Asynchronous Implementation of a Region-Based Distributed Optimal Power Flow Algorithm

Alyssia Dong, Roman Le Goff Latimier, Hamid Ben Ahmed

▶ **To cite this version:**

Alyssia Dong, Roman Le Goff Latimier, Hamid Ben Ahmed. Asynchronous Implementation of a Region-Based Distributed Optimal Power Flow Algorithm. 2022 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Oct 2022, Novi Sad, Serbia. 10.1109/ISGT-Europe54678.2022.9960399 . hal-04286137

HAL Id: hal-04286137
https://hal.science/hal-04286137v1

Submitted on 15 Nov 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Asynchronous Implementation of a Region-Based Distributed Optimal Power Flow Algorithm

Alyssia Dong, Roman Le Goff Latimier, Hamid Ben Ahmed
SATIE, CNRS, *ENS Rennes*
Bruz, France
{alyssia.dong, roman.legoff-latimier, benahmed}@ens-rennes.fr

***Abstract*—The scalability potential of distributed algorithms justifies the growing interest for decentralized optimal power flow (OPF) algorithms. However, the need for communication can significantly slow down the convergence of these algorithms when considering various communication hazards. To overcome this issue, an asynchronous framework is proposed where decentralized agents only wait for a part of the overall messages. Simulations considering communication and computation delays will highlight the speedup and robustness of the asynchronous version of the algorithm in the case of message losses.**

***Index Terms*—ADMM, Asynchronous, Communication, Decentralization, Optimal Power Flow**

I. Introduction

The ever growing integration of renewable energy into the electricity markets requires new ways to manage the power network, like solving a constrained optimization problem in a decentralized manner. Decentralization requires computing agents, representing a single node or a multitude of nodes grouped by regions, to solve their own local sub-problem and to communicate with their physical neighbors. This communication requirement may be the source of significant slowdowns of the algorithm convergence. Whilst such a decentralized algorithm is mathematically scalable, it is hardly applicable considering the messages delays and losses that occur in a real implementation. Even when using a reliable protocol like TCP, in which another message is sent again if the previous one got lost, the induced idle time of a region can significantly increase and slow down the entire resolution. By only waiting for a fraction of the information instead of all of it, the idle time can be cut down between each iteration. This is referred to as an asynchronous implementation. We already studied in [1] an asynchronous peer-to-peer electricity market algorithm and showed that it empirically sped up the convergence time by up to 40%.

The subject of asynchronous implementation of a decentralized optimal power flow problem has been discussed in several articles the past few years. The DC-OPF in [2] is solved in a distributed manner using lagrangian multipliers. The updates are asynchronous in the sense that the buses in the same area exchange information after each iteration, while communication between the areas occurs only after multiple iterations. This allows for a reduction of communication between regions. However, the computation and the communication delays are widely not considered. Another asynchronous DC-OPF [3] is solved using a lagrangian relaxation method, in which not all messages are waited for between each iteration. This allows to cut down idle time between iterations, and results in an up to 50% convergence speedup. An asynchronous, alternating directions method of multipliers (ADMM) based, AC-OPF distributed by regions is solved in [4]. The communication delays are randomly drawn within a range and the computation delays are fixed. They showed that under mild communication delays, the convergence speed is comparable to faster than in the synchronous case. However, there are no considerations over the different size partitioning of the testcase and the consequence over the computing delays. A multi-timestep, ADMM based, economic dispatch optimization is studied in [5]. Each local agent communicates only with their direct neighbors, skipping messages to speed up the convergence. The asynchronous results deviate from the optimal point if the rate of communication delay is high. The combination of an electrical and a heating system is studied in [6] using a relaxed ADMM method to distribute the optimization between an electrical power system and several district heating systems. They highlight that the relaxed ADMM allows convergence towards the optimal solution in presence of message losses as opposed to the classic ADMM results which are not optimal. Also, [7] studies a similar electrical and heat system for a combined day-ahead and real-time energy management optimization. They use event-triggered based communication in order to reduce the number of exchanged messages. In the presented articles, the asynchronous updates allow either a reduction of the number of exchanges between computing agents or a speedup of the convergence time. Considering a decentralized algorithm split between R regions, the computation delays depend on the size of each region and subsequently, the number of regions R. In an asynchronous resolution, both the computation and the communication delays play a role in the final convergence time, as shown in [8]. The previous articles did not address the influence of partitioning regarding the convergence time of the resolution.

