Convergence analysis of an asynchronous peer-to-peer market with communication delays

Alyssia Dong, Thomas Baroche, Roman Le Goff Latimier, H. Ben Ahmed

Convergence analysis of an asynchronous peer-to-peer market with communication delays

Alyssia Dong, Thomas Baroche, Roman Le Goff Latimier, Hamid Ben Ahmed

ENS Rennes, France

Abstract

With the growing number of distributed energy resources, the number of agents partaking in an electricity market is also bound to increase, which highlights the need of a scalable algorithm to clear the market. The interest towards distributed economic dispatch algorithms has grown in the last years as they were found to be scalable, unlike their centralized counterpart. However, these algorithms rely on communication between computational agents during the resolution of the problem. In the case where those computational agents are located far away from each other, the resolution may be impacted by the various hazards that can occur in the communication network. The asynchronous resolution of the algorithm provides a solution to the large variations in communication delays as it allows the agents to carry on with their computation despite not having received every messages from their market partners. This paper focuses on a peer-to-peer market clearing algorithm based on the alternating direction method of multipliers (ADMM) and its asynchronous version. Two communication delay models are introduced in order to study the effect of communication on the convergence of the synchronous and asynchronous versions of the algorithm. In a 110 agent testcase study, we show that the asynchronous resolution can speed up convergence by 40%, while also being more robust to delay variations.

Keywords: Peer-to-peer market, Asynchronous resolution, ADMM, Communication delays

1. Introduction

As the power network includes more and more distributed energy resources such as renewables, the number of producers participating in the electricity market significantly increases. This market consists of an agreement upon the amount of power that each agent, producer or consumer, will respectively deliver to or withdraw from the grid at a given horizon in the future. This economic dispatch problem can be extended by taking into account various degrees of physical constraints, as with the optimal power flow (OPF) problem [1].

Nowadays the market clearing mechanism is being solved as close as 15 minutes before the power delivery [2]. However, because the proportion of highly uncertain renewable energy is bound to increase significantly in the future, it may be interesting to push the market clearing closer to the delivery time in order to be more precise with the production forecasting. Indeed, reducing the production forecast error helps minimizing the power reserves set aside to compensate for these errors. It also reduces the risks of blackout of the power system. Hence, it becomes necessary to speed up, scale and strengthen the process of market clearing in order to solve the energy dispatch problem within a timely manner.

Although the market clearing is historically centralized, it can also be distributed or fully decentralized. In the distributed case, a central agent gather information from all market participants whereas in the fully decentralized case, there is no central agent and the participants communicate directly with each other. The interest for a fully decentralized peer-to-peer market has increased throughout the years as demonstrated by the research and development projects listed in [3], including, but not limited to, the *Enerchain*[1] project, a peer-to-peer platform allowing trading at different scales, from the wholesale market in Europe to local community exchanges, and the *Energy Collective*[2] project in Denmark, which investigates market designs for local energy communities. Indeed, peer-to-peer markets allow heterogeneous preferences, which gives flexibility on the choice of the trading partners, as well as direct communication between partners to speed up the trading process and ensure prompt transmission of energy [4]. The decentralization of the market clearing problem makes it much more scalable while introducing information exchanges between market agents [5].

In comparison to the distributed approach, the peer-to-peer market resolution needs much more message exchanges to arrive to completion, since instead of having to send one message to an unique central agent, each market participant has to send and receive messages directly to and from their commercial partners. Considering that in theory the peer-to-peer market will include hundreds or thousands of participants, the problem of idle time between each iteration presents an added challenge regarding the overall convergence time of the algorithm. In particular, communication hazards can have a bigger impact on the resolution. Provided that the communication protocol does not allow any message loss, the main drawback to distributed and decentralized market resolution lies within the variations in computation and communication delays. Indeed, the algorithms require that all agents compute and communicate their results to the other agents at each iteration. The late arrival of only one message will slow down the iteration for

[1] https://enerchain.ponton.de/

[2] http://the-energy-collective-project.com/

the entire market. Hence, despite the inherent scalability of the distributed and decentralized algorithms, their resolution time are very sensitive to delay variations, be it from computational delays or communication delays.

We will be focusing in this article on a peer-to-peer market clearing algorithm, using the alternating direction method of multipliers (ADMM), based on [6], in which all agents communicate to each other via a network protocol that provides ordered, lossless, bi-directional connections such as the Transmission Control Protocol/Internet Protocol (TCP/IP), and where computation delays are neglected. In order to make the convergence time of the resolution less sensitive to the various communication delays, we will propose an asynchronous resolution of the peer-to-peer algorithm in which the various agents perform their calculations asynchronously. A simple parametric communication delay model is introduced in order to analyze the impact of the communication network over the convergence time of this asynchronous market clearing algorithm.

We will start in section 2 by presenting the state of the art concerning distributed and decentralized asynchronous algorithms used for power networks. Then, in section 3, we will present the synchronous market clearing algorithm, before introducing the asynchronous version in section 4. The simulation framework for the new algorithm will be presented in section 5, while the related results will be explained in section 6.

2. Related work

The distributed resolution of an algorithm to solve power network based problems like OPF, constrained economic dispatch or phase estimation has been widely studied [7, 8, 9], including with ADMM based algorithms [10].

These algorithms are meant to be distributed over several computational agents that communicate with each other at every iteration. The speed up resulting from the distribution of the problem is limited by agents with longer computation and communication delays. This is why the asynchronous versions of these algorithms are introduced. The difference between the synchronous and asynchronous versions lies in the idle time between each iteration of the algorithm. In the synchronous version, an agent has to wait for the arrival of the messages of all its neighbors, whereas in the asynchronous version the agent is allowed to carry on with the next iteration after only receiving partial information.

The asynchronous distributed algorithms were first studied from a computer science approach, as the ADMM consensus algorithm is widely used in machine learning applications. In [11, 12] the asynchronous character of the resolution is performed by choosing for each iteration an agent at random. This agent is then the only one updating its local problem for the iteration considered. In [13, 14], a central master agent collects information from the distributed workers in an asynchronous way, meaning that it does not wait for the results of every workers. The master launches its computation after receiving a minimum number of updates and with a bounded delay condition i.e. each variable information used by the master must be at most τ iterations old. Given this bounded delay condition, the proof of convergence of the asynchronous algorithm is provided in [13]. In [15], the algorithm is distributed block-wise with several servers and workers. The asynchrony comes from the fact that the consensus variable is updated incrementally at each reception of a worker's update. The workers use the latest version of the consensus variable that they have received. In [16, 17], the problem becomes totally decentralized. In both papers, the asynchronism comes from a given bounded delay, and it is shown that it allows to significantly improve the total convergence time. However, the final objective function differs from the synchronous solution. Also, the communication delays are either empirical or drawn randomly from a uniform distribution, so there is no added communication delay coming from a designated model introduced in these papers, which seems reasonable given that the calculation times are predominant over the communication times in computer science.

