

# Column Generation Based Planning in Smart Grids Using TRIANA

Hermen A. Toersche, A. Molderink, J.L. Hurink and G.J.M. Smit  
Faculty of Electrical Engineering, Mathematics and Computer Science  
University of Twente  
Enschede, the Netherlands  
h.a.toersche@utwente.nl

**Abstract**—Column generation is applied to the TRIANA smart grid framework's planner and combined with distributed dynamic programming. Simulations show that this improves the resulting planning compared to methods described in previous work (extra peak reduction of 7.3%; large fluctuation reduction) at the cost of additional processing.

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

## I. INTRODUCTION

SMART GRIDS promise to revolutionize the way electrical power is transported and used. Flexible transmission systems and controllable assets, including those owned by end users—such as heat pumps, EVs and household appliances—will increase the ability to manage the grid from the demand side. Using this potential, the needed capacity investments in the near future, due to increased electrification and renewable generation, may be reduced. Furthermore, systems can respond better to markets and losses can be reduced.

To deliver on this promise, effective and coordinated control of these assets is essential. In contrast to many related works on smart grid control, we believe that smart grids should not be scheduled reactively, but actively—as is the case for the current grid. The reason for this is that the new assets also have states and by that have dependencies over time: limits on for example run length, ramp rates and departure times (of EVs). Therefore, a reactive control policy makes it difficult to control the indirect consequences of actions, i.e. effects on future flexibility.

For the current grid, these scheduling and control problems are well known, have been well studied and are practically solved; see for example [1]. However, for smart grids several complications arise. Most importantly, the expected number of dispatchable assets is much larger, typically ranging up to millions rather than hundreds. This means that centralized scheduling is not an option. Furthermore, since the controlled power per asset is relatively low, the scheduling effort needs to be limited, such that its benefits clearly outweigh its costs.

In earlier work, we presented what is now known as TRIANA, a general smart grid optimization framework [2]. It uses a hierarchical—and thereby scalable—scheduling approach. It plans both demand and supply, guided by forecasts of

expected behavior. To resolve forecast errors, furthermore an operational (real time) control is added. This three-step approach of forecasting–planning–operation gives good results in comparison with other control approaches [3].

For the TRIANA approach, we observe that the operational control policy degenerates to reactive control if the plan is not revised frequently [4]. To avoid this problem, more freedom must be given to the local controller; this may be realised by increasing the frequency at which the planning process is redone (or just revised) to be able to account for the long-term effects of real time control choices. However, the current planning approach (namely IDDP—iterative distributed dynamic programming [2]) may not allow such improvements without degrading solution quality. Although single IDDP iterations are fast and lead to rapid result improvements in the initial phase, it fails to achieve any substantial improvements in later iterations. We are therefore exploring new ways of making the planning process more effective.

Column generation [5] is a well-known technique from the domain of Operations Research that is used to solve very large linear optimization problems. It has been demonstrated to be highly effective for many problems that involve limited coupling of different entities—a description that applies to smart grids. Therefore, we previously investigated the use of column generation in a specific smart grid unit commitment problem [6].

In this work, we adopt column generation more generally for the distributed planning problem in the TRIANA framework. To validate and demonstrate the resulting approach, termed CGDDP—*column generation with distributed dynamic programming*, we reuse the 400-house FLEX STREET case presented in [7]. Whereas [7] compares TRIANA using IDDP to a reactive control approach, this work uses the case as a reference scenario to both discover and illustrate the features of CGDDP.

This paper is structured as follows. In Section II, we provide more background on related control approaches and on TRIANA, followed by a short review of column generation. Next, Section III presents the approach taken to integrate column generation in TRIANA, together with the corresponding formal definitions and design considerations. Subsequently, simulation results of our implementation of CGDDP are presented and interpreted in Section IV. Finally, conclusions are drawn in Section V.

