ENERGYCON 2014 • May 13-16, 2014 • Dubrovnik, Croatia

Cascaded Column Generation for Scalable Predictive Demand Side Management

Hermen A. Toersche #1, A. Molderink #, J. L. Hurink *, G. J. M. Smit #

# *Department of Computer Science, University of Twente*
* *Department of Applied Mathematics, University of Twente*
P. O. Box 217, 7500 AE Enschede, the Netherlands
1 h.a.toersche@utwente.nl

***Abstract*—We propose a nested Dantzig–Wolfe decomposition, combined with dynamic programming, for the distributed scheduling of a large heterogeneous fleet of residential appliances with nonlinear behavior. A cascaded column generation approach gives a scalable optimization strategy, provided that the problem has a suitable structure. The presented approach extends the TRIANA smart grid framework for predictive demand side management; the main goal of this framework is peak shaving. Simulations validate that the approach is effective, but also show that the performance degrades for smaller group sizes.**

***Index Terms*—Energy management, mathematical programming, power system management, smart grids.**

I. Introduction

The increasing use of electricity as an energy carrier, combined with the advent of large scale renewable generation, leads to a demand for new sources of flexibility in the electric power industry. Since batteries will probably remain expensive in the foreseeable future, researchers consider the use of demand side flexibility as a possibly cheaper and more efficient alternative. The electricity demand of some residential appliances such as washing machines, electric vehicles and heat pumps is to some extent shiftable in time. An aggregator may shape their combined demand, for example to match demand with supply, relieve grid congestion or operate on markets. This approach is called demand side management (DSM) and is a very popular topic in today's smart grid research [1].

Demand side management approaches have to be scalable, general, flexible and effective. Scalable means that solutions have to be found in reasonable time using little computing resources. General means that it can integrate appliances with different characteristics and needs. Flexible means that it can express various objectives, such as peak shaving and economic optimization. Effective means that it gives good, near-optimal solutions to the problem at hand. In practice, also other requirements are important, such as security, reliability and cost. While literature discusses approaches focusing on various subsets of these requirements, we are not aware of any approach that meets all of these requirements at the same time.

In earlier work, we presented a two-level optimization approach and applied it to a large DSM problem [2]. Using Dantzig–Wolfe decomposition [3], the DSM problem is partitioned into a master problem and a set of subproblems. In contrast to other related work, we do not restrict subproblems to linear models, as linear models are too restrictive to accurately model several real devices. For example, a washing machine can either be switched on or off; there are no in-between options.

As shown in [2], this optimization approach is very effective in terms of objective performance. At first sight, it also appears to be a scalable solution strategy: the subproblems can be distributed and solved in parallel. However, the scalability of the master problem is a serious concern. The master problem can become very hard to solve with an increasing number of subproblems; currently, it clearly dominates the solution process, even when the subproblems are not distributed. Based on these considerations, it is clear that the current method is not practical for a large number of households or the combined control of multiple neighborhoods.

The central idea of this work is that we can divide the master problem itself in smaller parts, each of which can be solved and combined far more efficiently than the original problem. We can do this because the problem has very simple coupling constraints; at each level, we only consider the aggregate electricity demand of all subproblems (over time). All other constraints and objectives (for example peak minimization) are subsequently modeled with these aggregate demand levels. Summarizing, there are only few and local coupling constraints between the subproblems.

In this paper, we exploit this loose coupling *within* the master problem described in [2]: we identify a Dantzig-Wolfe structure, which can be solved by a column generation approach. In turn, the resulting subproblems themselves have a similar structure as the master problem. Provided that there are suitable loose coupling constraints left, this process can be repeated as many times as needed, resulting in a nested column generation approach. For efficiency, we extend this to a *cascaded* approach.

The paper is structured as follows. In Section II, we provide background on current demand side management approaches and decomposition techniques; we address the TRIANA smart grid framework in more detail. In Section III, we introduce cascaded column generation in the context of TRIANA. We follow up with simulations in Section IV. Section V evaluates and discusses the results, as well as the approach in general. We conclude this paper in Section VI.

II. Background

We present a short background on demand side management for smart grids. In Section II-A, we discuss current DSM approaches. Section II-B continues with a discussion on decomposition. Finally, Section II-C ends with a more detailed overview of the TRIANA framework.

A. Demand Side Management

A demand side management framework allows an aggregator to treat a group of appliances as a part of the energy infrastructure. These frameworks split up the smart grid control problem both conceptually (to support more than one type of device) and computationally (by partitioning the optimization problem and allowing workload distribution). Decentralization is essential for scalability. Also, coordination is necessary: direct steering according to a shared signal (price, frequency, voltage) may result in excessive demand response, because too many devices respond to the signal.

We identify three DSM paradigms: transactive control, model predictive control (MPC) and voltage/frequency control; we address the first two paradigms, since these relate to this work.

1) Transactive Control: To overcome the problems of direct steering, in transactive control the aggregator introduces an arbiter. The arbiter determines which of the controlled appliances may use the available resources, according to some relative priority ordering. Each of the appliances specifies a set of control options from which the arbiter can choose. To account for the future, the priority ordering is in part determined by an estimate of the future system state. The selection is typically implemented with an on-line double-sided Walrasian auction. Well-known transactive control implementations include GridWise [4], PowerMatcher [5] and Intelligator [6].

2) Model Predictive Control: MPC is a control engineering technique which explicitly estimates the consequences of control decisions. The behavior is scheduled using a system model. For an example of MPC in an energy context, see [7]. This is more in line with conventional grid control models, see for example [8]. For large scale MPC, decomposition techniques are used (Section II-B); these generally assume linear models.

B. Decomposition

Decomposition provides a theoretical framework to partition problems, which makes decentralization possible. Decomposition schemes restate hard optimization problems as a set of easier yet equivalent connected optimization problems. The decomposition problem is studied in depth by both the operation research and control engineering communities [9]. This has resulted in numerous practical approaches with different performance characteristics and assumptions on the problem at hand. In the context of smart grids, several decomposition algorithms have been considered, in particular dual decomposition (e.g. [10]).

