Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 665–674

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Asynchronous Decentralized Framework for Unit Commitment in Power Systems

Paritosh Ramanan[1,2], Murat Yildirim[2], Edmond Chow[1], and Nagi Gebraeel[2]

[1] School of Computational Science and Engineering, College of Computing
[2] School of Industrial and Systems Engineering, College of Engineering
Georgia Institute of Technology,
Atlanta, Georgia, USA
{paritoshpr,murat}@gatech.edu, echow@cc.gatech.edu, nagi@isye.gatech.edu

Abstract

Optimization of power networks is a rich research area that focuses mainly on efficiently generating and distributing the right amount of power to meet demand requirements across various geographically dispersed regions. The Unit Commitment (UC) problem is one of the critical problems in power network research that involves determining the amount of power that must be produced by each generator in the power network subject to numerous operational constraints. Growth of these networks coupled with increased interconnectivity and cybersecurity measures have created a need for applying decentralized optimization paradigms. In this paper, we develop a novel asynchronous decentralized optimization framework to solve the UC problem. We demonstrate that our asynchronous approach outperforms conventional synchronous approaches, thereby promising greater gains in computational efficiency.

Keywords: decentralized optimization, unit commitment, asynchronous methods

1 Introduction

Large scale power networks form the backbone of the global energy infrastructure. A surge in global power consumption now demands better energy distribution schemes as well as increased efficiency in the management of power network resources and assets. Among the most critical assets of any power network infrastructure are the generators which produce electricity and the buses which demand power. The buses and generators form a part of the power network topology where the generators collectively aim to meet the total demand imposed by all the buses while also adhering to network-wide requirements on the flow of electricity. Power network optimization problems aim at better utilization of the generators and form a significant part of power systems research. The Unit Commitment (UC) problem [1] is one of the key optimization problems that has received considerable attention over the past few decades. The goal of the

UC problem is to determine how much power each generator has to produce given the cost of production and the power demand to be met.

The UC problem is a critical component for daily production planning for power utility companies. Utility companies that command power networks typically solve the UC problem many times throughout the entire day to determine the production levels of different generators on an hourly basis. The power network is subject to demand uncertainties that require constant monitoring and adjustments to power generation. There may also be sudden and unpredictable outages caused by natural or man-made phenomena that may disrupt the highly sensitive, hourly production schedule of the generators network-wide. An efficient algorithm for solving the UC problem therefore needs to be highly robust to sudden changes in operating conditions. The UC problem involves binary decision variables to decide which generators must be turned on or off at various time epochs across a fixed planning horizon. The resulting optimization model of the UC problem therefore becomes a Mixed Integer Programming (MIP) problem that falls under the category of non-convex optimization problems.

A large-scale power network is topologically divided into a number of regions which may represent different power utility companies or subsidiaries. Traditionally, a centralized model has been employed to conduct unit commitment on power networks. However, the centralized model has several drawbacks. First, centralized methods are unable to isolate potentially sensitive commercial operations data. Second, the performance of a centralized model for unit commitment deteriorates with increase in network size leading to poor scalability thereby making it unsuitable for large-scale power networks. Third, an attack on a single node can compromise the entire power network.

Decentralized optimization methods have lately emerged as a means to tackle the different operational issues presented above. Owing to a loose coupling between regions, the global UC optimization problem can be decomposed into smaller subproblems, each corresponding to a particular region. By iteratively solving each region's subproblem locally and exchanging information with neighboring subproblems, we can completely decentralize the solution to the global UC problem. Since each subproblem is locally held by the region itself, decentralized methods retain privacy of commercial data pertaining to each region. Further, decentralized methods enable solving for the global optimum only on the basis of local infrastructural data and relevant operational data points of neighboring regions thereby improving scalability.