In a power network context, the distributed agents are not located on the same machine, so the communication hazards can affect the algorithm's ability to run smoothly. There have been several studies of asynchronous algorithm applied to power networks. In [18], the mode estimation problem, which consists of the estimation of the phase and frequency oscillations in the various nodes of the power network, is solved by a distributed asynchronous ADMM with bounded delay thresholds for the central agent and the local agents. They used the same advanced communication delay model as [19], and found out that the convergence error was not monotonous anymore as more asynchrony was added.

The optimal power flow problem is also solved with asynchronous algorithms in [19, 20, 21, 22]. In [20], the general non-linear constrained optimization problem (AC-OPF) is solved with asynchronous ADMM in a completely decentralized manner. The iterations threshold comes from the number of messages that an agent receives, and the communication delays are randomly generated within a range. The AC-OPF is also solved in [21] using an asynchronous Lagrangian relaxation method. The global market is split into several local markets that communicate asynchronously with each other. The order of the messages correspond mostly to the difference of computation delays of the agents. The convergence rate of the asynchronous algorithm is higher than the synchronous one if there is a significant difference in computation speed of the agents. [19] also tackles the OPF problem in a distributed approach with a limited coordination of neighboring nodes. They used an advanced communication delay model and tested several delay-robust strategies to improve their results. The best strategy they found consisted of estimating the missing information due to the asynchronous threshold with a weighted autoregressive model.

The linear approximation of the aforementioned AC-OPF (DC-OPF) was asynchronously solved in [22] in a totally decentralized manner for microgrid applications. They made the agents randomly skip some iterations and found out that the more iterations they skipped, the farther from the optimal solution their results were compared to the synchronous algorithm.

An asynchronous version of the peer-to-peer economic dispatch problem is studied in [23] using both consen-

sus+innovation and ADMM. Computation and communication delays are randomly sampled from a random variable. They found out the asynchronous version took longer to converge as oscillations appeared during the resolution.

The communication delay models in the presented related works are varied. Because asynchronous consensus algorithms were first studied from a computer science approach, the communication delays were considered uniformly across all workers. Indeed, the distributed workers are either on the same computer using message passing interface (MPI) to communicate or are located within the same communication sub-network. Thus, simple communication models such as a fixed probability to arrive at each iteration [14, 15] or a probability to lose the message [17] were used. Some even used empirical communication delays as they directly implemented their algorithms on hardware [13, 16]. Extending the asynchronous algorithms to power network related problems, some papers have chosen to keep using simple communication delay models: [20] samples a delay from a uniform distribution and [21] uses a unique fixed delay of 10 ms and relies on the empirical computation delay differences. However, considering the application of distributed algorithm to power network where the computational agents might not be located in the same areas, these aforementioned communication delays might be limited in regards to real world application. Hence, some papers have used more complex communication delay models as in [23] using an exponential distribution, or in [15, 18] using an advanced delay model for wide area communication. These models account for larger communication delays being sampled more frequently, as experienced in real world application.

This article is a continuation of the work presented in this section in which it presents an asynchronous decentralized algorithm used to solve a power network dispatch problem. We particularly focused on extending the communication model by using a communication delay model which depends on the geographical location of the market participating agents. We first had to try several configurations of the asynchronous algorithm to find the one which leads to the right solution. Then, we empirically showed that the asynchronous version of the algorithm allowed for up to a 40% convergence time saving, while also making it more robust to communication delays variations.

3. Peer-to-peer electricity market algorithm

The classical algorithm used in this paper performs an economic dispatch between all participating agents. Each agent negotiates contracts to trade power with other agents, its commercial or trading partners. The sum of all trades of one particular agent represents the power that this agent will produce or consume.

One of the benefits of using trades, besides enabling a fully decentralized economic dispatch, is the possibility to add preferences to promote or penalize certain exchanges. For example, a consumer can choose to favor renewables producers even if their price is a bit higher than that of a traditional producer. In [6], the tariffs are used to favor exchanges within a region, thus avoiding congestion of weak power lines between regions.

3.1. Optimization problem

The economic dispatch of the market is solved by finding the solution to the following optimization problem:

$$\underset{[p_i]_{i\in\Omega}}{\text{minimize}} \sum_{i\in\Omega} f_i(p_i) \tag{1a}$$

$$\text{subject to} \sum_{i\in\Omega} p_i = 0 \tag{1b}$$

$$\underline{p_i} \le p_i \le \overline{p_i} \qquad i \in \Omega \tag{1c}$$

where:

- p_i represents the power exchanged by agent i, counted as positive if actually produced by the agent and negative if consumed,
- the *cost functions* $f_i(.)$ are convex functions representing the production/consumption costs of agent i,
- $\underline{p_i}$ and $\overline{p_i}$ are limits for production or consumption of agent i.

Equation (1b) represents the global power balance in the network, and equation (1c) sets the physical limits of production/consumption of the agents.

The scope of this contribution is to focus on the impact of the communication network. Therefore for the sake of clarity when interpreting the outcomes, we chose to focus on the simplest problem of economic dispatch. On the basis on the here presented results, we could apply the principles presented in this paper to an OPF problem in future work.

Trades are introduced to describe the commercial exchanges between agents. Let ω_i be the set of trading partners of agent i. If we consider agents to be either electricity producer or consumer, then we can assume that a producer will only have consumers as trading partners, and vice versa. The case where a producer purchases power from another producer to increase its capacity is not covered in this paper. Furthermore, if $j \in \omega_i$, i.e j is one of i's trading partner, then $i \in \omega_j$.

The trade between agent i and its trading partner agent j is noted as t_{ij}, and it indicates which proportion of the power p_i is dedicated to agent $j \in \omega_i$.

$$p_i = \sum_{j\in\omega_i} t_{ij} \tag{2}$$

The balance equation (1b) is fulfilled when there is a trade reciprocity for each trade:

$$t_{ij} = -t_{ji} \qquad i \in \Omega \quad j \in \omega_i \tag{3}$$

i.e. when agent i sells a quantity t_{ij} to agent j, then agent j buys the same quantity from agent i. Each trade exchange then assures the power balance. If $j \notin \omega_i$, then we set $t_{ij} = t_{ji} = 0$.

3.2. Strict convexification

The problem becomes non strictly convex with respect to the variables $(t_{ij})_{(i,j)\in\Omega^2}$ because there can be multiple combinations

that are solution to the minimization problem (1). A regularization term is added in the cost function (4a) so as to make the problem strictly convex with respect to the trade variables.

$$\text{minimize} \quad \sum_{i\in\Omega}\left(f_i(p_i) + \gamma \sum_{j\in\omega_i} {t_{ij}}^2\right) \tag{4a}$$

$$\text{subject to} \quad t_{ij} = -t_{ji} \qquad i \in \Omega \qquad j \in \omega_i \tag{4b}$$

$$p_i = \sum_{j\in\omega_i} t_{ij} \qquad i \in \Omega \tag{4c}$$

$$\underline{p_i} \leq p_i \leq \overline{p_i} \quad i \in \Omega \tag{4d}$$

This added term resembles the one in [6] in as it represents an added cost attributed to the bilateral trade t_{ij}. In this paper, this added cost enforces a more uniform distribution of trades for a given power production or consumption. Hence, it deters the market agents to buy large power quantities at a low cost. In a circumstance where the market agents do not have any limitations on their market partners, it prevents the arbitrage phenomenon where they buy large quantities to then sell at a higher price. We can refer to the added term in (4a) as an arbitrage penalty. We will show in section 6.1 how to properly set the arbitrage penalty factor γ.