## II. BACKGROUND

### A. Smart Grid Control

Smart grid related problems have been approached by applying control engineering theory, such as Model Predictive Control; see for example [8]. Such approaches often focus on the efficient operation of the devices themselves under a given pricing scheme, rather than optimizing for aggregate behavior. Furthermore, the number of considered appliances is often small (or even just one), and even when multiple appliances are considered, they are all of the same type. These factors result in both conceptual and computational scalability problems.

Integrated smart grid control approaches address these problems by partitioning the optimization problem both conceptually (by supporting more than one type of device) and computationally (by partitioning the optimization problem and allowing workload distribution). Explicit prediction is typically sacrificed. The major vehicles of abstraction are prices on the one hand—indicating the relative desirability of power consumption—and power consumption levels on the other hand. Direct price steering is well known to result in uncontrolled demand response: all devices respond to the lowest price. Therefore, approaches such as GridWise [9], PowerMatcher [10] and Intelligator [11] have adopted a transactive control approach. Transactive control is typically implemented with an on-line double-sided auction mechanism, in which a dynamic priority ordering determines how the available capacity is assigned. Explicit prediction is computationally hard to be added to a double-sided auction: due to dependencies, it does not scale beyond a few intervals. Therefore, to account for the future, the priority ordering is in part determined by an estimate of the future system state. Transactive controllers are, despite their reactive design, quite effective in terms of the most often considered objective: coordinated peak shaving [7].

In TRIANA [2], these perspectives are combined alongside legacy grid control: globally, transport and large generation resources are considered, whereas locally device behavior is optimized subject to the demands of the global problem. For scalability, we exploit the hierarchical structure of the problem (Fig. 1). The coordination aspect is handled in a scalable manner by an iterative feedback mechanism [12], which can be seen as a substitute for a double-sided auction with lookahead. The described work in this paper involves replacing this mechanism with column generation.

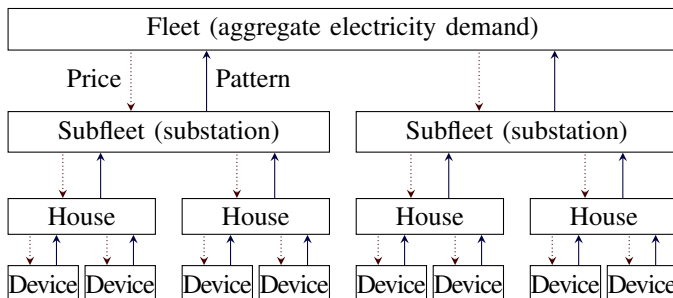


Fig. 1. Partitioned optimization approach in TRIANA.

### B. Column Generation

Linear programming (LP) is one of the most widely used techniques for mathematical optimization: many problems can be naturally expressed with it, and efficient (polynomial) methods exist to solve such problems. Even more problems can be expressed by applying (Mixed) Integer Programming (MIP), though this comes at the cost of increasing computational cost—in many cases even exponentially. As a result, solving large MIPs directly is infeasible. This means that nontrivial problems need to be modeled with care, in particular such that the number of integer variables is kept small.

Rather than considering a complete large LP problem at once, Dantzig–Wolfe decomposition [5] may be used to separate a given problem into its block structure, which can then be re-composed by means of column generation. The master problem is concerned with a set of connecting variables and associated constraints. A series of subproblems is concerned with their own set of local variables and constraints. The master problem uses the subproblems to explore new *columns* of the original solution space by sending *shadow prices* to its subproblems. These prices are derived from the linear program structure and express the relative objective importance of the shared variables. The subproblems must be optimized subject to these prices.

In the case of linear programming, this approach gives optimal results: when no improving columns are found by the subproblems, the given solution is optimal for the original problem. However, in the case of MIP, further effort is needed to guarantee optimality [13]. Since we need to avoid computational effort, we do not consider this in this work.