Existing approaches to decentralized unit commitment problems [2] adopt a synchronous approach wherein each iteration of the local subproblem and the subsequent information exchange is performed in tandem by all regions. Owing to a tight coupling between computation and communication, the information obtained from neighboring regions in the synchronous approach is always pertaining to the current iteration. Such synchronous models do not account for geographically distant computational nodes wherein transfer of data between nodes comes with its own communication delay. At different nodes, different subproblems and computing architectures with varying computational capabilities will lead to heterogenous processing times. This variability in the computation time would be further compounded by time-varying loading on the computing resources due to on-line control of many connected assets in the region. Therefore, methods that rely on synchronization of computational nodes do not account for delays incurred in practice. Consequently, in large-scale distributed systems involving many interconnections, achieving perfect synchrony is extremely difficult [3]. Furthermore, synchronous computational methods may simply fail to converge in the face of even a slight degree of asynchrony [4]. Even with a hypothetical synchronous computation system, a slow computational node or communication link can significantly increase the computational time by under-utilizing the computing resources. This causes all the computational nodes to wait until the problematic node completes the computation and data transfer. In addition, online instrumentation and digital control might make power systems vulnerable to cyber attacks and with the assumption of synchronization, any attack to stop or slow down the operations of a single local node can significantly impact global power system operations.

Asynchronous methods can be used to circumvent the aforementioned problems as well as to develop a more flexible and resilient computational platform. We stand to gain significant computational benefit by adopting an asynchronous approach wherein those regions with a smaller subproblem to solve may proceed onto their next iteration without waiting on the slower regions to finish. In doing so, each region executes its iterations independently of other regions and uses the latest available information from its neighbors for each of its iterations. In an asynchronous approach, the solution of the entire system is no longer held up by a slow computational node and a faster solution to the global UC problem can be expected.

Non-convex, MIP problems are challenging to solve in their own right. However, asynchronous nature adds another layer of complexity to such problems and their convergence behavior is still an open question in the optimization community. To the best of our knowledge, this paper is the first to attempt to outline an asynchronous decentralized MIP based framework to solve the UC problem. With a region based decomposition, our technique asynchronously solves for an optimal production schedule within each region while ensuring privacy as well as minimal sharing of commercial data between any two regions. We iteratively use the Alternating Direction Method of Multipliers (ADMM) to solve the UC problem locally and exchange information with neighbors, eventually converging to a solution close to that of the centralized problem. Our method can be orchestrated on cloud based infrastructure, institutional clusters or a distributed hybrid combination of both. We show that our method is capable of handling unbalanced computation among regions and computationally performs better than its synchronous counterpart without compromising on solution quality to a great extent.

Our paper is organized as follows. In Section 2 we discuss approaches in power systems that are pertinent to the UC problem. We then proceed towards the centralized and decentralized mathematical formulations of the UC problem in Section 3. We describe our novel asynchronous decentralized algorithm for solving the UC problem in Section 4. We detail the experimental setup used to validate our algorithm as well as discuss the results of our algorithm on the IEEE 118 bus case in Section 5. We present our concluding remarks in Section 6.

2 Related Work

In the field of power systems, owing to the complexity of MIP UC problems, there have been several attempts to seek a parallel solution of the UC problem. The work done in [5] attempts to solve the stochastic unit commitment problem for a fleet of sustainable energy sources wherein the imposed demand is time varying. The work done in [6] also uses a dual decomposition to solve a multi-area unit commitment problem in a synchronous fashion. The work done in [7] solves a unit commitment problem using a two stage approach via an incremental sub-gradient method to progress the dual variable while simultaneously recovering the primal feasibility.

Decentralized methods form a fast emerging sub category within distributed computing which are best suited for geographically distributed computing elements due to their fault-tolerant nature and high scalability [8]. In the context of decentralized approaches, augmented Lagrangian techniques like ADMM [10] have been popular for quite some time, owing to their fast convergence properties [11], but their applicability has mostly been limited to solving convex problems. However, recent works have focussed on application of ADMM to a fixed set of non-convex and non smooth distributed optimization problems under some strict conditions

[12]. The work done in paper [2] investigates large-scale decentralized unit commitment by solving the non-convex unit commitment problem subject to operational constraints by using dual decomposition and ADMM in a synchronous manner. The work done in the paper [9] demonstrates convergence of asynchronous ADMM in a distributed master-slave based computing model.