1) Dantzig–Wolfe Decomposition: Recently, Dantzig–Wolfe decomposition [3] has received considerable attention in the context of smart grids as an efficient alternative to dual decomposition (e.g. [2], [11], [12], [13]). This decomposition approach imposes several constraints on the structure of the problem. First, the problem should be linear. Second, the coefficient matrix must have a block-angular structure as in Fig. 1. By this, large parts of the matrix can be solved separately, except for a set of complicating, connecting constraints. This structure is quite common in optimization problems.

$$
\begin{bmatrix}
I & \overline{A}_1 & \cdots & \overline{A}_{T-1} & \overline{A}_T \\ \hdashline
 & A_1 & & & \\
 & & \ddots & & \\
 & & & A_{T-1} & \\
 & & & & A_T
\end{bmatrix}
\begin{bmatrix}
\overline{\mathbf{y}} \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{T-1} \\ \mathbf{x}_T
\end{bmatrix}
=
\begin{bmatrix}
\overline{\mathbf{b}} \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{T-1} \\ \mathbf{b}_T
\end{bmatrix}
$$

Fig. 1: Dantzig–Wolfe angular block structure for linear programs [3]. The top row represents the connecting constraints (with coefficient submatrices $\overline{A}_i$). The submatrices A_i represent the corresponding subproblems. All blank parts of the matrix are zero. I is the identity matrix.

After decomposition, the problem is solved with a column generation procedure. The problem is separated in a master problem which represents the connecting constraints, and a set of subproblems (one for each subblock $A_i\mathbf{x}_i = \mathbf{b}_i$). The general form of this procedure is as follows:

- Generate initial feasible set of columns
- While improving patterns exist:
  - Solve master problem $\mathcal{M}$
  - Translate prices λ from shadow prices π of $\mathcal{M}$
  - Maximize subproblems with $\mathbf{c} = \lambda$
    - Add solution to pattern set if reduced cost > 0
- Recover solution

The procedure adds the relevant parts from the subproblems to the master problem. The previous solution of the master problem is used to determine which parts are relevant: the shadow prices of the connecting constraints translate to objective coefficients for the subproblems. The subproblems can be solved in parallel. When no new parts can be found within the subproblems, the master problem solution corresponds to a globally optimal solution of the original problem.

2) Nested Decomposition: Column generation supports large problems, but the master problem may still become intractable when the problem is too large. Furthermore, in a distributed context, the communication with all the subproblems becomes an issue. Therefore, we want to further decompose the problem. The subproblems resulting from the decomposition are still linear programs. If the subproblems have a suitable structure, these may again be decomposed with Dantzig–Wolfe or a different decomposition scheme. Literature covers nested decomposition for linear programs [14]; more considerations should be taken into account for mixed integer programs [15].

Decomposition of the subproblems does not directly address the size of the master problem. However, the decomposition allows the subproblems to be larger; therefore, we can use larger decomposition groupings at the central master problem. Larger groupings reduce the size of the master problem. As before, to be able to use the approach, these groupings need to be loosely coupled.

C. TRIANA Framework

TRIANA is a framework for large scale distributed demand side management of households in smart grids [1]. The framework combines concepts from transactive control and model predictive control, and covers numerous demand side management applications, ranging from the operation of a fleet of microCHPs to refrigerator scheduling. The approach accounts for both the global and the local problems in a system. The predictive control process is divided in three stages: forecasting [16], planning [17] and operational control [18]. For scalability, the problem is partitioned along its hierarchical structure (Fig. 2). A feedback process iteratively refines the solutions of the parts at subsequent levels.

Many energy control approaches use linear models, since these models have interesting theoretical properties. However, while linear models usually fit well conceptually, they often fail to capture important practical considerations. The main problem for DSM is that the linearity does not account for the discrete switching behavior of individual devices.

The TRIANA framework allows nonlinear device models, which avoids these problems. For planning, we only require that these local problems must be able to optimize their demand $\mathbf{x}_i$ according to some price vector λ_i ($\min \mathbf{x}_i^\intercal \lambda_i$ subject to local constraints). Most of these problems can be stated as mixed integer programs (MIPs), but for efficiency reasons these are often solved by specific dynamic programs (DP).

The work in [2] replaces the iterative search procedure of [19] with a procedure based on Dantzig–Wolfe decomposition. In this work, we use linear models to describe the electricity infrastructure, and discrete models for the device problems at the bottom of the problem structure. This scheduling approach has substantially increased the quality of the found schedules. Furthermore, the flexibility of the planning procedure has improved. However, the presented approach is no longer scalable, because it depends on a monolithic master problem: the upper (global) part of the problem has not been partitioned.

A linear column generation approach can support very large systems. However, to address the nonlinearity of the subproblems, our column generation approach is not fully linear. The interpolation of columns is no longer guaranteed to find a valid solution: the linear column weighting problem ($y_{i,j} \in [0,1]$) changes to a binary column selection problem ($y_{i,j} \in \{0,1\}$), which is much harder to solve. The selection MIP is the main bottleneck in the design, which makes us consider nested decomposition for relatively small instances (more than 100s rather than 10 000s of subproblems).

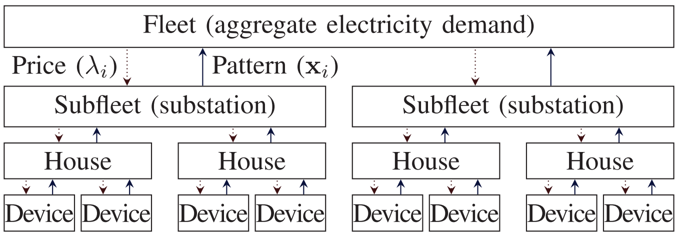


Fig. 2: Partitioned optimization approach in TRIANA.

III. Cascaded Column Generation in TRIANA

A. Nested Decomposition Scheme

We start from the problem in [2] which is decomposed according to the Dantzig–Wolfe scheme as in Fig. 1. Hereby, we treat the device-level problems as if these were linear (sub)problems of the form $A\mathbf{x} = \mathbf{b}$, and we assume that the local cost $z = \mathbf{c}^\intercal \mathbf{x}$ is integrated as a variable in the decision vector $\mathbf{x}$ (with objective coefficient $c_z = 1$). It remains to show how we can bring the original master problem (the top row of Fig. 1) in a form which is suitable for nested decomposition.