3.3. Local algorithm

The distribution of problem (4) is done via the ADMM method, and detailed in [6]. The resolution of the problem is done iteratively in a fully decentralized way. Each agent i computes the following iterations:

$$(p_i, \mathbf{t}_i)^{k+1} = \underset{(p_i,\mathbf{t}_i)}{\arg\min}\ f_i(p_i) + \ldots \tag{5a}$$

$$\sum_{j\in\omega_i}\left[\gamma\, {t_{ij}}^2 + \frac{\rho}{2}\left(\frac{t_{ij}^k - t_{ji}^k}{2} - t_{ij} + \frac{\lambda_{ij}^k}{\rho}\right)^2\right]$$

$$\text{s.t.} \quad p_i = \sum_{j\in\omega_i} t_{ij}$$

$$\underline{p_i} \leq p_i \leq \overline{p_i}$$

$$\lambda_{ij}^{k+1} = \lambda_{ij}^k - \rho\,\frac{t_{ij}^{k+1} + t_{ji}^{k+1}}{2} \qquad j \in \omega_i \tag{5b}$$

with k being the global iteration number, ρ being the penalty factor of the algorithm, and λ_{ij} being the dual variable associated with the element ij of the trade's reciprocity constraint (4b).

Note that (5b) can be written as:

$$\lambda_{ij}^k = \lambda_{ij}^{k-1} - \rho\,\frac{t_{ij}^k + t_{ji}^k}{2} \quad j \in \omega_i \quad k \geq 1 \tag{6}$$

Thus, the process performed by each agent is presented in Algorithm 1.

Algorithm 1 Synchronous Algorithm : Agent i local process

procedure SynchronousAgentProcess($\mathbf{t}_i^0, \boldsymbol{\lambda}_i^0$)
  $\mathbf{t}_i \leftarrow \mathbf{t}_i^0$ ▷ Initialization
  $\boldsymbol{\lambda}_i \leftarrow \boldsymbol{\lambda}_i^0$
  $k \leftarrow 0$
  send t_{ij}^0 to all $j \in \omega_i$
  repeat ▷ Proceed until convergence
    receive t_{ji}^k from all $j \in \omega_i$
    if $k \geq 1$ **then**
      update $\boldsymbol{\lambda}_i^k$ with (6)
    end if
    update $\mathbf{t}_i^{k+1}$ with (5a)
    send updated t_{ij}^{k+1} to all $j \in \omega_i$
    $k \leftarrow k + 1$
  until local residual $\leq \varepsilon$
end procedure

3.4. Residuals

The convergence of the algorithm is characterized by primal and dual residuals. Local residuals are defined for each agent i by:

$$r_i^k = \sum_{j\in\omega_i} (t_{ij}^k + t_{ji}^k)^2 \tag{7a}$$

$$s_i^{k+1} = \sum_{j\in\omega_i} (t_{ij}^{k+1} - t_{ij}^k)^2 \tag{7b}$$

The primal residual r_i reflects the equilibrium of the agent i's trades with its trading partners: the quantity of energy that agent i buys from agent j must correspond to the quantity of energy that agent j sells to the agent i. Thus, $r_i^k \xrightarrow[k\to\infty]{} 0$.

The dual residual s_i is a measure of the variation rate of the trades offered by agent i from one iteration to the next. When the algorithm converges, there are no more variations: $s_i^k \xrightarrow[k\to\infty]{} 0$.

Global residuals are defined by:

$$r^k = \sum_{i\in\Omega} r_i^k \tag{8a}$$

$$s^k = \sum_{i\in\Omega} s_i^k \tag{8b}$$

For the sake of simplicity, we can determine the convergence of the algorithm with the primal global residual (8a) only. We consider that the algorithm has converged at iteration k_{conv} when:

$$r^{k_{conv}} \leq \varepsilon \tag{9}$$

with ε depending on the agents physical limits:

$$\varepsilon = \varepsilon_{rel}^2 \cdot \sum_{i\in\Omega} \max(\underline{p_i}^2, \overline{p_i}^2) \tag{10}$$

This expression of ε takes into account the squared value of the maximum produced or consumed power in the market. It introduces the relative value ε_{rel} which lets us compare the convergence of different testcases.

3.5. Limits of the synchronous algorithm

We can observe in Algorithm 1 that at the beginning of every iteration, every agent stays inactive because it has to wait for all of its partners' messages before continuing iteration calculations. The various communication hazards therefore have a direct effect on the course of the algorithm resolution. The communication protocol used is assumed to take into account message loss and corrupted messages, such as the TCP protocol, which re-send messages when it realizes that they have been lost. A lost or corrupted message then simply leads to a longer communication delay.

If we consider even one communication channel with a high failure rate, then the majority of the waiting time for the entire market would have been spent waiting only for the last message passing through that channel. The more agents that participate in the peer-to-peer market, the longer the expected time of each iteration will be. It means that in terms of overall execution time, this algorithm is not scalable, even though in theory it is completely scalable thanks to the ADMM distribution in operational context. For example, let us consider the 110 agents testcase used in 5.4. For each iteration, 4800 messages in total are sent on the communication network. A 0.1 % loss rate leads to 4.8 lost messages on average per iteration. Using the TCP timeout-based retransmission, the sender re-sends a message after a delay a bit larger than the average round time trip if it does not receive acknowledgment of this message from the receiver. Thus, assuming that the second message does not get lost, the total communication delay for sending one message is going to be on average 3 times what the average communication delay is. Because during an iteration all message exchanges are done in parallel, the longest communication delay will be 3 times slower. Thus, on average, the synchronous algorithm will be 3 times slower to reach convergence than in the case where no messages are lost. The asynchronous algorithm allows the agents to cut this excess idle time by not waiting for all messages to arrive.

This is why we are studying the asynchronous version of the aforementioned algorithm in this paper. The next section will present the principles of asynchrony and the different strategies that can be applied.

Another way to speed up the resolution that is not considered here is to partition the market into regions instead of having each market participant solve its own problem. This reduces the number of exchanges during the resolution, thus limiting the added communication delays discussed above. As demonstrated in [24] for an OPF problem, a spectral clustering based method can be used to find the optimal partition of the problem in order to speed up the convergence. This partition relies on the computational and physical coupling between nodes. For an economic dispatch problem, the market agents can be grouped into connected communities [25] with agents who share the same economic interests. These communities, however, may not be optimally partitioned to speed up the convergence time since the computational and physical coupling is a priori not correlated to the economic interests.

4. Asynchronous peer-to-peer algorithm

4.1. Principle of an asynchronous algorithm

The basic principle behind what is referred to as an *asynchronous algorithm* is that the computations do not take into account all the information at every iteration.