In this article, we will study the asynchronous implementation of the decentralized optimal power flow algorithm presented in [9]. We will highlight that such algorithm results in the optimal solution and is more robust to communication delay variations coming from message loss or overall communication slowdowns. Both communication delays and

978-1-6654-8032-1/22/$31.00 ©2022 IEEE

computations delays will be taken into account in our studies.

After summarizing the decentralized OPF formulation of [9] in II, we propose in III an asynchronous implementation of the algorithm. The simulation platform of our study is introduced in IV as well as the communication and computation delay models. Finally, simulation results are presented in V before concluding on the study in VI.

II. Existing Decentralized Optimal Power Flow

The optimal power flow problem (1) finds the optimal injection plan for all generators in the electrical network so as to minimize their costs while also respecting the physical constraints of the network. Those constraints include power flow limitation in the lines, voltage amplitude and phase limitations and power limitations at each node.

$$\min \sum_{i \in \mathcal{N}} f_i(P_i) \tag{1a}$$

$$\text{w.r.t. } V_i \in \mathbb{C},\ P_i \in \mathbb{R},\ Q_i \in \mathbb{R},\ i \in \mathcal{N}$$

$$\text{s.t. } \underline{P_i} \leq P_i \leq \overline{P_i} \tag{1b}$$

$$\underline{Q_i} \leq Q_i \leq \overline{Q_i} \tag{1c}$$

$$\underline{V_i} \leq |V_i| \leq \overline{V_i} \tag{1d}$$

$$P_i + jQ_i = V_i \sum_{j \in \mathcal{N}_i} Y_{i,j}^* V_j^* \tag{1e}$$

$\mathcal{N}$ represents the set of all electrical nodes. For a node $i \in \mathcal{N}$, let us define $\mathcal{N}_i$ as the set of neighbor nodes, f_i the cost function, P_i and Q_i the active and reactive power respectively and V_i the complex voltage. $Y_{i,j}$ designates the complex admittance of the line connecting nodes i and j.

The resolution of such a problem is tenuous given the non-linearity of the constraints. Also, the bigger the problem is, i.e. the higher the number of buses and lines, the slower it gets while also demanding higher computational resources. To overcome this scalability issue, Erseghe proposed in [9] a decomposition of the OPF problem into regions using the ADMM decomposition method. Each region k is composed of a given set of nodes $\mathcal{R}_k$. We selected this algorithm for its robustness and the fact that it does not require any form of coordination. In this section, we only summarize the main elements of the algorithm. The original article [9] should be referred to for an exhaustive explanation.

At every iteration given in Alg. 1, each region k solves a local OPF problem and communicates with all its neighbor regions the updated value of its boundary nodes voltages $j \in \mathcal{O}_k$, where $\mathcal{O}_k$ is the set of region k boundary nodes. In turn, the region waits for all messages from those regions for their updated boundary values. Once all messages have been received by the region, the calculations can continue towards the next iteration. As soon as the algorithm has reached convergence, a consensus on boundary nodes voltages has been found between the neighbor regions and the overall solution is equivalent to the optimal solution of (1).

It is worth noting that the convergence speed of this algorithm strongly depends on the coefficients $a_{j,k,h}$ and $d_{j,k}$ that can be found in (2) and (3). As explained in more details in [9], those coefficients are calculated from tuning coefficients w_k specific to each region. Hence, it is important to find the right coefficient set in order to speed up the synchronous convergence as much as possible.