The original master problem in TRIANA of [2] gives a linear program which specifies the economic value and operational constraints of an energy system. The program describes the power balance equations and the corresponding power flow over time. The balance equations are organized in a tree, which corresponds to Fig. 2; every subproblem contributes to one of the balance equations in this tree. Every balance equation has a set of constraints over time $\overline{x}_t = \sum_i x_{i,t}, \forall t$ (or, more compactly: $\overline{\mathbf{x}} = \sum_i \mathbf{x}_i$), which we rewrite to $\overline{\mathbf{x}} - \sum_i \mathbf{x}_i = \mathbf{0}$. The imbalance variables $\overline{\mathbf{x}}$ describe the downward power flow (or upward demand) in the tree structure.

In [2], we found that the complete imbalance tree can be mapped to the structure of Fig. 1. In this paper, we consider the nodes in the tree separately. The elements of the tree naturally map to instances of the structure of Fig. 1: for every element, the coefficients of $\overline{\mathbf{x}}$ correspond to I; every balance subtree i corresponds to an $\overline{A}_i$, which describe the contribution $-\mathbf{x}_i$ to the master problem with coefficients $-I$. This process can be repeated to arbitrary depth, until the bottom level problems A_i are reached; these problems are decomposed as before. At this point, we have a tree structure of problems, connected by the balance constraints. Later in this section, we will solve this hierarchical structure by column generation.

The work in [2] lumps all demand into a single central balance equation. Since all subproblems are bound to the same equation, the presented approach can not be used directly to partition the problem. However, we can split off elements of the equation to a new balance group: for example, $\overline{\mathbf{x}} = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4$ can be rewritten as $\overline{\mathbf{x}} = \mathbf{x}_\text{a} + \mathbf{x}_\text{b}$ with $\mathbf{x}_\text{a} = \mathbf{x}_1 + \mathbf{x}_2$ and $\mathbf{x}_\text{b} = \mathbf{x}_3 + \mathbf{x}_4$; the assignment of $\mathbf{x}_\text{a}$ and $\mathbf{x}_\text{b}$ become separate subproblems. Due to associativity and commutativity, we can group the balance equation elements arbitrarily, as long as the resulting problem is equivalent. By repeated splitting, we can make trees of any depth. Similarly, we can (re)group equations, but only if no extra constraints have been imposed on its imbalance variables. We can decide on the structure of the problem tree according to the needs of the decomposition scheme.

In this nested decomposition scheme, the bottom subproblems are still nonlinear. Consequently, we still need to treat all ancestors in the problem tree as binary column selection problems. However, since we only need to consider the direct subproblems one level below each problem, the individual column selection problems can be significantly smaller than before; therefore, these problems should be much easier.

B. Column Generation Algorithm

1) Overview and Notation: We first give an overview of the column generation approach, and the used notational conventions. Next, Algorithm 1 presents the general column generation algorithm for the nested decomposition scheme of Section III-A. For completeness, Algorithm 2 states the behavior of the algorithm for the local device subproblems at the bottom of the problem tree. We follow up with practical improvements to the basic scheme.

We represent calls to the optimization algorithm by $i.\text{solve}(\lambda_i)$, where i is some problem and λ_i is a price vector. The problems are nested: each master problem generates prices for its subproblems, which in their turn generate prices for their subproblems, and so on. The solve operation is polymorphic; the implementation depends on the type of i. The vector $\overline{\mathbf{x}}$ represents abstract demand; by convention, the first entry of $\overline{\mathbf{x}}$ describes the local objective value (with weight 1), and the remaining entries describe electricity demand over time. Each call to solve should give the optimal assignment of $\overline{\mathbf{x}}$ for the given cost function coefficients $\mathbf{c} = \lambda_i$ (minimize $\mathbf{c}^\intercal\overline{\mathbf{x}}$). Let $\mathcal{I}$ represent the set of direct subproblems for the problem at hand, $\mathcal{M}$ the local MIP optimization problem, and let $\mathcal{M}_\text{r}$ be the LP relaxation of $\mathcal{M}$. The solution to a problem $\mathcal{M}$ is denoted by s (and s_r for $\mathcal{M}_\text{r}$). With $s_\text{r}.\pi(\mathbf{x}_i)$ we refer to the shadow prices of the balance rows of $\mathbf{x}_i$ in s_r (i.e. the rows of (3) in Section III-B2). For uniformity, we only consider minimization problems; maximization problems are covered by negating the objective. Finally, let $\mathcal{P}_i$ represent the active pattern set for a subproblem i. These sets describe the patterns which are considered in $\mathcal{M}$ and $\mathcal{M}_\text{r}$; each $\mathcal{P}_i$ is a subset of the patterns generated by i. Note that we do not explicitly describe the updates to $\mathcal{M}$ after changes to $\mathcal{P}_i$.

In Algorithm 2, we solve $\mathcal{M}$ for the local device problems with DP instead of MIP for efficiency reasons (Section II-C). By convention, the top level planning problem is $\text{solve}(1|\mathbf{0})$.

The parameters k_max and $k^k_\text{r,max}$ control the termination of the outer and inner loop in Algorithm 1. Later in this paper, we will vary these parameters by problem tree depth; we will refer to the inner iteration count at depth d with $k^{(d)}$.

2) General Problem MIP: The MIP $\mathcal{M}$ is the central part of the column generation procedure. We define $\mathcal{M}$ as:

$$\begin{aligned}
\min \quad & \mathbf{c}_\mathbf{x}^\intercal\overline{\mathbf{x}} && && (1)\\
\text{s.t.} \quad & \overline{\mathbf{x}} - \overline{\mathbf{x}}^* - \sum_{i\in\mathcal{I}} \mathbf{x}_i = \mathbf{0} && && (2)\\
& \mathbf{x}_i - \sum_{q\in\mathcal{P}_i} y_{i,q}q.\mathbf{x} = \mathbf{0} && \forall i \in \mathcal{I} && (3)\\
& \mathbf{y}_i^\intercal \mathbf{1} = 1 && \forall i \in \mathcal{I} && (4)\\
& \text{(application constraints)} && && (5)\\
& y_{i,q} \in \{0,1\} \quad \forall i \in \mathcal{I}, && q \in \mathcal{P}_i && (6)\\
& \overline{\mathbf{x}}, \overline{\mathbf{x}}^*, \mathbf{x}_i \in \mathbb{R}^{|\mathbf{c}_\mathbf{x}|} && \forall i \in \mathcal{I} && (7)
\end{aligned}$$