## III. COLUMN GENERATION FOR TRIANA

### A. Overview

As became clear in Section II-B, the first step in column generation (after defining the original problem) is to identify the master problem, the subproblems and their connecting variables. Given the hierarchical architecture of the IDDP approach, this step can be made with little effort: the subnodes (substations, houses, devices) map to subproblems; their demand levels over each time interval are the connecting variables. The master problem relates to aggregate demand, i.e. the sum of demand levels over each time interval. Though other relations can be considered later on, this small set of connecting variables is sufficient for many applications, including peak shaving, imbalance reduction and virtual power plant operation (the same applications as IDDP was applied to).

In IDDP, the top-level and middle-level problems are solved using a randomized algorithm. The bottom-level subproblems are solved to optimality by dynamic programming (DP), i.e. by state-space exploration. Using DP is much more efficient or sometimes even the only option when nonlinear constraints are prevalent, yet is functionally almost equivalent to MIP. The dynamic programs are solved in parallel for performance reasons; these programs can be solved locally for reduced communication overhead (this also protects privacy to some extent). The problems at the different levels are iteratively solved in succession: the higher problems generate prices; in

turn, the lower problems optimize demand according to these prices. The higher problem then revises the prices according to observed aggregate demand.

In this work, the basic problem can be described as a MIP similar to [6], though we do not need to make this MIP explicit. We derive a column generation master problem, which will be described in Section III-B. Next, Section III-C reformulates the corresponding subproblems such that these can be optimized by dynamic programming, allowing them to be solved efficiently. Except for the *values* of the prices, the lowest subproblems are now equal to the subproblems of IDDP.

Analogous to IDDP, we refer to this method as CGDDP: column generation with distributed dynamic programming. The ‘iterative’ of IDDP has been dropped because column generation is iterative by definition.

This work addresses both the top level and the bottom level of the hierarchical approach. To accommodate larger problems or to decrease the time used to reach a solution, the hierarchical structure can be used.

### B. Master Problem

1) *Variables and Objective:* Let  $x^{(k)}(t)$  be the aggregate demand at time interval  $t$  in iteration  $k$ : these are the central variables in this formulation. Let  $i \in \mathcal{I}$  be the set of subproblems (houses);  $\mathcal{P}_i^{(k)}$  is the set of *patterns* provided by  $i$  in iteration  $k$ ; a pattern specifies the demand in each time interval. These pattern sets are initially empty:  $\mathcal{P}_i^{(0)} = \emptyset$ . Variables  $y_{q_i}^{(k)}$  indicate whether a particular pattern  $q_i \in \mathcal{P}_i^{(k)}$  is selected. Exactly one pattern must be selected for every  $i$  in each  $k$ . The selected pattern  $q_i$  contributes the pattern demand  $p_{q_i}(t)$  to  $x^{(k)}(t)$ .

Next, we define an objective similar to that used within the IDDP approach. Since multiple objectives are involved, we consider their linear combination. The main objective is minimizing peak demand. Therefore, we introduce a variable  $x_{\max}^{(k)}$  which expresses the maximum of all  $x_t^{(k)}$  (corresponding weight:  $w_{P_{\max}}$ ). Next, the difference between the maximum and minimum demand should be minimized. This difference is expressed by a variable  $x_{\text{down}}^{(k)}$  (weight  $w_{P_{\text{down}}}$ ).

Subsequently, rapid demand changes should be avoided: if this is not added explicitly, CGDDP may provide impractical solutions with instantaneous demand changes. To prevent this, we introduce variables  $d_t^{(k)}$  which express the absolute difference between consecutive values of  $x$ . The sum of these values is added to the objective (weight  $w_D$ ). Furthermore, a variable denoted as  $d_{\max}^{(k)}$  is used to express the maximum value of all absolute differences (weight  $w_{D_{\max}}$ ).