Our algorithm differs significantly from all approaches presented in this section. We primarily employ a region based topological decomposition of the power network and solve each region's sub problem in a decentralized and asynchronous fashion. Our problem is not a consensus optimization problem and owing to its asynchronous and non-convex nature is much more challenging to solve. In this paper we demonstrate that our algorithm provides a good solution quality in a decentralized and asynchronous setting. We now proceed to the problem formulation.

3 Problem Formulation

The UC problem is usually solved over a fixed planning horizon of size T, divided into discrete time epochs. We denote the set of all generators as $\mathcal{G}$ and the set of buses as $\mathcal{B}$. The UC optimization problem can be represented as follows:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y} \tag{1a}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{y} + \boldsymbol{B}\boldsymbol{x} = \boldsymbol{E} \tag{1b}$$

$$\boldsymbol{F}\boldsymbol{y} = \boldsymbol{H} \tag{1c}$$

where $\boldsymbol{x}$ is a vector of length $|\mathcal{G}|\times T$ whose components are binary values specifying if generators are turned on or turned off across each time epoch for each generator in the network. Similarly, $\boldsymbol{y}$ is a real-valued vector of length $(|\mathcal{B}|+|\mathcal{G}|)\times T$ whose components represent dispatch variables specifying the level of production on generators as well as the electric phase angles on separate buses.

The objective function (1a) in Problem (1) involves two terms $\boldsymbol{c}^\top\boldsymbol{x}$ and $\boldsymbol{d}^\top\boldsymbol{y}$ which represent the cost associated with turning generators on or off across the planning horizon, referred to as commitment, and the cost incurred in production of electricity respectively. The objective is to minimize this total cost across the entire network subject to constraints that either focus on every generator separately as in (1b), or link all the generator operations together as in (1c). The first class of constraints given by (1b) include limits on generator production capacities and embody flexibilities such as how fast production can change within an hour. The second set of constraints given by (1c) ensure that network demand is satisfied while ensuring that electricity flow is in line with transmission capacities. We refer the interested reader to [13] [2] for a more detailed formulation of the UC problem.

Given a region based partition, we can divide the UC problem stated in Problem (1) into smaller region based subproblems. In such a case the objective function of each subproblem only involves variables dealing with assets of a particular region. Similarly, each region also ends up holding its own set of constraints corresponding to (1b), (1c) involving assets specific to that region alone. Therefore, given a region based partition, the UC problem may be solved in an entirely decentralized manner where each processing element handles the UC sub-problem of a particular region. Every region iteratively tries to obtain a balance in terms of the dispatch variables with respect to the subproblem residing in its neighboring regions.

We define $\mathcal{G}_r$ as the set of generators in region r and $\mathcal{R}_r$ as the set of neighboring regions of r. Similarly, $\boldsymbol{x}_r, \boldsymbol{y}_r$ represent the binary commitment and dispatch variables of region r. In the decentralized case, ADMM is applied by relaxing constraint (1c). ADMM iteratively uses Lagrangian penalty terms $\boldsymbol{\lambda}$ and ρ to ensure that the final solution satisfies this constraint within acceptable accuracy. Therefore, ADMM ensures that flow of electricity is balanced between regions causing production of each region to adjust accordingly thereby satisfying demand per region indirectly for the entire network.