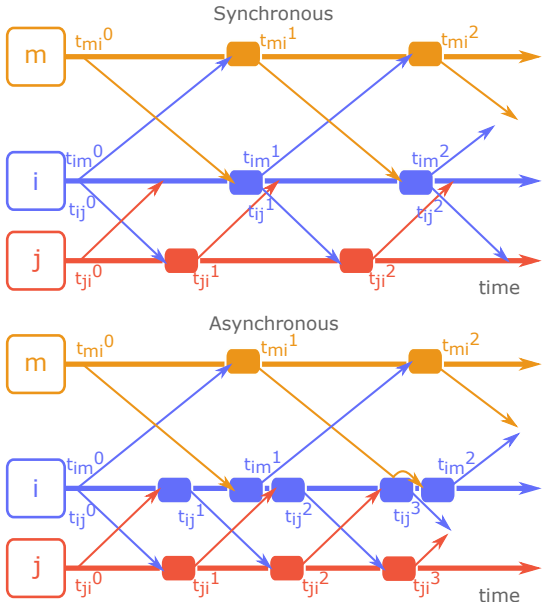


Figure 1: Message exchanges diagram in a three agents market : i, j and m, where $\omega_i = \{j, m\}$ and $\omega_j = \omega_m = \{i\}$, in the synchronous and the asynchronous cases. The diagonal arrows represent the message exchanges over time and the colored rectangles represent the processing time of a local iteration. The asynchronous case here consists of agent i only waiting for one message instead of two before performing its calculations.

In the asynchronous version of our algorithm, local agent i does not have to wait for each of his trading partners update message before proceeding to its calculation, which then reduces the delays of each iteration. Figure 1 represents the message exchanges over time in the synchronous and asynchronous cases for a three agent market. In the asynchronous case, the frequency of agent j's calculations doubles compared to the synchronous version.

However the lack of information that arises from the asynchronism makes the algorithm slower to converge in terms of number of iterations, that is to say it needs more communication. Therefore, a balance must be found between asynchronism and number of exchanged messages.

4.2. Test of the various strategies of the asynchronous algorithm

Because every agent is now advancing at its own pace, a global iteration counter k is no longer relevant in an asynchronous resolution. We define the local iteration counter k_i that corresponds to agent i number of computations. In addition, many resolution schemes are possible because we now dissociate the trade update and the price update of Algorithm 1. We introduce a linkwise iteration counter k_{ij} which indicates the number of updates the trade t_{ij} has been through. Likewise,

we introduce the counter l_{ij} for the number of updates of the dual variable λ_{ij}. Let us note $\Phi_i^{k_i}$ the subset of ω_i from which agent i received updates at the beginning of iteration k_i.

The asynchronous versions of equations (5) are then given by the following equations:

$$(p_i, \mathbf{t}_i)^* = \underset{(p_i,\mathbf{t}_i)}{\arg\min}\ f_i(p_i) + \ldots \tag{11a}$$
$$\sum_{j\in\omega_i}\left[\gamma\, {t_{ij}}^2 + \frac{\rho}{2}\left(\frac{t_{ij}^{k_{ij}} - t_{ji}^{k_{ji}}}{2} - t_{ij} + \frac{\lambda_{ij}^{l_{ij}}}{\rho}\right)^2\right]$$
$$\text{s.t.}\quad p_i = \sum_{j\in\omega_i} t_{ij}$$
$$\underline{p_i} \le p_i \le \overline{p_i}$$
$$\lambda_{ij}{}^* = \lambda_{ij}^{l_{ij}} - \rho\,\frac{t_{ij}^{k_{ij}+1} + t_{ji}^{k_{ji}+1}}{2} \qquad j \in \omega_i \tag{11b}$$

We observe in (11a) that the whole vector $\mathbf{t}_i$ can possibly be updated. Two strategies can be implemented: only update the variables t_{ij} such that $j \in \Phi_i^{k_i}$ (*partial trades update*), or update all t_{ij} for $j \in \omega_i$ (*total trades update*). Likewise, the same kind of choice can be made for the dual variable λ_{ij} : *partial* or *total prices update*. All these possibilities have been tested empirically and compared in Table 1. While some configurations have converged to a solution, there is only one configuration which reached the optimal solution of problem 4: the partial trades and partial prices update configuration. We will be focusing on this configuration for the rest of the manuscript. This configuration has been tested on several IEEE testcases and reached convergence with the condition of bounded communication delays. The obtained results are equal to the synchronous version of the algorithm, which we can consider optimal since the objective function is convex. While we do not provide a mathematical proof of convergence for this particular asynchronous algorithm, a proof of convergence for a distributed asynchronous consensus ADMM problem is provided in [13], given a bounded delay condition. Another proof of convergence is provided in [17] for a decentralized relaxed ADMM asynchronous algorithm, following the assumption that every message has a non zero probability to arrive to destination at each iteration and that those probabilities are mutually independent over time. Let us note that in our case, the communication models presented in 5.2 respect the bounded delay condition, but that the order of exchanged messages can not ensure the independence of the random variables describing the arrival of messages.

Table 1: Asynchronous algorithm configurations under partial communication

	Partial trades update	Total trades update
Partial prices update	Convergence : yes Optimality : yes	Convergence : yes Optimality : no
Total prices update	Convergence : no Optimality : no	Convergence : yes Optimality : no

It follows that the linkwise iteration counters for the trades and for the dual variables are equal $k_{ij} = l_{ij}$ and that they are increased when $j \in \Phi_i^{k_i}$.

Another constraint makes the algorithm perform better: $\Phi_i^{k_i}$ is set in a way that the received trade update message t_{ji} is associated to the tradewise iteration $k_{ji} = k_{ij}$. If a message is received where $k_{ji} > k_{ij}$, it is stored in memory and will be taken into account in the next iteration when $k_{ji} = k_{ij}$. The resulting equations are then given as follows:

$$\lambda_{ij}^{k_{ij}} = \lambda_{ij}^{k_{ij}-1} - \rho\,\frac{t_{ij}^{k_{ij}} + t_{ji}^{k_{ji}}}{2} \qquad j \in \Phi_i^{k_i} \qquad k_{ij} \ge 1 \tag{12a}$$
$$(p_i, \mathbf{t}_i)^* = \underset{(p_i,\mathbf{t}_i)}{\arg\min}\ f_i(p_i) + \ldots \tag{12b}$$
$$\sum_{j\in\omega_i}\left[\gamma\, {t_{ij}}^2 + \frac{\rho}{2}\left(\frac{t_{ij}^{k_{ij}} - t_{ji}^{k_{ji}}}{2} - t_{ij} + \frac{\lambda_{ij}^{k_{ij}}}{\rho}\right)^2\right]$$
$$\text{s.t.}\quad p_i = \sum_{j\in\omega_i} t_{ij}$$
$$\underline{p_i} \le p_i \le \overline{p_i}$$
$$t_{ij}^{k_{ij}+1} = t_{ij}^* \qquad j \in \Phi_i^{k_i} \tag{12c}$$

and the asynchronous version of the algorithm is given in Algorithm 2.