Algorithm 1 Region k parallel processing synchronous algorithm, from [9]

while Convergence not reached **do**
  if $t = 0$ **then**
    Initialize local voltages $\mathbf{v}_k = [v_{k,j}]_{j \in \mathcal{V}_k}$
  else
    Update local voltages via

$$\mathbf{v}_k \leftarrow \arg\min_{\mathbf{v} \in \mathcal{Q}_k} F_k(\mathbf{v}) + \sum_{j \in \mathcal{O}_k} d_{j,k} |v_j - \beta_{k,j}|^2 \tag{2}$$

  end if
  Send boundary values $v_{k,j}$, $j \in \mathcal{O}_k$ to neighbor regions
  while the boundary values $v_{h,j}$, $j \in \mathcal{O}_k \cap \mathcal{O}_h$ have not yet been received from neighbor regions h **do**
    Wait
  end while
  Evaluate the mixing values for $j \in \mathcal{O}_k$

$$u_{k,j} \leftarrow \sum_{h \in \mathcal{M}_j} \frac{1}{2} \tilde{a}_{j,k,h} (v_{k,j} - v_{h,j}) \tag{3}$$

  if $t = 0$ **then**
    Reset memory $m_{k,j} \leftarrow 0$, $j \in \mathcal{O}_k$
  else
    Update memory $m_{k,j} \leftarrow m_{k,j} + u_{k,j}$, $j \in \mathcal{O}_k$ (4)
  end if
  $\beta_{k,j} \leftarrow v_{k,j} - u_{k,j} - m_{k,j}$, $j \in \mathcal{O}_k$ (5)
  $t \leftarrow t + 1$
end while

III. Asynchronous algorithm

In the synchronous implementation that we just presented, the region must wait for all messages at each iteration before continuing its computations. This can be the source of significant slowdowns if one of their neighbors has a long computation time or if the communication network is saturated and loses the message. In this case, the majority of the idle time between iterations is caused by one region which takes a longer time to answer back. Moreover, this delay will spread to other regions and worsen with message losses. To overcome this delay issue, several studies have been done on asynchronous implementations of such distributed algorithms [1], [3], [4], [8]. In this framework, instead of waiting for all messages to arrive to one region before going on with the computations, the computation is now triggered by the arrival of specific messages. The data that has not been received yet is replaced by the data from the previous corresponding received message.

Once the local region k receives all messages concerning a certain node $j \in \mathcal{O}_k$ from neighbor regions, then this node is ready for the iteration computation. We also say that node

j is *activated*. Let us define $\mathcal{A}_k^t$ the set of activated nodes at iteration t and region k. The trigger for next iteration $t+1$ is not the number of received message, but the number A_k^t of boundary nodes ready for computation, i.e. the cardinality of $\mathcal{A}_k^t$. The asynchronism parameter δ is defined as the ratio of the minimum number of boundary nodes taken into account at every iteration over the total number of boundary nodes of the region:

$$A_k^t \geq \lceil \delta \cdot |\mathcal{O}_k| \rceil, \ \forall t, \forall k \tag{6}$$

with $\lceil \cdot \rceil$ the ceiling function, guaranteeing $A_k^t \geq 1$. This asynchronism parameter is the same throughout all regions. When $\delta = 100\%$, the region has to wait for all boundary nodes to be activated from its neighbors, it is thus equivalent to the synchronous version of the algorithm. To prevent the algorithm to self-obstruct, i.e. two regions are stuck waiting for the other one's messages and freeze the entire resolution, a timeout is implemented. The variables updates in (2)-(5) are only conducted for activated nodes $j \in \mathcal{A}_k^t$. Likewise, boundary nodes values are only communicated for activated nodes. The optimality of this new asynchronous algorithm is verified on a testcase in section V-A.

IV. Simulation platform

We apply the asynchronous algorithm to the IEEE 118 bus test case [10] illustrated in Fig. 1. The computation time as well as the communication time are simulated. The simulation runs on the Julia programming language, using SimJulia for the discrete event simulation environment, and PowerModels to solve the local problems. The simulations are done on an AMD Ryzen 9 3950X chipset @3.5GHz. Subsequently, the study will also be performed over multiple network partitions in order to observe the impact of region size over the convergence time of the algorithm. The parameters for every partition are given in Table I.

A. Computation delays

We assume an affine relationship between the computation time and the size of the region, which is equal to the number of local variables. The coefficients of the affine relationship were estimated by running the resolution of the IEEE testcases included in the matpower dataset, with number of variables ranging from 3 to 1200, and by performing a linear regression over the computation time.