This formulation avoids large demand changes; however, if a large demand change can not be avoided, it does not spread the smaller demand changes. Indeed, the search tends to find ‘optimal’ solutions with step changes between demand plateaus. This could be solved by a nonlinear (e.g. quadratic) cost function, or an approximation thereof; however, this is computationally expensive. Instead, we estimate the ‘desired’ demand pattern: for a short horizon  $N'_t$ , we define  $x_{t'}^{(k),\alpha}$  as a line with slope  $\alpha^{(k-1)}$ , starting at the current demand level  $x_0$ .  $N'_t$  should be chosen small enough to form a good fit, yet large

enough to be useful in guiding the profile. We subsequently minimize the sum of distances  $d_{t'}^{(k),\alpha}$  (weight  $w_{D^\alpha}$ ) and the peak distance  $d_{\max}^{(k),\alpha}$  (weight  $w_{D_{\max}^\alpha}$ ) to this pattern. The value of  $\alpha$  can be derived by the MIP; however, for efficiency we use the value which best fits the solution of the previous iteration: it stabilizes in a few iterations. Although this approach is affected by a similar problem as the difference sum approach, it is far more likely to find sensible solutions.

The primary objectives (minimization of the peak demand and the maximum-to-minimum demand difference) are considered over a planning horizon of  $N_t$  intervals. The other objectives are evaluated over a horizon  $N'_t$ ; to decrease the problem size, this also limits the number of difference variables. This horizon should be longer than the planning interval  $N_e$ , but not longer than the planning horizon:  $N_e \leq N'_t \leq N_t$ .

2) *Problem Formulation:* The resulting master problem is specified as follows (for a specific  $k$ ). For conciseness and clarity, common transformations are omitted.

$$\begin{aligned} \min \quad & w_{P_{\max}} x_{\max}^{(k)} + w_{P_{\text{down}}} x_{\text{down}}^{(k)} \\ & + w_D \sum_{t'} d^{(k)}(t') + w_{D_{\max}} d_{\max}^{(k)}(t') \end{aligned} \quad (1)$$

$$\begin{aligned} \text{s.t.} \quad & x^{(k)}(t) = \sum_i \sum_{q_i} y_{q_i}^{(k)} p_{q_i}(t) \quad \forall t \end{aligned} \quad (2)$$

$$\sum_{q_i} y_{q_i}^{(k)} = 1 \quad \forall i \quad (3)$$

$$x_{\max}^{(k)} \geq x^{(k)}(t) \quad \forall t \quad (4)$$

$$x_{\text{down}}^{(k)} \geq x_{\max}^{(k)} - x^{(k)}(t) \quad \forall t \quad (5)$$

$$d^{(k)}(t') = |x^{(k)}(t') - x^{(k)}(t' - 1)| \quad \forall t' \quad (6)$$

$$d_{\max}^{(k)} \geq d^{(k)}(t') \quad \forall t' \quad (7)$$

$$x^{(k),\alpha}(t') = x_0 + \alpha^{(k-1)} t' \quad \forall t' \quad (8)$$

$$d^{(k),\alpha}(t') = |x^{(k)}(t') - x^{(k),\alpha}(t')| \quad \forall t' \quad (9)$$

$$d_{\max}^{(k),\alpha} \geq d^{(k),\alpha}(t') \quad \forall t' \quad (10)$$

$$y_{q_i}^{(k)} \in \{0, 1\} \quad \forall i, q_i \quad (11)$$

$$\alpha^{(k-1)} \in \mathbb{R} \quad (12)$$

$$t \in \{1 \dots N_t\}, \quad t' \in \{1 \dots N'_t\}, \quad i \in \{1 \dots N_i\} \quad (13)$$

3) *Patterns and Pricing:* After solving the master problem, new patterns are generated within the subproblems. The subproblem patterns are derived with the local DPS, subject to a price vector  $\lambda_i^{(k-1)}$  (see Section III-C). Because the patterns are homogeneous (i.e. simply added), we can omit the indexing by  $i$  and derive  $\lambda^{(k-1)}$  from the shadow prices  $\pi^{(k-1)}$  of (2) in the relaxation of the MIP from Section III-B2. This implies that, unlike for IDDP, the pricing is the same for all subproblems: column generation provides the needed price diversity implicitly. As the first price vector, we use  $\lambda^{(0)} = \langle -1, \dots, -1 \rangle$ , which should avoid over-consumption during the first iterations.