Algorithm 2 Asynchronous Algorithm : Agent i local process

procedure AsynchrounousAgentProcess($\mathbf{t}_i^0, \boldsymbol{\lambda}_i^0$)
- $\mathbf{t}_i \leftarrow \mathbf{t}_i^0$ ▷ Initialization
- $\boldsymbol{\lambda}_i \leftarrow \boldsymbol{\lambda}_i^0$
- $k_{ij} \leftarrow 0$ for $j \in \omega_i$
- $k_i \leftarrow 0$
- send t_{ij}^0 to $j \in \omega_i$
- **repeat** ▷ Proceed until convergence
  - receive $t_{ji}^{k_{ji}}$ from $j \in \Phi_i^{k_i}$ such as $k_{ji} = k_{ij}$
  - update λ_{ij} with (12a) for $j \in \Phi_i^{k_i}$ if $k_{ij} \ge 1$
  - update $\mathbf{t}_i$ with (12b-12c)
  - send updated $t_{ij}^{k_{ij}+1}$ to $j \in \Phi_i^{k_i}$
  - $k_{ij} \leftarrow k_{ij} + 1$ for $j \in \Phi_i^{k_i}$
  - $k_i \leftarrow k_i + 1$
- **until** local residual $\le \varepsilon$

end procedure

4.3. Determination of the set $\Phi_i^{k_i}$

The order of arrival of the messages sent by agents $j \in \omega_i$ to agent i defines the set $\Phi_i^{k_i}$. In the synchronous algorithm, $\Phi_i^{k_i} = \omega_i$ therefore agent i must wait for the messages coming from all of its partners in order to proceed to the next iteration, which results in long idle times waiting only for a few messages, thus delaying the entire market.

In the asynchronous version, we have to determine the set $\Phi_i^{k_i}$ for each of agent i iteration k_i. In related works, the set is either determined by:

- the arrival of messages within a given delay between each local iteration k_i [18, 19],
- the arrival of a given number of messages since the previous iteration [20],
- the first condition to be fulfilled between both of the above [16, 14],
- a probability that the message arrive to the agent [17, 13].

The first three points aforementioned necessitates a communication delay model while the last one is more simple to implement.

Let $j \in \omega_i$ and let us note $X_{ji}^{k_i}$ the random boolean variable describing if $j \in \Phi_i^{k_i}$, i.e. if the message t_{ji} sent by j arrived to agent i to be processed during its iteration k_i. Thus:

$$\Phi_i^{k_i} = \left\{ j \in \omega_i \mid X_{ji}^{k_i} = 1 \right\} \tag{13}$$

The proof of convergence of the asynchronous algorithm in [13], where there is a central master agent M that collects the workers variables, requires all the random variables $X_{jM}^{k_M}$ to be independent and identically distributed. Which means that at each of the master iteration k_M, all the workers updates have the same chance to be processed by the master.

This condition might be correct in the case of parallel computing where the communication takes place on a local network (as with MPI communication), but it is limited when it comes to communication between agents that are located far away (for example if they are not part of the same local network). In fact, the order of arrival of the various messages and their delays depends on the communication network, which cannot be properly modeled by a random variable such as $X_{ji}^{k_i}$. If two agents are really close to one another compared to the rest of their trading partners, then they will have a lot more updates on their common trade.

In [15] however, the convergence is proved with the assumption of bounded delays for each link. We will introduce a communication delay model that respects this assumption.

The iteration trigger chosen in this study is the number of messages arrived since last iteration. The messages that arrive after the trigger (i.e. during the computation) are kept in a local buffer and considered as received messages for the next iteration.

We can note that this choice of trigger makes the random variable $X_{ji}^{k_i}$ dependent from the other variables $X_{j'i}^{k_i}$ with $j' \in \omega_i \backslash \{j\}$ because the chance that a message sent from agent j to i arrives in time to participate in iteration k_i depends on the number of messages that agent i already received since its last iteration. It also needs a communication delay model in order to compute the order of arrival of the messages.

Let us note $N_{trig,i}$ the number of messages received by agent i that triggers the calculations of the iteration. Given that agent i has $|\omega_i|$ trading partners which it communicates with, and that it only counts messages that have $k_{ji} = k_{ij}$, there can only be one message per partner, thus $N_{trig,i} \leq |\omega_i|$ messages.

The case where $N_{trig,i} = |\omega_i|$ for all $i \in \Omega$ refers to the synchronous algorithm.

Considering one market Ω, it would be inconceivable to study all possibilities of $N_{trig,i}$ for each $i \in \Omega$ for combinatorial considerations. We therefore set a ratio $\delta \in [\,0, 1]$ such that:

$$N_{trig,i} = \lceil \delta \cdot |\omega_i| \rceil \qquad i \in \Omega \tag{14}$$

with $\lceil \cdot \rceil$ being the ceiling function. The ceiling function has been preferred over the floor function so that $N_{trig,i}$ is at least equal to 1. For example, if agent i has a total of 100 partners, $\delta = 0.1$ would set $N_{trig,i}$ to 10 messages. The case $\delta = 1$ corresponds to the synchronous resolution as $N_{trig,i} = \lceil |\omega_i| \rceil = |\omega_i|$. We will refer to δ as the *asynchronism parameter*.

5. Simulation framework

5.1. Simulation environment

The simulation of the optimization problem resolution given all the exchanged messages was implemented in the Julia language. The package *SimJulia* was used as a discrete-event simulation environment, whereas the sub-problems given by (12b) were solved by the package *OSQP*, which stands for operator splitting quadratic program. This package solves convex quadratic problems [26].

The use of Julia, which is a just-in-time compiled programming language, made the final code efficient and fast enough that it allowed the computation of many market resolutions in a short amount of time, as is necessary when realizing a Monte-Carlo simulation. The calculations were performed with a Intel Core i7-8850H CPU@2.60GHz and each simulation took 5 seconds on average to compute with the 110 agents testcase presented in 5.4, and the gaussian communication model presented in 5.2.

The values of the variables during the resolution are saved at a given sampling frequency. This allows post-processing calculations.

5.2. Communication delay models

Among all the communication hazards, we only take into account communication delays in this paper. Indeed, we can assume that the communication protocol that is used handles retransmission and error-detection, as with the TCP/IP protocol. Thus, communication hazards such as packet loss or corruption results in the application layer as an added delay.

We introduce two stochastic communication delay models: a gaussian based model and a more advanced model which presents a distribution that fits real Internet communication delays. Our models are very simple and only depend on three parameters, which allows us to have a better interpretation of the results based on these parameters.

Let d_{ij} be the distance between agents i and j. This distance is not directly associated with physical distance, but more with distance in terms of the communication network (e.g. number of routers that the message go through). The first communication delay model consists of a random gaussian variable,

presented in (15), which represents the communication delay between agents i and j.

$$\Delta T^*_{ij} \sim \mathcal{N}\left(\alpha\, D_{ij} + \beta,\, \sigma_{ij}\right) \tag{15}$$

The gaussian model was chosen for its simplicity and sampling speed.

- The mean of the communication delay $\mathbb{E}[\,\Delta T^*_{ij}] = \alpha\, D_{ij} + \beta$ takes into account the distance D_{ij} but also a constant minimum delay β. If $\alpha = 0$, then all communication delays have the same mean value.

- The standard deviation is proportional to the mean communication delay. Let us set the parameter $\sigma \in [\,0, 1]$ such that when $\sigma = 1$, then $3\,\sigma_{ij} = \mathbb{E}[\,\Delta T^*_{ij}]$. Thus : $\sigma_{ij} = \frac{\sigma}{3}\,\mathbb{E}[\,\Delta T^*_{ij}]$.