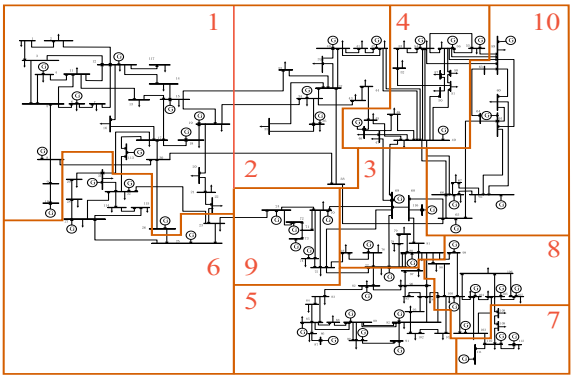

Fig. 1. IEEE 118 test case 10 regions partition.

TABLE I
Regions and weights of the 118 bus test case different partitions. The 3, 5 and 8 regions partitions are derived from the 10 regions one.

#Partitions	Weights w_k	Region partitioning
10	0.221, 0.362, 0.401, 0.157, 0.708, 0.594, 0.767, 0.048, 0.112, 0.538	$\mathcal{R}_1$: {1-22,26,30,113,117} ; $\mathcal{R}_3$:{68,69,76-81,116,118 } ; $\mathcal{R}_2$:{33-42} ; $\mathcal{R}_4$:{45-58} ; $\mathcal{R}_{10}$:{59-67} ; $\mathcal{R}_5$:{82-97,101,102} ; $\mathcal{R}_7$:{108-112} ; $\mathcal{R}_6$:{23,25,27-29,31,32,114,115} ; $\mathcal{R}_8$:{98-100,103-107} ; $\mathcal{R}_9$:{24,70-75}
8	0.222, 0.287, 0.415, 0.433, 0.600, 0.600, 0.388, 0.134	$\mathcal{R}_1$; $\mathcal{R}_2$; $\mathcal{R}_3 \cup \mathcal{R}_9$; $\mathcal{R}_4 \cup \mathcal{R}_{10}$; $\mathcal{R}_5$; $\mathcal{R}_6$; $\mathcal{R}_7$; $\mathcal{R}_8$
5	0.139, 0.938, 0.849,0.400, 0.526	$\mathcal{R}_1 \cup \mathcal{R}_6$; $\mathcal{R}_2$; $\mathcal{R}_3 \cup \mathcal{R}_3$; $\mathcal{R}_4 \cup \mathcal{R}_{10}$; $\mathcal{R}_5 \cup \mathcal{R}_7 \cup \mathcal{R}_8$
3	0.184, 0.345, 0.773	$\mathcal{R}_1 \cup \mathcal{R}_5 \cup \mathcal{R}_6 \cup \mathcal{R}_7 \cup \mathcal{R}_8$; $\mathcal{R}_2$; $\mathcal{R}_3 \cup \mathcal{R}_4 \cup \mathcal{R}_9 \cup \mathcal{R}_{10}$

$$\Delta_{comp}(k) = 0.25 \cdot Nvar(k) + 8 \quad \text{(ms)} \tag{7}$$

with $\Delta_{comp}(k)$ the computation time in milliseconds for region k, and $Nvar(k)$ the number of local variables. For the purpose of our study, the computation delays are only determined by the number of local variables of each region.

B. Communication delays

The communication delays are simulated each time a message is sent from a region to another. Considering a reliable communication protocol like TCP-IP, any lost message is detected and sent again after a certain timeout. We consider constant communication delays throughout the whole study if not specified otherwise: $\Delta_{comm} = 50$ ms, as we believe the geographical variations induce negligible delay variations compared to the ones induced by message losses. A message loss rate is added to this constant communication delay: each message sent by a region has a probability p_{loss} to be discarded by a router on the way. Every time it occurs to one message, an extra $2 \cdot \Delta_{comm}$ is added to its total end-to-end delay.

V. Simulation results

A. Asynchronous solution and definition of convergence time

Firstly, let us empirically verify that the final solution of the asynchronous algorithm corresponds to the optimal power flow solution. The asynchronous decentralized algorithm is applied to the 10 regions partition testcase, and the corresponding results are presented in Fig. 2. The active power state during the resolution is plotted in Fig. 2a over an arbitrary 120 seconds duration, for an asynchronous parameter value set to $\delta = 10\%$. This highlights the convergence of the asynchronous algorithm. The comparison between the optimal result and the corresponding asynchronous result is shown in Fig. 2b.