For each region r, we obtain $\hat{\boldsymbol{y}}_{r'}$ which is an estimate of $\boldsymbol{y}_{r'} \quad \forall r' \in \mathcal{R}_r$, and solve the following problem:

$$\min_{\boldsymbol{x}_r,\boldsymbol{y}_r} \quad \boldsymbol{c}_r^\top \boldsymbol{x}_r + \boldsymbol{d}_r^\top \boldsymbol{y}_r + \boldsymbol{\lambda}_r^\top (\boldsymbol{F}_r \boldsymbol{y}_r + \sum_{r' \in \mathcal{R}_r} \boldsymbol{F}_{r'} \hat{\boldsymbol{y}}_{r'} - \boldsymbol{H}_r) + \rho \left\| \boldsymbol{F}_r \boldsymbol{y}_r + \sum_{r' \in \mathcal{R}_r} \boldsymbol{F}_{r'} \hat{\boldsymbol{y}}_{r'} - \boldsymbol{H}_r \right\|_2^2 \tag{2a}$$

$$\text{subject to} \quad \boldsymbol{A}_r \boldsymbol{y}_r + \boldsymbol{B}_r \boldsymbol{x}_r = \boldsymbol{E}_r \tag{2b}$$

$$\tag{2c}$$

where $\boldsymbol{\lambda}_r$ is updated as follows.

$$\boldsymbol{\lambda}_r = \boldsymbol{\lambda}_r + \rho(\boldsymbol{F}_r \boldsymbol{y}_r + \sum_{r' \in \mathcal{R}_r} \boldsymbol{F}_{r'} \hat{\boldsymbol{y}}_{r'} - \boldsymbol{H}_r) \tag{3}$$

With the decentralized formulation of the UC problem presented above, we now move on to describe the asynchronous decentralized UC algorithm.

4 Algorithm Design

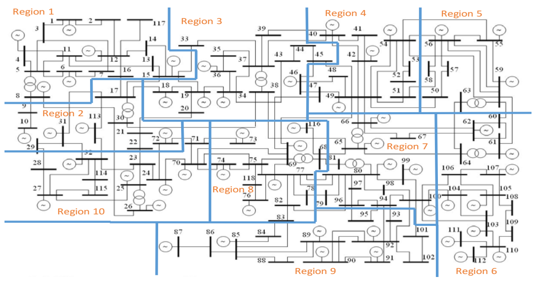
As stated in Section 1, solving the non-convex decentralized UC problem is not an easy task. However, it is much easier to solve a convex version of the same problem [2]. The optimization model given in Problem(2) can be relaxed into a convex case by removing the binary constraint for commitment variables in $\boldsymbol{x}_r$ and making them continuous in the interval $[0, 1]$. We can use the solution from the relaxed version as a good initial guess to solve the actual decentralized UC optimization model given by Problem (2) with the binary condition enforced. Therefore we solve the decentralized UC optimization model given by Problem (2) in two phases. It is important to note that both phases are solved in a purely asynchronous and decentralized manner.

Algorithm 1 represents the asynchronous decentralized algorithm for solving the UC problem which is solved in convex and non-convex phases. In the former phase, we relax the binary commitment vector $\boldsymbol{x}$ such that all its elements lie in the continous interval $[0, 1]$ making Problem (2) convex whereas in the latter phase, we impose the constraint such that $\boldsymbol{x} \in \{0, 1\}$ thereby solving its non-convex version. Each node then receives the dispatch estimate $\hat{\boldsymbol{y}}_{r'}$ from each of its neighboring region r'. Local convergence in either case is determined if the norm of the primal and dual dispatch variables falls below a fixed threshold denoted by α and β respectively. Global convergence occurs when every region attains local convergence. We designate a master node to determine global convergence using local convergence values of worker nodes asynchronously. The algorithm terminates upon attaining global convergence.