Equation (1) gives the objective value of the problem as the dot product of cost and the (abstract) demand pattern $\overline{\mathbf{x}}$. Next, (2) states that $\overline{\mathbf{x}}$ is equal to the elementwise sum of all demand. The vector $\overline{\mathbf{x}}^*$ gives the demand of the master problem itself, which we use to inject the 'demand' of the local objective. The vectors $\mathbf{x}_i$ give the selected demand for every subproblem $i \in \mathcal{I}$; the possible demand patterns from $\mathcal{P}_i$ are added to $\mathbf{x}_i$ in (3), weighted by a set of indicators $y_{i,q}$ ($q \in \mathcal{P}_i$). These indicators describe whether $\mathcal{M}$ chooses to use pattern q for subproblem i. Equation (4) arranges the mutual exclusion of these indicators for each subproblem; we omit this constraint when $\mathcal{P}_i = \emptyset$, because this leads to the contradiction $0 = 1$. The indicator variables are binaries (6), and the variables $\mathbf{x}$ are real numbers (7).

Until here, all equations are application independent. Equation (5) describes the application-specific aspects of the problem. Users may define auxiliary variables within these constraints, and must define constraints on $\overline{\mathbf{x}}^*$ (e.g. $\overline{\mathbf{x}}^* = \mathbf{0}$). In this work, we reuse the application constraint set from [2].

3) General Problem Algorithm: We will now present the column generation procedure which populates and uses $\mathcal{M}$. The procedure is more complicated than the one outlined in Section II-B1, because our problem contains binary variables.

Algorithm 1 describes the modified procedure. For efficiency reasons, we normally do not run the column generation up to termination; the number of iterations is governed by k_max and $k^k_\text{r,max}$. To keep $\mathcal{M}$ tractable, the work in [2] agressively prunes inactive columns. As a consequence, we need to perform column selection in every iteration. Minimizing $\mathcal{M}$ is computationally expensive. We observe that there is no need to generate a solution in every iteration, as the prices are derived from $\mathcal{M}_\text{r}$. Therefore, we can postpone the column selection problem to the solution recovery phase. To keep the number of binaries in $\mathcal{M}$ low, we limit the number of iterations which can generate patterns to $k^k_\text{r,max}$ (line 4); we also terminate the inner loop when the reduced cost test (line 9) does not admit any new patterns. Alternatively, we could also prune irrelevant patterns based on the column weights $\mathbf{y}_{\text{r},i}$ in s_r.

Algorithm 1 General group column generation

```
Input: Coefficients $\mathbf{c}_\mathbf{x}$ corresponding to $\mathbf{x}$
Output: Set of patterns
 1: $\overline{\mathcal{P}} \leftarrow \emptyset$, $\mathcal{P}_i \leftarrow \emptyset$ for all $i \in \mathcal{I}$
 2: $\mathcal{M}.\mathbf{c}_\mathbf{x} \leftarrow \mathbf{c}_\mathbf{x}$
 3: for $k = 1$ to $k_\text{max}$ do
 4:     for $k_\text{r} = 1$ to $k^k_\text{r,max}$ do
 5:         $s_\text{r} \leftarrow$ minimize $\mathcal{M}_\text{r}$
 6:         for all $i \in \mathcal{I}$ do {in parallel}
 7:             $\lambda_i \leftarrow s_\text{r}.\pi(\mathbf{x}_i)$
 8:             for all $q \in i.\text{solve}(\lambda_i)$ do
 9:                 $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{q\}$ if $(s_\text{r}.\mathbf{x}_i - q.\mathbf{x})^\intercal\lambda_i > 0$
10:             end for
11:         end for
12:     end for
13:     $s \leftarrow$ minimize $\mathcal{M}$
14:     $\overline{\mathcal{P}} \leftarrow \overline{\mathcal{P}} \cup \{\langle\mathbf{x} = s.\mathbf{x}\rangle\}$
15:     $\mathcal{P}_i \leftarrow \{\mathcal{P}_{i,q}$ where $s.y_{i,q} = 1\}$ for all $i \in \mathcal{I}$
16: end for
17: return $\overline{\mathcal{P}}$
```

Algorithm 2 Local problem

```
Input: Coefficients $\mathbf{c_x}$ corresponding to $\mathbf{x}$
Output: Set containing one pattern
1: $\mathcal{M}.\mathbf{c_x} \leftarrow \mathbf{c_x}$
2: $s \leftarrow$ minimize $\mathcal{M}$
3: return $\{\langle \mathbf{x} = s.\mathbf{x} \rangle\}$
```

Algorithm 3 Homogeneous group problem

```
Input: Coefficients $\mathbf{c_x}$ corresponding to $\mathbf{x}$
Output: Set containing one pattern
1: $\mathcal{P}_i \leftarrow i.\,\mathrm{solve}(\mathbf{c_x})$ for all $i \in \mathcal{I}$ {in parallel}
2: $q_i = \arg\min_{q \in \mathcal{P}_i} (\mathbf{c_x}^\intercal q.\mathbf{x})$ for all $i \in \mathcal{I}$
3: return $\{\langle \mathbf{x} = \sum_{i \in \mathcal{I}} q_i.\mathbf{x} \rangle\}$
```

Subsequently, $\mathcal{M}$ selects the best pattern set (line 13). We use this pattern set as the basis for a new column generation phase. The effect of this is twofold. First, the size of the master problem is kept small. Second, the mismatch between the integer and the relaxed solution is (temporarily) eliminated. The disadvantage is that part of the column space has to be re-explored in the inner loop; a less aggressive column pruning strategy may be considered in future work.

C. Practical Improvements

1) Bootstrap Procedure: As in [2], we apply a bootstrap procedure to improve the convergence rate. This procedure modifies λ_i after line 7. This procedure exploits knowledge on what the (approximate) desired profile is; typically, the profile should be as flat as possible, and as close as possible to 0. To reflect the different convergence behavior of the column generation procedure in this work, we replace the static bootstrap iteration limit of [2] with an adaptive limit. The bootstrap pricing stops when the relative objective value improvement per iteration of $\mathcal{M}_\mathrm{r}$ falls below a prespecified value; in the experiments in this paper, we have chosen 1%.