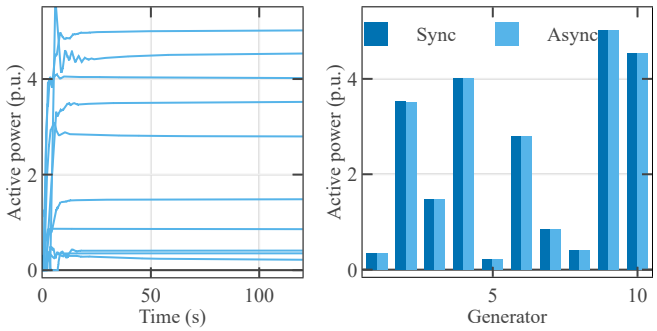

Fig. 2. Generators active power representation for the asynchronous OPF implementation, for an asynchronous parameter $\delta = 10\%$. Figure (a) represents the variable values during the calculations of the asynchronous decentralized OPF and figure (b) shows the comparison between the final result with the optimal synchronous result.

We observe that the asynchronous result is very close to the optimal one. A comparison value is defined in (8) in order to quantify the error between the asynchronous solution and the optimal solution at any given time t. The optimal solution, given by $(P_i^*, Q_i^*)_{i\in\mathcal{N}}$, is computed using the PowerModels Julia package.

$$c(t) = \sum_{i\in\mathcal{N}} \|(P_i(t) + jQ_i(t)) - (P_i^* + jQ_i^*)\|^2 \qquad (8)$$

The comparison value at the final time of Fig. 2 is equal to $7.4 \cdot 10^{-5}$ p.u. We define the convergence time of the asynchronous algorithm as the time t_{conv} when the comparison value $c(t)$ falls below a certain residual ϵ : $c(t_{conv}) \leq \epsilon$. For the remainder of this article, the residual value is set to $\epsilon = 10^{-3}$ p.u.

B. Influence of the asynchronism parameter over the convergence time

The influence of the asynchronous algorithm on the overall convergence time is shown in Fig. 3 for a 10 region testcase with no message loss, i.e. $p_{loss} = 0\%$. Each region alternates through periods of computation followed by idle periods. Fig. 3 represents the computation delays in green and the idle delays in yellow, averaged over all regions of the testcase.

Firstly, we note that the computation time increases as the asynchronous parameter δ decreases, which implies that a greater number of computations are needed in the asynchronous version to reach the same level of convergence than the synchronous algorithm. Also, at the same time, the idle period decreases as δ decreases, which is natural given that the idle periods depend on the number of boundary nodes values received by the region at each iteration. However, the idle delay decrease is not sufficient to counter the increase of computation time needed to reach convergence. This leads to a general slowdown of the asynchronous algorithm compared to the synchronous algorithm when considering a lossless communication network, where all communication delays are constant and equal.

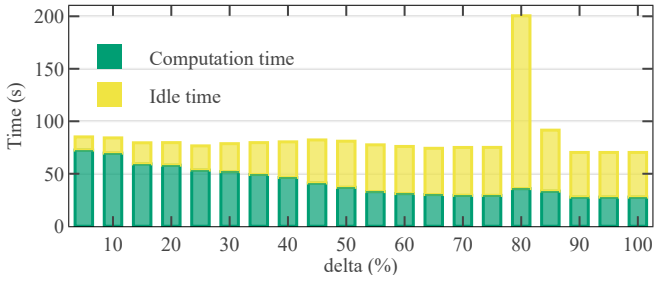

Fig. 3. Convergence time with respect to the asynchronous parameter δ, with no message loss $p_{loss} = 0\%$. The computation delays and the idle delays are averaged over the testcase regions.

We note that when $\delta = 80\%$ and $\delta = 85\%$, the idle delays are notably larger than for any other δ value. This is due to a self obstructing phenomenon which occurs when two regions wait for each other's message in order to resume their calculations, which then increase the total idle time.