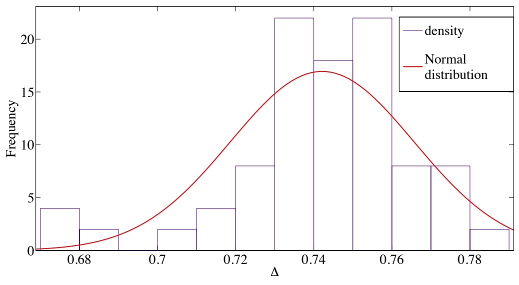
Algorithm 1 Asynchronous Decentralized UC Algorithm run on each processor

```
Initialize $\boldsymbol{x}_r, \boldsymbol{y}_r$
$k \leftarrow 0$
for phase = convex,non-convex do
    while global convergence is not acheived do
        send $\boldsymbol{y}_r$ to regions $\forall r' \in \mathcal{R}_r$
        in convex phase, set $\boldsymbol{x}$ such that $0 \leq x_i \leq 1, \forall x_i \in \boldsymbol{x}$
        in non-convex phase, set $\boldsymbol{x}$ such that $x_i \in \{0,1\}, \forall x_i \in \boldsymbol{x}$
        solve Problem (2)
        compute $\bar{\boldsymbol{y}}_r^k = \sum_{r' \in \mathcal{R}_r} \boldsymbol{F}_{r'} \hat{\boldsymbol{y}}_{r'}$
        update $\boldsymbol{\lambda}$ according to (3)
        if $(||\boldsymbol{y}_r - \bar{\boldsymbol{y}}_r^k|| < \alpha)$ and $(||\bar{\boldsymbol{y}}_r^k - \bar{\boldsymbol{y}}_r^{k-1}|| < \beta)$ then
            local_convergence $\leftarrow 1$
        else
            local_convergence $\leftarrow 0$
        end if
        $k \leftarrow k + 1$
    end while
end for
```

5 Experimental Setup and Results



(a) 10 region partition of IEEE 1118 bus case



(b) Distribution of Δ, with $\mu = 0.74, \sigma = 0.0235$

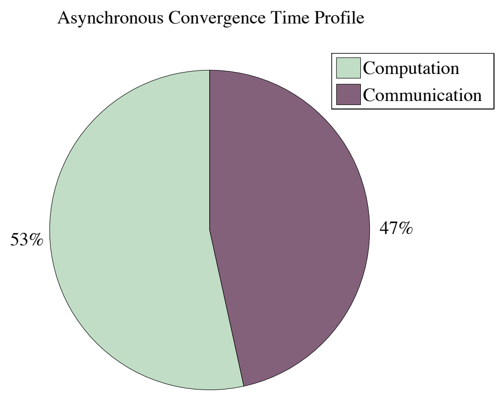
Figure 1: IEEE 118 bus case[1]

We simulate a large power network topologically divided into 10 regions based on the IEEE 118 bus case involving 54 generators derived from *MATPOWER* [14]. We solve the unit commitment problem over a planning horizon comprising of 24 time epochs spanning an entire day. The objective behind this simulation is to determine convergence behavior of Algorithm 1 owing to its non-convex nature. We simulate a geographically dispersed set of nodes on a high performance cluster comprising of Intel Xeon CPUs with a clock rate of 2.80GHz with each core representing one region. We used Remote Memory Access (RMA) primitives of MPI with passive target synchronization to facilitate asynchronous message passing. The *mpi4py* [15] package was used as an interface to the MPI library for our experiments. *Gurobi 6.5* [16] was used for performing the optimization with respect to Problem (2) on each node. We evaluate Algorithm 1 with respect to its synchronous counterpart obtained by inserting an MPI *Barrier*

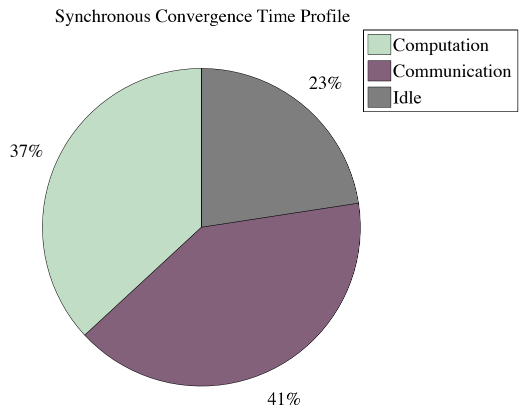
[1]http://energy.komisc.ru/dev/test˙cases

call at the beginning of each iteration of Algorithm 1.