The generated pattern  $q_i^{(k)}$  is only added to  $\mathcal{P}_i^{(k)}$  if the reduced cost is positive. The search ends when no patterns are added, or when a limit  $N_k$  on  $k$  is reached. Patterns which have not been used or generated recently are pruned (LRU).

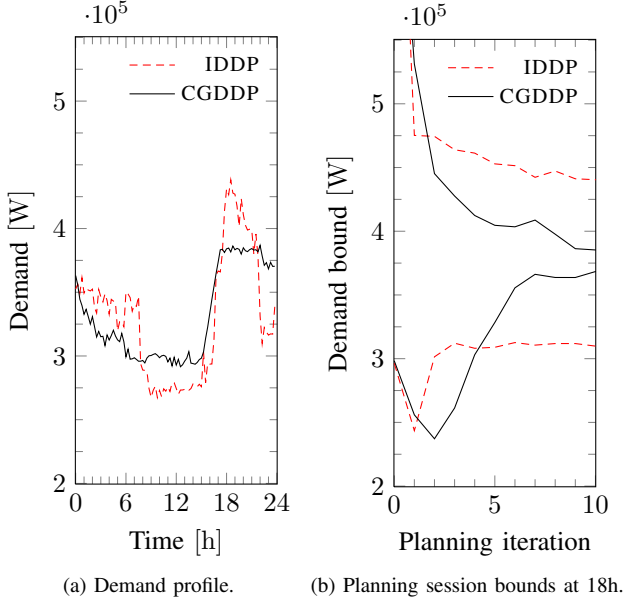


Fig. 2. Difference between planning with IDDP and CGDDP.

### C. Subproblem

1) *Problem Formulation*: The optimization subproblem used to generate new patterns is defined as:

$$\max (\lambda^{(k-1)})^\top (p'_{q'_i} - p_{q_i}) \quad (14)$$

$$\text{s.t.} \dots \quad (15)$$

Here,  $q'_i$  is the new pattern which has demand  $p'_{q'_i}(t)$ . The current pattern of  $i$  is  $q_i$ , which has demand  $p_{q_i}$ . The constraints in (15) are device-specific and are not of relevance to this work.

We may simplify the problem by increasing (14) by  $(\lambda^{(k-1)})^\top p_{q_i}$ , which is a constant. Next, we choose to negate  $\lambda^{(k-1)}$  and solve the equivalent minimization problem:

$$\min (-\lambda^{(k-1)})^\top p'_{q'_i} \quad (16)$$

$$\text{s.t.} \quad (15)$$

For the reduced cost (see Section III-B3), we now require the objective value to be smaller than  $-(\lambda^{(k-1)})^\top p_{q_i}$ .

2) *Bootstrap Procedure*: Preliminary simulations show that the general column generation formulation is often inefficient at finding good starting patterns due to overshoot. To reduce this, we modify the price vector during the first  $N_b$  iterations of the master problem. We use  $x^{(k-1)}(t)$  as a heuristic to predict the relative demand weight of the time intervals: the corresponding element of  $\lambda^{(k-1)}$  is decreased by  $B^{(k)} \frac{(N_b - k + 1)}{N_b} x_t^{(k-1)}$  for  $1 \leq k \leq N_b$ , where  $B^{(k)}$  is chosen such that  $\frac{B^{(k)}}{x_{\max}^{(k-1)}} \geq w_{P_{\max}}$ .

When a modified price vector is used (i.e.  $k \leq N_b$ ), the reduced cost termination condition is not applied, because the condition may falsely conclude that improving patterns no longer exist.