Because the presented gaussian variable can be negative, the real random variable considered for communication delay is $\Delta T_{ij} = \max(\Delta T^*_{ij}, 0)$.

The proposed communication delay model takes into account the global communication network state:

- network congestion is associated to the linear delay parameter α;

- whereas the minimum communication delay is associated to the constant delay parameter β;

- finally, added communication delays due to packet retransmission can be seen as the stochastic delay parameter σ, but it appears that the gaussian model is quite limited in that regard.

In section 6.4, we chose to compare the gaussian delay model with a more advanced model $\Delta T'_{ij}$, inspired by [18, 19], which has a long tail distribution that showcases longer delays at a higher probability. This model features four parameters which take into account the Internet traffic delays as well as the routers processing delays. The probability density function (PDF) $\phi(t)$ of the end-to-end communication delay is given as follows:

$$\phi(t) = p'\,\phi_2(t) + (1 - p')\,\phi_1(t) * \phi_2(t), \qquad t \geq 0 \tag{16}$$

where $\phi_1(t)$ is the PDF of the Internet traffic delay, $\phi_2(t)$ is the PDF of the router processing delay and $\phi_1(t) * \phi_2(t) = \int_0^t \phi_2(u)\phi_1(t-u)du$. Here, p' is the probability of open period of the path with no Internet traffic, it is set to $p' = 0.58$ for all communication links in the market, following the numerical value of [18]. The Internet traffic delay is approximated by a gaussian PDF with expected value μ' and variance σ'^2. The router processing delay is modeled by an alternating renewal process with exponential closure period when the Internet traffic is on, with the PDF $\phi_2(t) = \lambda'\, e^{-\lambda' t}$ where λ'^{-1} represents the mean length of the closure period.

For comparison purposes, the parameters $(\lambda',\, \mu',\, \sigma')$ are tuned to fit the mean value and variance of the gaussian communication delay model for each communication link:

$$\begin{aligned}(\lambda'_{ij},\, \mu'_{ij},\, \sigma'_{ij}) = \underset{(\lambda', \mu', \sigma')}{\arg\min} &\left(\alpha D_{ij} + \beta - \mathbb{E}\left[\Delta T'_{ij}(\lambda',\, \mu',\, \sigma')\right]\right)^2 \\ &+ \left(\sigma^2_{ij} - \mathbb{V}\left[\Delta T'_{ij}(\lambda',\, \mu',\, \sigma')\right]\right)^2\end{aligned} \tag{17}$$

Finally, the delays are sampled with an inverse transform sampling method, using the cumulative distribution function given in [18]. It takes on average $130\,ns$ to sample a delay with the gaussian model, whereas it takes on average $30\,\mu s$ to sample a delay with the advanced model, which makes the simulation about 200 times slower. We will investigate in what extent the gaussian model can be a simpler yet representative alternative to the advanced model regarding the convergence time of the algorithm.

5.3. Computation delays

We consider in this article that the computation delays are negligible compared to the communication delays. Indeed, after a short survey of ping requests to various servers, we can approximate the order of magnitude of the end-to-end Internet communication delay from 10 ms to 100 ms. However, the measured computation delays at each iteration for the testcase presented in section 5.4 are at most equal to a few milliseconds for the agents with the most trading partners, and equal to 0.2 ms for the other agents. For simplicity sake, we decided to neglect the computation delays, but it would be interesting to study their influence on the convergence time when the computation delays and the communication delays share the same order of magnitude.

5.4. Testcase presentation

The testcase used in this article is composed of 110 agents including 30 producers and 80 consumers. Their parameters were randomly generated and are given in the appendix. In this testcase, the set of trading partners ω_i of each consumer i is equal to the set of all producers. Therefore, every producer's set of trading partners is equal to the set of all consumers. Each agent is given a randomly generated 2D location. The distance d_{ij} between agents i and j is then computed as the euclidean distance between both agents' locations. The ADMM penalty factor is set to $\rho = 10$.

5.5. Convergence requirement

We consider that the algorithm has reached convergence when the primal residual invalidates the relation given by (9), with $\varepsilon^2_{rel} = 10^{-9}$. The values of the problem's variables at convergence are noted p^*_i and t^*_{ij}.

6. Simulation results

6.1. Influence of the arbitrage penalty factor γ

We will illustrate the role of the arbitrage penalty term in (4a). Let us first set the arbitrage penalty factor γ to zero. As

explained in 3.2, when $\gamma = 0$, the problem (4) is not strictly convex with respect to the trade variables t_{ij}, whereas it is strictly convex with respect to the power variables p_i. This results in solutions that are distinct from one another when the problem is solved synchronously or asynchronously. Yet, we need all solutions to be equal in order to compare the performances of the algorithm.

At convergence, the power variables are all very close to the synchronous solution ($\delta = 1$) for any value of the asynchronism parameter δ. However, the trade variables are different, as illustrated in Table 2. This table shows the power variable value of agent 32 and its first four trade variables for two values of δ. As we can see, the power variable is equal for all values of δ, whereas the trade variables are different from one another.

Table 2: Values of the optimization variables at convergence for various values of δ, $\gamma = 0$ and $\varepsilon_{rel} = 10^{-12}$

Optimization variable	Value for $\delta = 1$	$\delta = 0.7$	$\delta = 0.2$
p_{32}	-44.03	-44.03	-44.03
$t_{32,1}$	-1.58	-1.69	-1.73
$t_{32,2}$	-1.73	-1.94	-1.72
$t_{32,3}$	-1.44	-1.31	-1.31
$t_{32,4}$	-0.52	-0.10	-0.00

The initial problem (1) is effectively solved by the peer-to-peer algorithm, be it synchronous or asynchronous. However, we want the trade solutions to be equal for any value of δ.

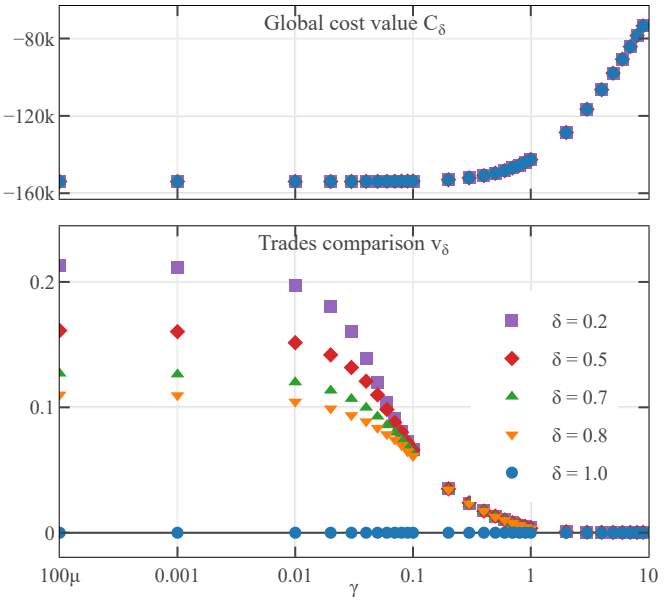


Figure 2: Global cost value at convergence and tradewise distance v_δ from the synchronous solution as a function of the arbitrage penalty factor γ (log scale).