An asynchronous algorithm is stochastic in nature, therefore we performed a total of 50 runs of Algorithm 1 with the same run time parameters and recorded the communication and the computation time for each of the runs with $\alpha = 5$ and $\beta = 10$. For the synchronous version, apart from the communication and computation time, we also record the idle time each processor spends owing to the *Barrier* call. In order to establish a high degree of asynchrony, we add a small delay of 0.2 secs for half of the regions in both cases. We measure the degree of asynchrony, Δ, on the basis of the ratio between the least number of iterations vs the maximum number of iterations performed by any node in the asynchronous case. Figure 1(b) shows the distribution of Δ measured over all the 50 runs. A mean value of 0.74 implies that for every 100 iterations performed by the fastest region in the system, the slowest region performs on an average close to 74 iterations. Therefore, Figure 1(b) demonstrates that the 10 region partition of the 118 bus case has a high degree of asynchrony.



(a)

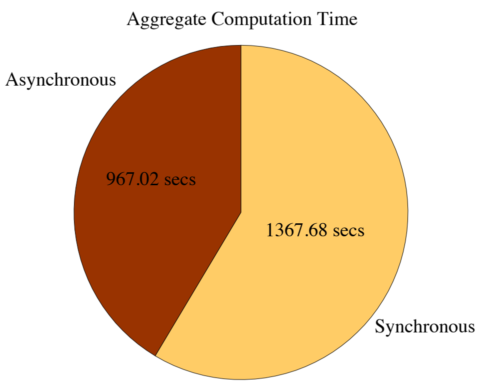


(b)

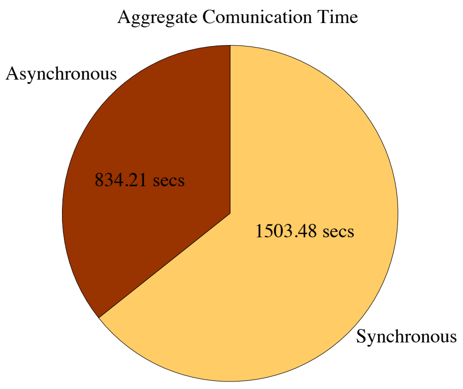
Figure 2: Analysis of Convergence Time profile for Asynchronous and Synchronous Methods

Figure 2(a) pictorially depicts data pertaining to the fraction of time spent by the processors performing communication and computation tasks in the asynchronous case. Figure 2(b) shows the fraction of time spent idle by the processors in addition to the fraction of time spent in communication and computation tasks in the synchronous case. From Figures 2(a), 2(b) we can infer that the asynchronous version of Algorithm 1 spends a substantial part of its time indulging in actual computation tasks as compared to its synchronous counterpart. Moreover, it can also be noted from Figure 2(b) that in the synchronous algorithm nodes spend a significant amount of time in an idle state.

We calculate the average communication and computation time for each of the nodes and obtain an aggregate of total CPU time spent in each task by summing over the respective average values for all the nodes. We present the aggregate values of CPU time separately for the computation as well as communication related tasks for both synchronous and asynchronous versions of Algorithm 1 in Figure 3. From Figure 3(a) and Figure 3(b) we can see that the total aggregate CPU time spent in both computation and communication is in fact lesser in the asynchronous case than that of the synchronous version. Therefore, from Figure 3 we can infer that Algorithm 1 promises faster convergence while tolerating a good degree of asynchrony while a synchronous version incurs a significant amount of idle time. From Figures 2, 3 we also



(a)