## IV. SIMULATION: RESULTS AND EVALUATION

### A. Flex Street Case

To demonstrate and validate CGDDP, we used the 400-house FLEX STREET-MODERATE year simulation case described in [3]

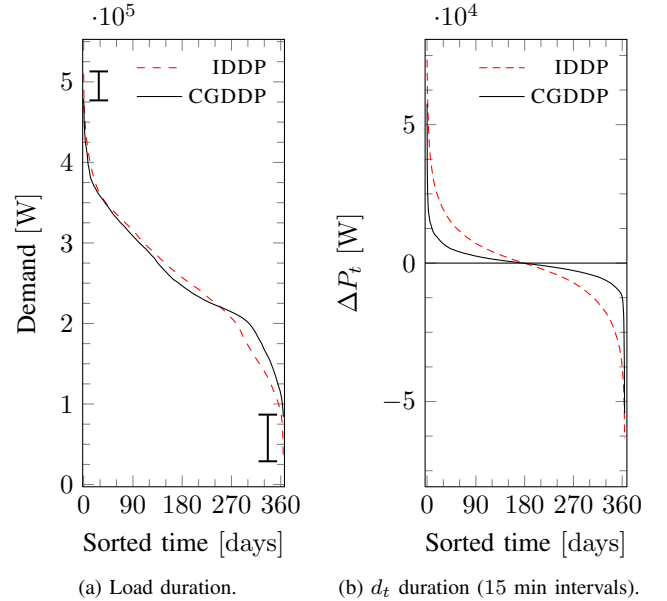


Fig. 3. Duration curves of results using IDDP and CGDDP.

and [7]. This case provides extensive demand-side flexibility. The objective of this case is to minimize peak demand and to flatten the demand profile. The case does not explicitly account for household comfort; instead, this is guarded by constraints (e.g. temperature limits and task deadlines). Forecast errors are not considered here: although these can have a significant effect on the results, the focus of this paper is on planning, not on operational control. The work in [7] already shows that control is essential for this scenario. Therefore, we use IDDP rather than ‘no control’ as the reference case.

All simulations were performed on a system with dual Intel Xeon E5645 processors (24 threads). The column generation related MIPs were solved by CPLEX 12.5, with parallel mode enabled, a MIP gap tolerance of 1% and a 20 second time-out. Both IDDP and CGDDP are truncated after 10 feedback iterations.

### B. Simulation Results

The results suggest that replacing IDDP with CGDDP leads to a demand peak reduction of about 10–15% and a significant decrease in variability, as well as faster convergence of the planning process. Since the results are similar for most of the days, we illustrate the results with the aggregated electricity demand on a typical winter day in Fig. 2a (on winter days, the effects are clearer visible). The corresponding minimum and maximum demand during the planning session at 18:00 on this day are shown in Fig. 2b.

The annual load duration curves (Fig. 3a) confirm this peak reduction, although less strongly (-7.3%). Periods of low demand are also better exploited (demand ‘valleys’ are filled), leading to an increased base load. The overall slope of the curve can be attributed to the demand and PV supply difference over the course of a year.

The variability of demand is much smaller for CGDDP than for IDDP; this is supported by Fig. 3b. In the presented results, the sum of absolute differences between demand values is

reduced by as much as 62.8%. The highest demand change peaks appear to be improved as well, but this does not occur consistently between simulations.

The year simulation applying IDDP consumed 3 hours and 29 minutes (real/wall time). The CGDDP simulation consumed 23 hours and 25 minutes. Clearly, the column generation master problem demands more processing power.

### C. Evaluation

The introduction of column generation improves the schedules computed by the planning process significantly in comparison with IDDP, both from the objective and subjective points of view. Intuitively, we expected the difference between IDDP and CGDDP to be larger than occurring in the case. Though the optimal solution remains unknown, we believe that most of the peak reduction potential for this scenario has now been exploited, therefore leaving little room for further improvement. Other scenarios may give different results, yet we believe that a positive contribution is almost always present.