2) Nested Complexity: When we consider solving a problem at some level in the tree as the basic operation, the presented nested column generation algorithm has a computational complexity of $\mathcal{O}(\sum_{d'=1}^{d_\mathrm{max}} \prod_{d=1}^{d'-1} n_\mathrm{g}^{(d)} \prod_{d=1}^{d'} k^{(d)})$: $n_\mathrm{g}^{(d)}$ is the group size at level d, $k^{(d)}$ is the number of iterations at level d ($k^{(d)} = \sum_{k=1}^{k_\mathrm{max}} k_{\mathrm{r,max}}^k$) and d_max is the depth of the problem tree. Due to massive parallelism in the subproblems, we can ignore the product of $n_\mathrm{g}^{(d)}$, and focus on the product of $k^{(d)}$. This part means that the approach appears not to scale well. However, the supported number of bottom subproblems also scales exponentially, according to $\prod_{d=1}^{d_\mathrm{max}} n_\mathrm{g}^{(d)}$. If we are able to keep $k^{(d)}$ and d_max small, the scaling behavior may be acceptable.

Reducing d_max implies that we have to *increase* $n_\mathrm{g}^{(d)}$. This contrasts to the original intent to decrease the group size, such that the column selection MIP becomes smaller. We consider to avoid this problem by using something different than MIP for column selection, which does not have this scalability problem; we discuss this alternative approach later on.

To reduce $k^{(d)}$, we have to find solutions in very few iterations. The bootstrap procedure addresses this in part. For subproblems with a specific structure, we can reduce $k^{(d)}$ to 1 (see Section III-C4).

The column space generated in previous subproblem invocations often serves as a good starting point for the following search. Therefore, we do not clear the pattern set $\mathcal{P}_i$ before every solve (Algorithm 1, line 1). Furthermore, column generation does not require subproblems to give an optimal solution in every iteration; it is merely interested in improving solutions. The procedure formally only requires an optimal solution in the final iteration, which certifies that no further columns with reduced costs exist. Combining column space reuse with the use of suboptimal columns suggests a communicating cascaded design: the master problem continually updates the prices, and the subproblems supply improving columns according to these prices. In this design, we can trade off between $k^{(1)}$ and $k^{(d)}$ ($d > 1$), where $k^{(1)}$ can be interpreted as 'sweeps' over the problem tree. Since $k^{(d)}$ can now be a lot smaller, the overall effort can be reduced to an acceptable level.

3) Approximate Solutions: Next to reducing $k^{(d)}$, we can also reduce the effort per iteration; due to the nested structure, this is particularly useful for the lower problems. Only the final iteration needs an optimal solution; the rest may use any mechanism giving feasible patterns with positive reduced cost.

For the middle and top level problems, this means that we can often avoid solving the column selection problem in line 13 to optimality. We replace the MIP with a maximum-weight selection on $\mathbf{y}_{\mathrm{r},i}$: for every $i \in \mathcal{I}$, we define $q_{\mathrm{r},i} = \arg\max_{q \in \mathcal{P}_i} y_{\mathrm{r},i,q}$. We use $\mathbf{q}_\mathrm{r}$ as a guess for the best integral combination of patterns. For notational convenience, we denote the problem of evaluating $\mathbf{q}_\mathrm{r}$ as $\mathcal{M}_{\mathbf{q}_\mathrm{r}}$. In Section IV-C, we show that $\mathcal{M}_{\mathbf{q}_\mathrm{r}}$ can be used as a good starting approximation of $\mathcal{M}$.

4) Independent Subproblems: If the subproblems at some depth do not interact, we can largely avoid the column generation procedure, since the subproblem price vector will not change. In particular, we consider the case where the application constraints (5) in $\mathcal{M}$ only contain $\overline{\mathbf{x}}^* = \mathbf{0}$, i.e. $\mathcal{M}$ only adds up the found patterns with no extra constraints or local costs. In that case, all subproblem price vectors λ_i will be equal to $\mathbf{c_x}$. Therefore, we only need one column for each subproblem. This column should be optimal for the original price vector. The sum of these columns provides an optimal assignment of $\overline{\mathbf{x}}$ for this $\mathbf{c}$. Because this case is very common in TRIANA, we use a specialized procedure.

Algorithm 3 presents this specialized procedure. In line 1, we send $\mathbf{c_x}$ to all subproblems, and we gather the corresponding subproblem solutions. The subproblems may also provide supplementary solutions which are suboptimal with respect to $\mathbf{c_x}$; therefore, we select the best solution q_i for each i in line 2. Finally, line 3 constructs and returns the solution from $\mathbf{q}$. Note that, for this special case, we do not need an iterative process, nor do we need to generate prices.

Algorithm 3 can be used directly as the top level problem to perform a simple price based optimization ($\mathbf{c_x} = \alpha|\lambda$, where α is the relative weight of the local objective), or to aggregate separate problems ($\mathbf{c_x} = 1|\mathbf{0}$).

