantly improve the results, but adds computation time.

For PSM, the required optimization time is generally around 12 ms per house when all 12 threads are filled. For the considered cluster sizes, the optimization time scales linearly with the number of households (or clusters) when all processing cores are used. A speedup factor of 3.4 is obtained when comparing the distributed to the sequential implementation with 800 households. We note that the initialization phase requires significant time in both cases. Each cluster required approximately 75 MB of RAM (no swap memory was used).

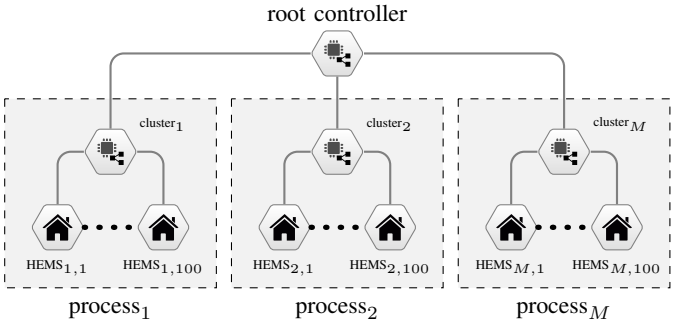


Fig. 5. Overview of the structure with M clusters of 100 households, running in separate processes, communicating with one root controller.

V. Conclusions and Future Work

The scalability of the Profile Steering algorithm is analyzed in this paper. A significant increase in efficiency is achieved using the presented technique of splitting the problem in a way such that multiple profiles can be accepted per iteration, and the possibility to prune children with little flexibility. The latter is not useful on a higher level, however. Simulation results show that, for a scheduling problem with similar flexibility, this results in a linearly scaling computational time with the number of children while the obtained results remains similar. Furthermore, it is demonstrated that inherent parallelism scales well over multiple processes. With the presented improvements, Profile Steering can be used to analyze both small (e.g. microgrids) and large systems (e.g. large aggregator portfolios).

Despite these significant improvements to Profile Steering, there remain directions for future work. One observation is that not all children have significant flexibility, thus it may be beneficial to communicate a larger portion of the problem to all children, such that the children propose a nearly optimal solution in earlier iterations. However, this requires more knowledge of the available flexibility per time interval. Another option for improvements lays in the field of clustering techniques. By organizing the clusters differently, e.g. based on flexibility, easier or more balanced problems may be formulated that can be optimized more efficiently with Profile Steering. Especially with large flexibility portfolios of aggregators this could be beneficial.

On the other hand, with larger hierarchical trees, different objectives for lower level clusters may arise due to e.g. local constraints or local generation. Local objectives should be introduced in Profile Steering to also benefit from locally generated energy, where a trade-off between local and global objectives can be made. This way, device specific costs, such as battery wearing, may also be incorporated. Such future research should also include incentive schemes that specify how participants can benefit from offering flexibility.

Acknowledgment

The authors thank Dutch national program TKI iDeego (project ORTEP) and RVO for their support.

TABLE II
Numerical results of the distributed implementation for 96 of the 192 scheduled intervals

		objective value		optimization time [s]		time per house [ms]	
M	Houses	PSM	PSM*	PSM	PSM*	PSM	PSM*
1	100	46.59	46.59	6.3	5.5	63.5	55.3
2	200	46.64	46.63	7.1	6.4	35.7	31.8
4	400	46.62	46.63	8.3	7.7	20.7	19.4
6	600	46.64	46.65	10.9	8.1	18.1	13.4
8	800	46.63	46.63	12.7	12.5	15.9	15.7
12	1200	46.64	46.63	15.8	17.6	13.1	14.6
16	1600	46.63	46.62	19.4	22.8	12.1	14.3
24	2400	46.62	46.62	28.8	33.6	12.0	14.0
32	3200	46.63	46.62	37.7	48.2	11.8	15.1
48	4800	46.62	46.61	54.4	74.8	11.3	15.6
64	6400	46.63	46.61	74.2	111.7	11.6	17.5
96	9600	46.63	46.62	112.4	180.0	11.7	18.8
128	12800	46.63	46.61	151.4	265.5	11.8	20.7
192	19200	46.63	46.62	223.3	438.8	11.6	22.9

References

[1] B. J. Claessens, S. Vandael, F. Ruelens, and M. Hommelberg, "Self-learning demand side management for a heterogeneous cluster of devices with binary control actions," in *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, October 2012, pp. 1–8.

[2] M. E. T. Gerards and J. L. Hurink, "On the value of device flexibility in smart grid applications," in *2017 IEEE Manchester PowerTech*, June 2017, pp. 1–6.

[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 *PowerTech, 2015 IEEE Eindhoven*, June 2015, pp. 1–6.

[4] R. Halvgaard, L. Vandenberghe, N. K. Poulsen, H. Madsen, and J. B. Jørgensen, "Distributed Model Predictive Control for Smart Energy Systems," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1675–1682, May 2016.

[5] T. M. Hansen, R. Roche, S. Suryanarayanan, A. A. Maciejewski, and H. J. Siegel, "Heuristic Optimization for an Aggregator-Based Resource Allocation in the Smart Grid," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1785–1794, July 2015.

[6] G. Hoogsteen, A. Molderink, J. L. Hurink, and G. J. M. Smit, "Generation of flexible domestic load profiles to evaluate demand side management approaches," in *2016 IEEE International Energy Conference (ENERGYCON)*, April 2016, pp. 1–6.

[7] G. Hoogsteen, "A Cyber-Physical Systems Perspective on Decentralized Energy Management," Ph.D. dissertation, University of Twente, December 2017.

[8] S. Iacovella, F. Ruelens, P. Vingerhoets, B. J. Claessens, and G. Deconinck, "Cluster Control of Heterogeneous Thermostatically Controlled Loads Using Tracer Devices," *IEEE Transactions on Smart Grid*, vol. 8, no. 2, pp. 528–536, March 2017.

[9] International Energy Agency (IEA), "World Energy Outlook, Executive Summary," Tech. Rep., 2016.

[10] K. Kok and S. Widergren, "A Society of Devices: Integrating Intelligent Distributed Resources with Transactive Energy," *IEEE Power and Energy Magazine*, vol. 14, no. 3, pp. 34–45, May 2016.

[11] J. Rivera, C. Goebel, and H. A. Jacobsen, "Distributed Convex Optimization for Electric Vehicle Aggregators," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1852–1863, July 2017.

[12] H. A. Toersche, "Effective and efficient coordination of flexibility in smart grids," Ph.D. dissertation, University of Twente, October 2016, ISBN 978-90-365-4197-8.

[13] T. van der Klauw, "Decentralized Energy Management with Profile Steering - Resource Allocation Problems in Energy Management," Ph.D. dissertation, University of Twente, May 2017.

[14] M. G. Vayá, G. Andersson, and S. Boyd, "Decentralized control of plug-in electric vehicles under driving uncertainty," in *IEEE PES Innovative Smart Grid Technologies, Europe*, October 2014, pp. 1–6.