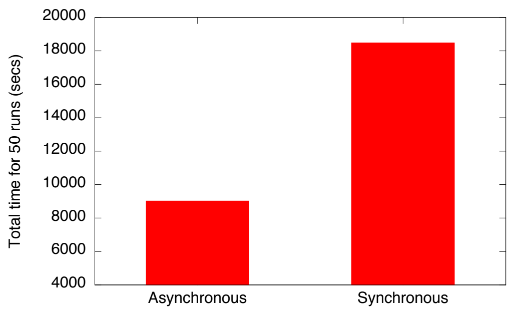
(b)

Figure 3: Analysis of aggregate total CPU time spent in Asynchronous and Synchronous methods

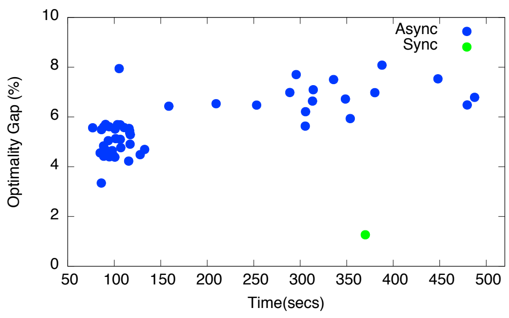
Synchronous		Asynchronous	
Mean (μ)	Std. Dev. (σ)	Mean (μ)	Std. Dev. (σ)
369.57	0.72	180.40	122.02

Table 1: Convergence time analysis of Algorithm 1

infer that even though the asynchronous method spends lesser time performing computation related tasks, it is computationally more efficient than the synchronous case as the fraction of time spent in computation is higher for the asynchronous case.



(a) Cumulative Behavior



(b) Optimality Gap

Figure 4: Performance Analysis of Algorithm 1

Table 1 shows the mean μ and standard deviation (σ) of 50 runs of both the synchronous and the asynchronous versions of Algorithm 1 with the same parameters. From Table 1 we can see that the synchronous case exhibits very little variation in convergence time and is relatively consistent across all runs whereas the asynchronous case shows considerable variation in run time. However, we can also infer that on an average, the asynchronous method is almost twice as fast as the synchronous method and convergence time values within 1.5σ of the mean in the asynchronous case are still lower than the synchronous case.

In modern large-scale power systems, the UC problem is solved in a high frequency manner throughout the entire day. In order to demonstrate the computational gains in a high frequency environment, we apply the asynchronous method to solve the UC problem over 50 consecutive runs with the same runtime parameters. Figure 4(a), a bar graph demonstrates the cumulative time taken by our algorithm with respect to its synchronous counterpart over all the 50 runs thereby implying the tremendous computational advantage presented by the asynchronous algorithm in a high frequency setting.

To measure the quality of solution of the optimization model, we use a quantity known as the *optimality gap*. It is the relative difference in objective values with respect to the optimal solution of the problem which is provided by the centralized method in our case. Therefore, we used the objective value yielded by the centralized solution to calculate our optimality gap.

Figure 4(b) shows a scatter plot of the optimality gap yielded by Algorithm 1 versus total convergence time over all the 50 runs in blue while the synchronous result is shown in green. We can see that the optimality gap for the asynchronous algorithm is higher than that of the synchronous although the asynchronous version mostly takes much lesser time to arrive at a tolerable optimality gap. While an optimality gap of zero is an interesting theoretical problem, in practice it is more preferable to obtain a small optimality gap in a much faster way[17]. Further, we can also infer from the figure that the optimality gap has an upper bound at approximately 8%. Since our problem is non-convex and cannot be reduced to a consensus optimization problem in its relaxed state, we see a variation in solution quality and a strong correlation between solution time and quality can be inferred. This variation is due to bad solution paths taken by the algorithm which leads to a deteriorating optimality gap.