C. Influence of message losses over the convergence time

In real conditions, the probability for a message to get lost in the communication network is not null. Fig. 4 studies the influence of the message loss probability p_{loss} on the convergence time of the algorithm, for the synchronous case $\delta = 100\%$ and the asynchronous case $\delta = 10\%$. We observe for the synchronous version $\delta = 100\%$ that the computation time is not modified by the message loss rate. The idle time is the only one that is impacted by the message losses, as it significantly increases with the message loss probability p_{loss}. The asynchronous algorithm is robust against message losses in that its idle time does not increase as much with p_{loss} as in the synchronous algorithm. This allows the asynchronous algorithm to become faster than its synchronous counterpart from $p_{loss} = 2\%$ and up, justifying the need for an asynchronous framework in real life conditions.

D. Effects of regions size

The partitioning of the problem into regions affects the decentralized resolution, even considering the original synchronous algorithm. As briefly explained in section II, the convergence speed of the original OPF algorithm heavily depends on the regions tuning coefficient set $(w_k)_k$, presented

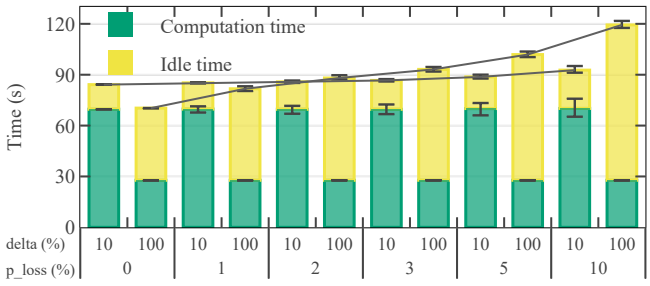

Fig. 4. Average computation and idle time with respect to the message loss probability p_{loss} for $\delta = 10\%$ (left) and $\delta = 100\%$ (right). The error bars represent the standard deviation over 20 simulations.

TABLE II
AVERAGE CONVERGENCE TIME FOR THE SYNCHRONOUS ALGORITHM UNDER A 5% MESSAGE LOSS RATE AND MAXIMUM SPEEDUP OF THE ASYNCHRONOUS ALGORITHM.

Number of regions	3	5	8	10
Synchronous convergence time (s)	35	47	138	103
Max asynchronous speedup (%)	8.27	12.3	12.4	18.8
Corresponding delta (%)	10	30	20	60

in Table I. Those coefficients have been manually tuned to minimize as much as possible the convergence time of each testcase partition. Table II shows the average convergence time of the synchronous algorithm for various partitions of the 118 bus testcase for a message loss rate of 5% and a fixed communication delay of 50 ms per message. Even considering that a bigger region has longer computation delays, the 3 region and 5 region cases are still significantly faster than the cases which contain more regions. However, the 8 regions case turns out to be the slowest one among the cases tested.

In order to quantify results dedicated to the asynchronous resolution of the algorithm, we will focus on the maximum speedup for each partition for a message loss rate of $p_{loss} = 5\%$, also compiled in Table II. It seems that the more regions the problem is split into, the greater the speedup of the asynchronous resolution compared to the synchronous resolution. However, there does not seem to be any rule about the asynchronous parameter δ value corresponding to this maximum speedup. This study should be further developed using more partitions of the same testcase, or even various partitions of different testcases.

E. Effects of communication time

One last parameter affecting the total convergence time is the communication delay. So far, a fixed communication delay $\Delta_{comm} = 50$ ms was considered. For the following results, the message loss rate is set to $p_{loss} = 5\%$ whilst the fixed communication delay Δ_{comm} varies from 25 to 200 ms. Fig. 5 shows the computation and idle time with respect to the fixed communication delay and for three values of the asynchronous parameter δ: 10, 50 and 100%. The synchronous convergence time increases by 370% whereas the asynchronous one ($\delta = 10\%$) increases only by 168%. The asynchronous version is consequently more robust to global communication delay increases.