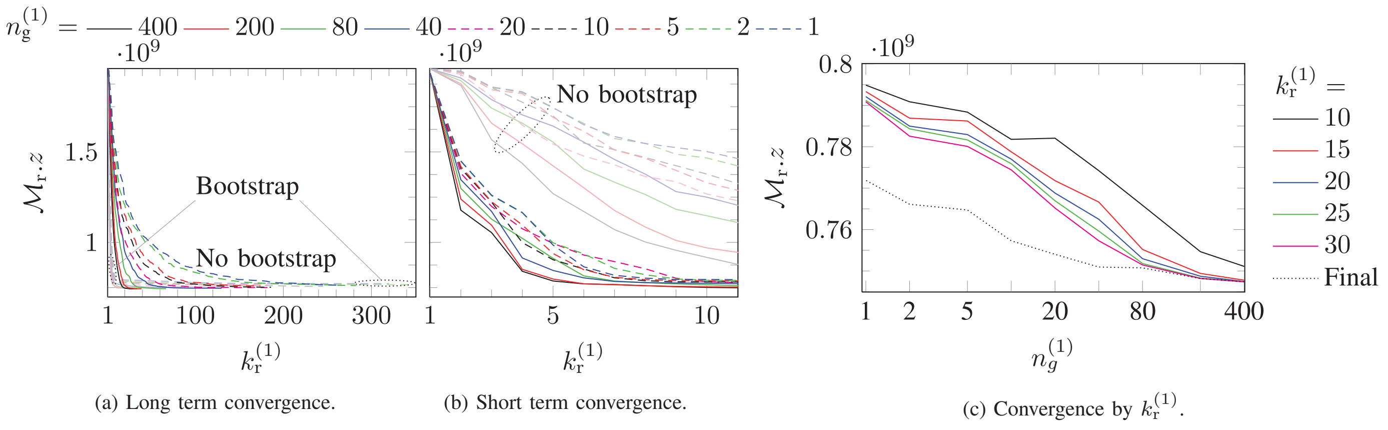


Fig. 3: Objective convergence of the top level problem, by group and iteration count.

IV. Simulation Experiments

A. Experiment Setup

To evaluate the presented techniques, we use the 400–house FLEX STREET scenario [20]. This scenario considers DSM using many domestic appliances, including EVs, heat pumps, batteries and washing machines. We use the same combined demand peak and variation minimization objective as in [2].

As indicated in the previous section, we aim to reduce the number of iterations $k^{(d)}$ used at each level. Therefore, we are interested in the convergence behavior of the nested column generation procedure, subject to different groupings. We are also interested in the impact of the integrality constraints.

To have an interesting scenario without too much computational effort, we choose the start of the evening of the first day as the period of interest for the experiments. This time period gives sufficient time to avoid start-up problems, yet covers a hard to schedule period: the demand profile must ramp up from a mid-day PV supply valley to the evening heat demand peak. To have an equal set up for all experiments, we always run the simulation up to the start of the chosen period with a baseline control method.

B. Group Size, Bootstrap Versus Convergence Behavior

As pointed out in Section III-C2, a low iteration count is essential to make the problem scalable. Therefore, we evaluate the convergence behavior; we run the algorithm until it terminates ($k_{\max} = 1$ and $k_{\mathrm{r},\max} = \infty$). To evaluate the importance of group size, we make $n_\mathrm{g}^{(1)}$ groups of $n_\mathrm{g}^{(2)} = \frac{400}{n_\mathrm{g}^{(1)}}$ houses. For this experiment, we apply the method described in Section III-C4 at both the lower $(d = 2)$ and the house level $(d = 3)$. For the results, we expect that a smaller number for $n_\mathrm{g}^{(1)}$ makes the master MIP much easier, but we need to generate more columns. The cases $n_\mathrm{g}^{(1)} \leq 10$ reflect the optimization of the behavior of an individual house; the larger cases represent the optimization of a neighborhood.

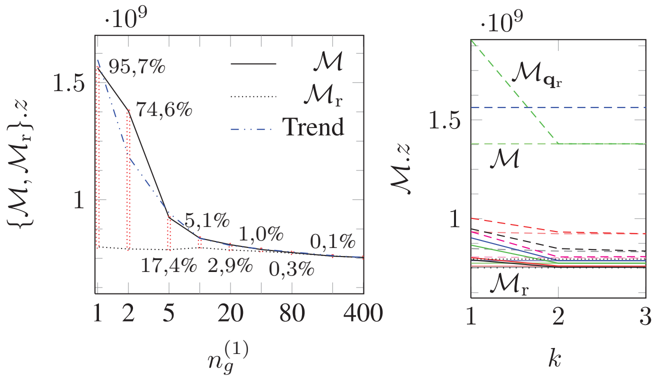
Fig. 3 presents the long term and short term convergence behavior of the top level problem; it plots the objective value of $\mathcal{M}_\mathrm{r}$ against k_r for various choices of $n_\mathrm{g}^{(1)}$. As can be seen from the graphs, the column generation procedure has a very long tail with marginal improvement; therefore, we present the long term and the short term behavior separately. The bootstrap method presented in Section III-C1 has a dominant effect on the short term convergence behavior. To illustrate this effect clearly, Fig. 3a focuses on the case without bootstrap, whereas Fig. 3b focuses on the case with bootstrap; in each graph, semitransparent lines represent the other case.

Fig. 3a demonstrates that, without bootstrap, a large number of subproblems $n_\mathrm{g}^{(1)}$ improves the convergence rate significantly. The column generation procedure can consider the smaller problems separately, which increases the flexibility of the master problem. For smaller $n_\mathrm{g}^{(1)}$, the master problem needs to request a new column to combine the columns from a lower level in a different way. Nevertheless, column generation manages to find a good relaxed solution even when all houses are lumped into a single group. With the bootstrap procedure, the number of iterations to termination slightly decreases for large $n_\mathrm{g}^{(1)}$, and increases for small $n_\mathrm{g}^{(1)}$; a possible explanation for this is that this procedure initially does not follow the structure of the problem, which can be good or bad.

The short term results (Fig. 3b) look quite different. With bootstrap, the difference is much smaller; for all considered group sizes, the procedure already converges in 5–10 iterations. At $k_\mathrm{r} = 10$, the objective difference between the smallest and the largest group size is 6%. Regardless of group size, subsequent iterations offer only marginal improvement; the largest extra improvement is found for $n_\mathrm{g}^{(1)} \geq 10$. To make this more clear, Fig. 3c considers the progress at specific iterations in the process. Problems with a large $n_\mathrm{g}^{(1)}$ progress towards the final result far more quickly; for small $n_\mathrm{g}^{(1)}$, this process is very slow.

C. Group Size Versus Integers

In Section IV-B, we found that already after $k_\mathrm{r} = 10$ iterations a good linear combination of patterns is found. However, we need a good *selection* of patterns: of every subproblem, exactly one pattern must be chosen. We expect that the integrality constraint is harder for smaller groups, because it may be more difficult to find a good selection of patterns due to limited diversity.



(a) Objective penalty of column selection, by group count ($k_\mathrm{r}^{(1)} = 10$, $k = 1$).