However, the presented approach still does not provide the optimal solution of the basic MIP. For computational efficiency, the MIPs are not always solved to optimality, subject to the limits noted in Section IV-A. Also, we might not observe the full solution space, due to the use of the relaxation for the generation of shadow prices.

Looking at the DP modeling of devices in [2], not all local flexibility is exposed; for example, EVs can be charged only at a limited number of power set-points. This may become a problem when (highly) local problems are considered.

Restricting the planning to short periods may lead to situations where forcing the main objective (peak reduction) to be minimized leads to critical situations for secondary objectives, even though the real value of this period is far from critical for a longer time period (e.g. over one month).

Having a generic mathematical framework for the planning process has proven to be very valuable: it allows the problem to be approached without the distraction of implementation details. This abstraction makes it possible to reason about the use and extension of general techniques. Without this framework, column generation would never have been considered as a possible solution strategy.

Although they do not cause immediate problems, the long simulation times are not helpful during development. The corresponding performance penalty is counter-productive in achieving the goal of increasing the frequency of planning. Also, the use of a commercial solver will be expensive due to the licensing fees for non-academic use. Therefore, the performance degradation caused by the column generation approach, and in particular its dependence on a quite large MIP, should be addressed in upcoming work.

## V. CONCLUSION

Column generation with distributed dynamic programming (CGDDP) is a very promising alternative to iterative distributed dynamic programming (IDDP) for demand management in smart grids. We observe a further peak reduction of 7.3% over IDDP

in the FLEX STREET scenario. By applying column generation, demand variability is also significantly decreased.

However, these planning improvements come at the cost of increased computational costs; therefore, these costs need to be addressed in future work. The problem complexity can be reduced by further partitioning of the problem. The column generation equation may also be optimized in a way that is different from MIP, or be applied only after a good solution has already been found, for example by using IDDP.

The approach described in this paper suggests new smart grid control applications. It has become possible to both express and optimize more complex objective functions: the proposed approach naturally handles problems with multiple, possibly conflicting, objectives.

However, this paper does not consider forecast errors. Therefore, such errors need to be addressed in future work; otherwise, planning driven smart grids will remain an impractical, theoretical dream.

## ACKNOWLEDGMENT

Bodo Manthey's excellent Optimization Modeling course has helped a lot in understanding the subject matter. The authors would also like to thank Reviewer 1 for his useful comments.

## REFERENCES

- [1] 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.
- [2] 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.
- [3] F. N. Claessen, "Smart grid control: An analysis of control systems within a Dutch residential microgrid incorporating decentralised renewable energy resources," Master's thesis, Utrecht University, December 2011.
- [4] H. A. Toersche, S. Nykamp, A. Molderink, J. L. Hurink, and G. J. M. Smit, "Controlling smart grid adaptivity," in *Innovative Smart Grid Technologies (ISGT) Europe*, IEEE PES, Berlin, Germany, October 2012.
- [5] G. B. Dantzig, *Linear Programming and Extensions*. Princeton U.P., 1998.
- [6] M. G. C. Bosman, A. Molderink, V. Bakker, G. J. M. Smit, and J. L. Hurink, "Multilevel unit commitment in smart grids," in *Proceedings of the 1st International Conference on Operations Research and Enterprise Systems, Vilamoura, Portugal*. SciTePress, 2012, pp. 361–370.
- [7] 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 smart grid control systems using the Flex Street model," *Renewable Energy*, 2012, under review.
- [8] 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.
- [9] D. Hammerstrom, T. Oliver, R. Melton, and R. Ambrosio, "Standardization of a hierarchical transactive control system," *GridInt*, vol. 9, 2009.
- [10] M. Hommelberg, B. van der Velde, C. Warner, 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.
- [11] 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.
- [12] 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.
- [13] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Oper. research*, vol. 46, no. 3, pp. 316–329, 1998.