VI. CONCLUSION AND PERSPECTIVE

In the present contribution, the study of the asynchronous resolution of the decentralized optimal flow presented in [9] has been performed on a 118 bus testcase. Computation and communication delays have been simulated, and their influence over the total convergence time have been studied and compared to the synchronous resolution. We proposed an asynchronous framework of a decentralized OPF algorithm, and showed that it converges towards the optimal solution. Applied to the IEEE 118 bus testcase, the asynchronous resolution is faster than the synchronous one when the message loss rate

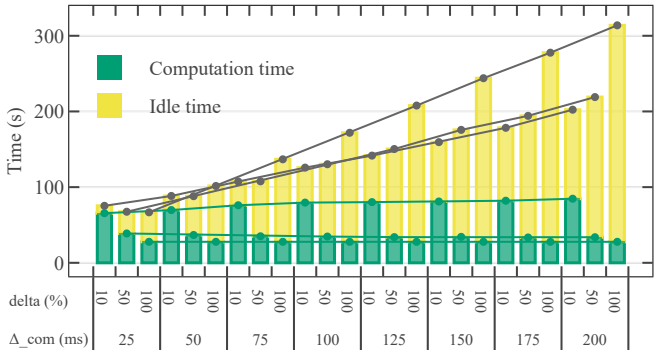

Fig. 5. Average computation and idle time with respect to the fixed communication delay Δ_{comm} and the asynchronous parameter δ. The message loss rate is fixed at $p_{loss} = 5\%$.

is over 2%. We showed that the asynchronous resolution is indeed more robust against longer and varied communication delays than the synchronous resolution. It also allows for a bigger speedup as the number of partition increases. The partitioning influence over the convergence time, however, requires further study. A bigger testcase should be investigated in order to further highlight the influence of the asynchronous resolution over a greater number of regions. Simulations over different partitions of the same testcase for a given number of regions also need to be considered to make the study less dependent on partitioning. This however would require to find the appropriate tuning parameters for every partition.

REFERENCES

[1] A. Dong *et al.*, "Convergence analysis of an asynchronous peer-to-peer market with communication delays," *SEGAN*, vol. 26, p. 100 475, Jun. 2021, ISSN: 2352-4677.

[2] J. Mohammadi *et al.*, "Asynchronous distributed approach for DC Optimal Power Flow," in *IEEE PowerTech 2015*, Aug. 2015, ISBN: 9781479976935.

[3] A. Huang *et al.*, "Asynchronous decentralized method for interconnected electricity markets," *Int. J. Electr. Power Energy Syst.*, vol. 30, no. 4, pp. 283–290, May 2008, ISSN: 01420615. DOI: 10.1016/j.ijepes.2007.10.001.

[4] J. Guo *et al.*, "Impact of communication delay on asynchronous distributed optimal power flow using ADMM," in *SmartGridComm 2017*, 2018, ISBN: 9781538640555. DOI: 10.1109/SmartGridComm.2017.8340718. arXiv: 1711.01702.

[5] M. H. Ullah *et al.*, "Distributed Energy Optimization in MAS-based Microgrids using Asynchronous ADMM," in *ISGT 2019*, Institute of Electrical and Electronics Engineers Inc., Feb. 2019, ISBN: 9781538682326. DOI: 10.1109/ISGT.2019.8791568.

[6] X. Liang *et al.*, "Relaxed alternating direction method of multipliers for hedging communication packet loss in integrated electrical and heating system," *J. Mod. Power Syst. Clean Energy*, vol. 8, no. 5, pp. 874–883, Sep. 2020, ISSN: 21965420. DOI: 10.35833/MPCE.2020.000163.

[7] Y. Li *et al.*, "Event-Triggered-Based Distributed Cooperative Energy Management for Multienergy Systems," *IEEE Trans. Ind. Inform.*, vol. 15, no. 4, pp. 2008–2022, Apr. 2019.

[8] A. Dong *et al.*, "Asynchronous algorithm of an endogenous peer-to-peer electricity market," *IEEE PowerTech 2021*, Jun. 2021.

[9] T. Erseghe, "Distributed optimal power flow using ADMM," *IEEE Trans. Power Syst.*, vol. 29, no. 5, pp. 2370–2380, 2014, ISSN: 08858950. DOI: 10.1109/TPWRS.2014.2306495.

[10] R. Christie, *118 Bus Power Flow Test Case*, 1993. [Online]. Available: http://labs.ece.uw.edu/pstca/pf118/pg_tca118bus.htm (visited on 09/22/2021).