(b) Objective penalty, by column selection method and $n_\mathrm{g}^{(1)}$.

Fig. 4: Objective penalty resulting from integrality.

To investigate this, we present the objective value penalty which results from the integrality constraint in Fig. 4a. For smaller $n_\mathrm{g}^{(1)}$, this penalty is very large; it becomes very small for larger $n_\mathrm{g}^{(1)}$. We observe the following trend (also depicted in Fig. 4a): the objective penalty is almost equal to $1/2n_\mathrm{g}^{(1)}$ for $n_\mathrm{g}^{(1)} \geq 10$, and around $1/n_\mathrm{g}^{(1)}$ for $n_\mathrm{g}^{(1)} < 10$; a possible explanation for the difference is that for small $n_\mathrm{g}^{(1)}$ the relaxed solution has not yet settled.

Interestingly, the MIP becomes *harder* rather than easier with a smaller number of subproblems. The number of columns per subproblem increases: the number of iterations increases, and the probability of adding a new column becomes larger. Furthermore, the solver can no longer consider the smaller problems separately, which reduces the flexibility of the master problem. This makes it harder to meet the MIP gap limit (which is set to 1%).

In Section III-B3, we consider to keep only the current selected columns from one iteration of k to the next to eliminate the mismatch between the linear and the integer solution. The simulation results in Fig. 4b show that removing this mismatch does not give any substantial improvement. We believe that no price vector maps to a suitable profile when $n_\mathrm{g}^{(1)}$ is too small.

In Section III-C3, we furthermore propose to replace the MIP with a maximum-weight selection. To avoid generating the same problem over again, we limit $k_\mathrm{r}^{(1)}$ to 2 for $k \geq 2$. Fig. 4b includes the results of this approach (labeled $\mathcal{M}_{\mathbf{q}_\mathrm{r}}$). At $k = 1$, the objective value is a lot worse than for MIP: even for large $n_\mathrm{g}^{(1)}$, adding the highest-weight patterns together often gives a poor solution. However, at $k = 2$, the method recovers the patterns it should not have removed, which almost fully eliminates the difference with $\mathcal{M}$. This means that we can practically choose not to use the MIP altogether; consequently, the problem becomes much easier computationally. This trades the effort on the master problem with extra subproblem effort.

D. Cascaded Column Generation

In a last series of experiments, we want to evaluate the behavior of column generation with a cascaded problem configuration, as described in Section III-C2. The main parameter is $k_{\mathrm{r,max}}^{(d)}$, which determines the branching per level; it should be chosen as low as reasonably possible. We control the number of iterations at the top level ($k_{\mathrm{r,max}}^{(1)}$) separately. We partition the group of 400 houses in 20 groups of 20 houses ($n_\mathrm{g}^{(1)} = 20$), and we take the same global objective as in Section IV-B. For each of the 20 groups, we also use this objective, which means that we account for local peaks and demand changes in the network with the same weight. In this case, the local objective clearly supports the global objective: we expect that schedules which are good on a local scale are together also good globally. The method does not depend on this support, but it does improve the convergence speed.

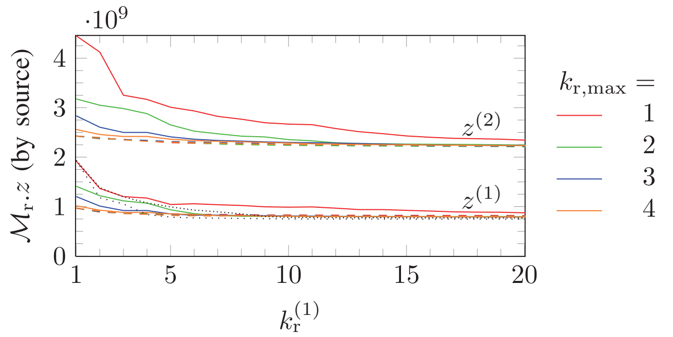


Fig. 5: Joint objective convergence of the top level problems and the subproblems by $k_{\mathrm{r,max}}$.

Fig. 5 presents the results of these simulations. We break down the objective value by source: the top curves correspond to the sum of the subproblem objective values (denoted $z^{(2)}$), and the bottom curves represent the 'top level' objective value ($z^{(1)} = \mathcal{M}_\mathrm{r}.z - z^{(2)}$). Due to reduced economy of scale at the subproblem level, $z^{(2)}$ is greater than $z^{(1)}$. For reference, we also include the results of Section IV-B corresponding to $z^{(1)}$ for $n_\mathrm{g}^{(1)} = 20$ and $n_\mathrm{g}^{(1)} = 400$ with dotted lines.

The results show that $k_{\mathrm{r,max}} = 1$ gives very slow convergence: the subproblems have little room to explore solutions which are acceptable for both the master and the local objective. For $k_{\mathrm{r,max}} = 2$ and higher values, the results are far better; however, the complexity scales exponentially by depth. Despite that the global and the local problems use the same objective, there are conflicts between the interests of the master problem and the subproblems; as a result, the values of $z^{(1)}$ and $z^{(2)}$ are not monotonically decreasing. To further point out this conflict, it should be noted that the value of $z^{(1)}$ in the found solution is 5% higher than in Section IV-B.

We observe that the problem spends a lot of top level iterations for the local optimization of its subproblems, which the system can also solve independently. Therefore, it makes sense to first optimize the system locally with for example $k_{\mathrm{r,max}} = 20$ before doing the cascaded optimization with a low $k_{\mathrm{r,max}}$. Fig. 5 includes the results for this with dashed lines. In this case, the high-k_r local optimization almost solves the top level problem. In the following iterations, a diverse column set is already available, so there is little need to generate new columns in the subproblems; consequently, $k_{\mathrm{r,max}}$ becomes unimportant—even $k_{\mathrm{r,max}} = 0$ may work well, provided that good columns are available for the problem at hand.

V. Evaluation

Column generation again proves to be a very effective and flexible scheduling approach. However, to find solutions in a reasonable number of iterations, several specific changes are necessary; this paper extends the set of changes of [2] for smaller groups and for hierarchical planning. Furthermore, we removed the main bottleneck (the master MIP) which prevented the use of the approach for larger problems; the results show that now larger instances can be tackled, but as currently no larger scenarios are available, we are not able to investigate the upper limits in size.