Let's set the synchronous solution as reference for comparison : $p^*_{i,\delta=1}$ and $t^*_{ij,\delta=1}$ for $i \in \Omega$ and $j \in \omega_i$. Two quantities are plotted in Figure 2 as a function of the arbitrage penalty factor γ:

- the global cost value at optimum

$$C_\delta = \sum_{i \in \Omega} f_i(p^*_{i,\delta}) \tag{18}$$

- the normalized tradewise difference between the synchronous solution and other asynchronous solutions

$$\mathrm{v}_\delta = \mathrm{v}\left(\mathbf{T}^*_\delta, \mathbf{T}^*_{\delta=1}\right) = \frac{\sum_{i \in \Omega} \sum_{j \in \omega_i} |t^*_{ij,\delta} - t^*_{ij,\delta=1}|}{\sum_{i \in \Omega} \sum_{j \in \omega_i} |t^*_{ij,\delta=1}|} \tag{19}$$

We observe in Figure 2 that the optimal cost value is always equal to the synchronous solution whatever the value of δ, which is expected considering that all power variables p_i are equal. The tradewise difference decreases towards zero along with γ increasing, which means the trades solutions t_{ij} tends to be equal. However, the global cost increases with γ which means the optimal solution is different from the case where $\gamma = 0$.

Table 3: Global cost increase compared to the case $\gamma = 0$ and the associated normalized tradewise difference

γ	Global cost increase	Tradewise difference
0	0	21%
1.0	+8%	0.4%
9.0	+48%	0.03%

This figure can help us pick which γ value lets us have the smallest tradewise difference $\mathrm{v}\left(\mathbf{T}^*_\delta, \mathbf{T}^*_{\delta=1}\right)$ for all δ, without modifying the optimal cost value too much. For the following results, the γ variable will be set to 1.0. This value allows to have a maximum normalized tradewise difference $v_{0.2} = 0.4\%$ for a 8% increase in the global cost compared to the case $\gamma = 0$, as shown in Table 3.

6.2. *Influence of the communication network state on the global convergence time*

The state of the communication network is modeled by the parameters linear and constant delay parameters α and β in Eq. (15). As communication delays increase, the convergence time of the algorithm increases too.

Figure 3 shows the influence of the linear and constant delay parameters α and β on the convergence time for different values of the asynchronism parameter δ, in the deterministic case i.e. $\sigma = 0$, and with the arbitrage penalty factor set to $\gamma = 1.0$. As we can see in all four figures, the convergence time varies almost linearly with α and β.

Table 4 presents the median values of the gradient of the convergence time (CT) with respect to α and β. We can observe in the fourth column that the ratio between the derivatives is approximately equal to 1 except when the asynchronism parameter $\delta = 1.0$. This shows that both parameters α and β have the same impact on the convergence time of the algorithm for the asynchronous resolutions. However, for the synchronous

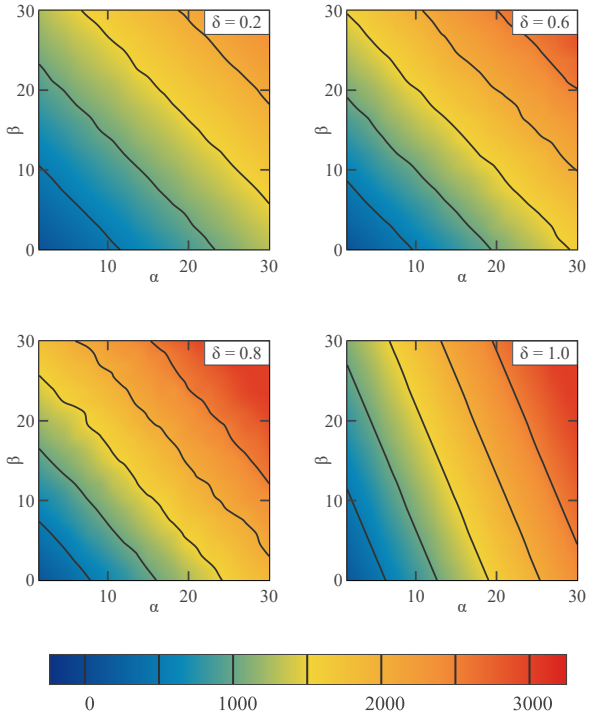


Figure 3: Convergence time (in time units) as a function of the communication network parameters α and β for several values of the asynchronism parameter δ. Fixed parameters : $\gamma = 1.0$ and $\sigma = 0$.

Table 4: Gradient of the convergence time with respect to the parameters α and β

Asynchronism parameter δ	$\frac{\partial \mathrm{CT}}{\partial \alpha}$ (*)	$\frac{\partial \mathrm{CT}}{\partial \beta}$ (*)	$\frac{\partial \mathrm{CT}}{\partial \alpha} / \frac{\partial \mathrm{CT}}{\partial \beta}$
1.0	80.0	33.3	2.40
0.8	60.0	53.3	1.13
0.6	53.3	46.7	1.14
0.4	46.7	40.0	1.17
0.2	40.0	40.0	1.00

(*)Median value over all points.

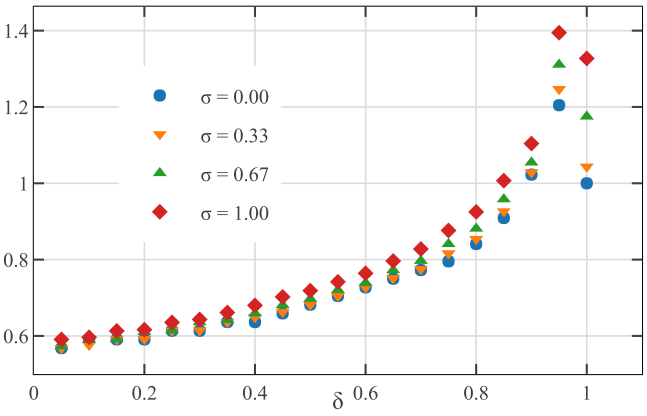
resolution, there is a greater impact of the parameter α which makes the synchronous algorithm more sensitive to markets where communication delays are varied.

In the next section, we will study the influence of the asynchronism parameter δ on the convergence time.

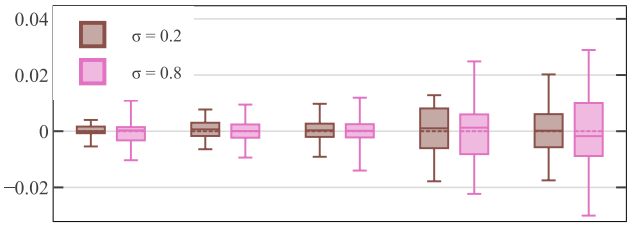
6.3. *Influence of the asynchronism parameter δ on the global convergence time and on the number of exchanged messages*

From this point forward, we will consider that the parameters of the communication model (α, β) are equal to fixed values (α_0, β_0), and we will study the influence of the asynchronism parameter δ and of the stochastic parameter σ on the algorithm for both the gaussian and the advanced delay model presented in 5.2.