6 Conclusion

In this paper, we present an asynchronous decentralized algorithm for solving the MIP UC problem in power networks. By decomposing the power network into regions with smaller subproblems, we can decentralize the solution to the UC problem wherein each region solves its locally held UC problem with respect to its network constraints and on the basis of information received from its neighbor regions. In order to handle the complexity stemming from the non-convex nature of our problem and make our solution more robust, we employ a two-stage asynchronous solution where the first phase solves a relaxed version of the UC problem followed by the non-convex MIP version of the same problem. We use a 10 region partition of the IEEE 118 bus problem to demonstrate that our asynchronous decentralized algorithm outperforms its synchronous counterparts in terms of the computation, communication and idle time as well as with respect to the convergence time and the solution quality as assured by the optimality gap. We show that our algorithm is specifically suited to high frequency applications of the UC problem owing to a significantly less time incurred in its overall execution as opposed to the traditional synchronous approach.

From the analysis of the results we can draw a number of conclusions. First, solving non-convex problems asynchronously using ADMM may lead to bad solution paths in some cases which affects the solution quality. Second, there exists a very strong correlation between solution time and quality in such cases. Third, despite the non-convex nature of the problem asynchronous methods are capable of providing significant computational benefit while yielding a solution that is in close approximation to that of synchronous methods.

Acknowledgement. This material is based upon work supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Award Number DE-SC-0016564.

References

[1] N. P. Padhy, "Unit commitment-a bibliographical survey," *IEEE Transactions on Power Systems*, vol. 19, pp. 1196–1205, May 2004.

[2] M. J. Feizollahi, M. Costley, S. Ahmed, and S. Grijalva, "Large-scale decentralized unit commitment," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 97 – 106, 2015.

[3] O. Babaoglu, R. Davoli, L. A. Giachini, and M. G. Baker, "Relacs: A communications infrastructure for constructing reliable applications in large-scale distributed systems," in *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*, vol. 2, pp. 612–621, Jan 1995.

[4] T. Wu, K. Yuan, Q. Ling, W. Yin, and A. H. Sayed, "Decentralized consensus optimization with asynchrony and delays," *arXiv preprint arXiv:1612.00150*, 2016.

[5] A. Papavasiliou, S. S. Oren, and B. Rountree, "Applying high performance computing to transmission-constrained stochastic unit commitment for renewable energy integration," *IEEE Transactions on Power Systems*, vol. 30, pp. 1109–1120, May 2015.

[6] P. Srikantha and D. Kundur, "Distributed optimization of dispatch in sustainable generation systems via dual decomposition," *IEEE Transactions on Smart Grid*, vol. 6, pp. 2501–2509, Sept 2015.

[7] I. Aravena and A. Papavasiliou, "A distributed asynchronous algorithm for the two-stage stochastic unit commitment problem," in *2015 IEEE Power Energy Society General Meeting*, pp. 1–5, July 2015.

[8] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[9] R. Zhang and J. T. Kwok, "Asynchronous distributed admm for consensus optimization," in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, vol. 32 of *ICML'14*, pp. 1701–1709, 2014.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, pp. 1–122, Jan. 2011.

[11] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, pp. 1750–1761, April 2014.

[12] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *arXiv preprint arXiv:1511.06324*, 2015.

[13] M. Yildirim, X. A. Sun, and N. Z. Gebraeel, "Sensor-driven condition-based generator maintenance scheduling part ii: Incorporating operations," *IEEE Transactions on Power Systems*, vol. 31, pp. 4263–4271, Nov 2016.

[14] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, pp. 12–19, Feb 2011.

[15] L. Dalcin, R. Paz, M. Storti, and J. D., "Mpi for python: Performance improvements and mpi-2 extensions," *Journal of Parallel and Distributed Computing*, vol. 68, no. 5, pp. 655 – 662, 2008.

[16] G. O. Inc., "Gurobi optimizer reference manual," 2015.

[17] K. Hedman, M. Ferris, R. O'Neill, E. Fisher, and S. Oren, "Co-optimization of generation unit commitment and transmission switching with n-1 reliability," in *IEEE PES General Meeting*, pp. 1–1, July 2010.