A *nested* column generation approach gives a prohibitively large problem, as the size of the problem grows exponentially by the iteration count with the depth. Instead, we use a *cascaded* column generation approach, which reuses earlier subproblem solutions to reduce the iteration count. Furthermore, we can locally combine solutions in a different way without consulting other subproblems, which avoids the growth in complexity.

To be practical, the approach needs extra information, especially for smaller groups. The bootstrap procedure defines the start of the column generation search; a good entry point can avoid a large part of the search. This procedure exploits knowledge on what the (approximate) desired profile is. Alternatively, the local problems can be provided with more information about the global problem; this substantially accelerates the solution process (Section IV-D).

The main problem of the column generation approach, which makes the aforementioned changes necessary, is that the approach fails to communicate its needs to the subproblems in an effective way. Prices provide a convenient narrow interface, but these only represent the currently active set of constraints, and not the constraints which (may) become active in a later iteration. Price optimization leads to excessive responses (oscillating behavior), unless there are local incentives to prevent this. As an alternative, it may be better to request a desired profile rather than a price vector response; in the case of Section IV-C, one may instruct the subproblems to generate the solution of the relaxed problem.

VI. Conclusion

Cascaded column generation is a promising approach for solving large scale energy scheduling problems. A hierarchical structure results in a scalable approach. However, the group size and iteration count at each level must be chosen with care, because these affect the effectiveness of the approach. The practical lower group size limit is approximately 10 subproblems; the upper limit has not been reached yet. A large group size (≥ 100) makes the search easier, since the problem has more freedom to combine solutions, but this can also make the master problem harder.

An unexpected but very practical contribution of this work is an iterative column selection method, which allows us to remove the mixed integer program at the expense of extra work in the subproblems. As a result, we can handle large groups more easily than originally intended.

For future work, we propose several improvements. Instead of sending prices, we may choose to communicate part of the objective of the master problem more directly. For example, the master problem can determine a target profile for the subproblem, which the subproblem can schedule independently. Also, subproblems may expose more than only the electricity profile to make the scheduling problem easier.

Acknowledgment

This research is conducted within the DREAM project supported by STW (#11842).

References

[1] A. Molderink, V. Bakker, M. G. C. Bosman, J. L. Hurink, and G. J. M. Smit, "Management and control of domestic smart grid technology," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 109–119, Sept 2010.

[2] H. A. Toersche, A. Molderink, J. L. Hurink, and G. J. M. Smit, "Column generation based planning in smart grids using TRIANA," in *Innovative Smart Grid Technologies (ISGT) Europe, IEEE PES*, Lyngby, Denmark, October 2013.

[3] G. B. Dantzig, *Linear Programming and Extensions.* Princeton U.P., 1963.

[4] D. Hammerstrom, T. Oliver, R. Melton, and R. Ambrosio, "Standardization of a hierarchical transactive control system," *GridInt*, vol. 9, 2009.

[5] M. Hommelberg, B. van der Velde, C. Warmer, I. Kamphuis, and J. Kok, "A novel architecture for real-time operation of multi-agent based coordination of demand and supply," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, July 2008, pp. 1–5.

[6] M. Ghijsen and R. D'hulst, "Market-based coordinated charging of electric vehicles on the low-voltage distribution grid," in *Smart Grid Modeling and Simulation (SGMS), 2011 IEEE First International Workshop on*, 2011, pp. 1–6.

[7] R. Halvgaard, N. Poulsen, H. Madsen, and J. Jorgensen, "Economic model predictive control for building climate control in a smart grid," in *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*, 2012.

[8] D. Phan and J. Kalagnanam, "Distributed methods for solving the security-constrained optimal power flow problem," in *Innovative Smart Grid Technologies (ISGT), IEEE PES*, January 2012, pp. 1–7.

[9] R. Scattolini, "Architectures for distributed and hierarchical model predictive control—a review," *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, 2009.

[10] M. Juelsgaard, L. C. Totu, S. E. Shafiei, R. Wisniewski, and J. Stoustrup, "Control structures for smart grid balancing," in *Innovative Smart Grid Technologies (ISGT) Europe, IEEE PES*, Lyngby, Denmark, Oct. 2013.

[11] P. Mc Namara and S. McLoone, "Hierarchical demand response using Dantzig–Wolfe decomposition," in *Innovative Smart Grid Technologies (ISGT) Europe, IEEE PES*, Lyngby, Denmark, October 2013.

[12] L. Sokoler, K. Edlund, L. Standardi, and J. Jørgensen, "A decomposition algorithm for optimal control of distributed energy system," in *ISGT Europe, IEEE PES*, Lyngby, Denmark, Oct. 2013.

[13] L. Standardi, N. K. Poulsen, J. B. Jørgensen, and L. E. Sokoler, "Computational efficiency of economic MPC for power systems operation," in *ISGT Europe, IEEE PES*, Lyngby, Denmark, October 2013.

[14] C. R. Glassey, "Nested decomposition and multi-stage linear programs," *Management Science*, vol. 20, no. 3, pp. 282–292, 1973.

[15] M.-C. Noël and Y. Smeers, "Nested decomposition of multistage nonlinear programs with recourse," *Math. Prog.*, vol. 37, no. 2, pp. 131–152, 1987.

[16] V. Bakker, "TRIANA: a control strategy for smart grids," Ph.D. dissertation, University of Twente, January 2012.

[17] M. G. C. Bosman, "Planning in smart grids," Ph.D. dissertation, University of Twente, Enschede, July 2012.

[18] A. Molderink, "On the three-step control methodology for smart grids," Ph.D. dissertation, University of Twente, May 2011.

[19] M. G. C. Bosman, V. Bakker, A. Molderink, J. L. Hurink, and G. J. M. Smit, "Planning the production of a fleet of domestic combined heat and power generators," *European journal of operational research*, vol. 216, no. 1, pp. 140–151, July 2011.

[20] F. N. Claessen, B. Claessens, M. P. F. Hommelberg, A. Molderink, V. Bakker, H. A. Toersche, and M. A. van den Broek, "Comparative analysis of tertiary control systems for smart grids using the Flex Street model," *Renewable Energy*, 2014, accepted.