The chosen simulation parameters are $(\alpha_0, \beta_0) = (5.0, 1.0)$ which gives an average communication delay between all



(a) Normalized average convergence time $\mathbb{E}(\mathrm{ct}(\delta, \sigma))$ as a function of δ



(b) Boxplot of the centered normalized convergence time $\mathrm{ct}'(\delta, \sigma)$

Figure 4: Values of the convergence time as a function of the asynchronism parameter δ for several values of σ, using the gaussian communication delay model. Figure (a) represents the average value of $\mathrm{ct}(\delta, \sigma)$ while figure (b) represents the quartiles of $\mathrm{ct}'(\delta, \sigma)$.

agents of $\alpha_0 \cdot 1 + \beta_0 = 6$ time units $= 60$ ms. We set the time unit to 10 ms, which corresponds to the lower order of magnitude of the end-to-end Internet communication delay, as shown in section 5.3. The fact that σ is no longer equal to 0 means that the simulation becomes stochastic with regards to communication delays. Therefore, we need to compute several simulations for a given σ and study the statistics of the results, as we would in a Monte-Carlo simulation. The number of draws for each parameter group value is $N_{draws} = 50$.

Let us set as reference the convergence time of the synchronous and deterministic resolution of the peer-to-peer market. The normalized convergence time is a stochastic variable defined in Eq. (20) and its centered version is defined in Eq. (21).

$$\mathrm{ct}(\delta, \sigma) = \frac{\mathrm{CT}(\delta, \sigma)}{\mathrm{CT}(\delta = 1, \sigma = 0)} \tag{20}$$

$$\mathrm{ct}'(\delta, \sigma) = \mathrm{ct}(\delta, \sigma) - \mathbb{E}\left[\mathrm{ct}(\delta, \sigma)\right] \tag{21}$$

The reference convergence time value is equal to $\mathrm{CT}(\delta = 1, \sigma = 0) = 440$ time units $= 4,4$ s. Given that the average communication delay is equal to 60 ms and that the computation delays have been neglected, it takes approximately 73 exchanges between agents before convergence of the algorithm.

We focus at first on the gaussian communication delay model. The average value of $\mathrm{ct}(\delta, \sigma)$ is plotted as a function of δ in Figure 4a for several values of σ. The box plot of the centered variable $\mathrm{ct}'(\delta, \sigma)$ is shown in Figure 4b.

Several remarks can be made from Figure 4:

- $\mathbb{E}(\text{ct})$ is strictly increasing with the asynchronism parameter δ, except in the synchronous case $\delta = 1$.
- $\mathbb{E}(\text{ct}) < 1$ for $\delta \leq 0.8$, which means that the asynchronous cases are faster at reaching convergence compared to the synchronous algorithm. However, $\mathbb{E}(\text{ct}) > 1$ for $\delta \geq 0.9$, which means that the synchronous case is faster. This can be explained by the fact that when $\delta \geq 0.9$, the waiting time of each agent is almost as long as in the synchronous case, but they have to compute more iterations in order to compensate for the lack of information.
- $\mathbb{E}[\text{ct}(\delta = 0, \sigma = 0)] \simeq 60\%$ resulting in a 40% speedup against the synchronous reference case is obtained for the smallest possible δ, which corresponds to $N_{trig,i} = 1$ for all $i \in \Omega$. As soon as an agent receives a message, it proceeds to make an update and send back the computed value to the original sender.
- For a given δ, $\mathbb{E}(\text{ct})$ increases with the stochastic parameter σ, i.e. the variations of the communication delays.
- The box plot in Figure 4b shows that the normalized convergence time dispersion gets smaller when δ gets smaller. This implies that the convergence time becomes more predictable when the algorithm gets more asynchronous, and that it is less affected by the variations in communication delays.

6.4. *Comparison of the gaussian and the advanced communication delay models*

The communication delay model that has been used in section 6.3 was the gaussian model, which is faster to compute than the more realistic model presented in section 5.2. In this section, we want to compare the convergence time results and discuss the advantages and drawbacks of both the gaussian and the more advanced models.

To do so, the stochastic delay parameter is set to $\sigma = 0.2$. As described earlier, the model parameters are chosen so that the expected value and the variance of both models match. The average values of the normalized convergence time are displayed in Table 5 for various values of the asynchronous parameter δ. First we observe that the convergence time is greatly underestimated when using the gaussian model instead of the advanced one for the synchronous version ($\delta = 1$). This can be explained by the fact that the advanced model has a long tail distribution, which means that longer delays have more chance to be sampled than with the gaussian model. This slows down the synchronous resolution since the agents have to wait for every messages to arrive before they can resume their computation. This does not affect as much the asynchronous resolution, hence we do not see a big discrepancy in convergence time when using the gaussian or the advanced model.

Figure 5 represents a boxplot of the centered normalized convergence time $ct'(\delta, \sigma = 0.2)$, comparing both communication delay models. The same comments as for Table 5 can be made here: the use of the gaussian model gives similar results as the advanced model for the asynchronous algorithm, however it underestimates the dispersion of the convergence time for the synchronous algorithm.

Table 5: Average normalized convergence time comparison between both communication delay models, at a fixed stochastic delay parameter $\sigma = 0.2$.

δ	Gaussian model $\mathbb{E}[ct]$	Advanced model $\mathbb{E}[ct]$
0.2	0.588	0.593
0.6	0.714	0.722
1.0	1.348	1.015

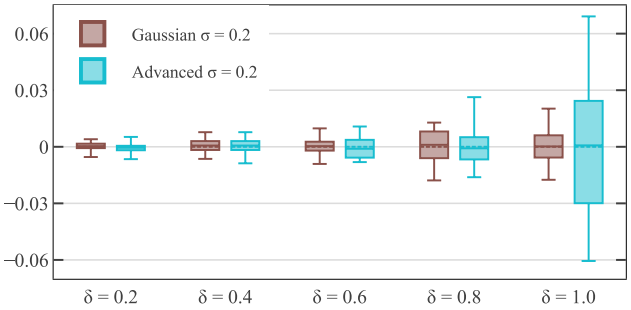


Figure 5: Boxplot of the centered normalized convergence time $\text{ct}'(\delta, \sigma = 0.2)$ comparing the gaussian and the advanced delay models.

In conclusion, the gaussian communication delay model is a good alternative to the more advanced model when simulating the asynchronous resolution of the algorithm. However, it underestimates both the average value and the dispersion of the convergence time of the synchronous resolution. For the rest of this article, the gaussian model will be used but we must keep in mind that the more realistic convergence time of the synchronous resolution is going to be larger than approximated by the gaussian model.

6.5. *Influence of the asynchrony on the number of exchanged messages*

We showed in Figure 4 that the more asynchrony the better in terms of convergence time. However, this decrease in convergence time is traded off against an increased number of exchanged messages.

Let us set the reference number of exchanged messages as before:

$$n(\delta, \sigma) = \frac{N_{mes}(\delta, \sigma)}{N_{mes}(\delta = 1, \sigma = 0)} \tag{22}$$

with N_{mes} being the number of exchanged messages to achieve convergence, and $N_{mes}(\delta = 1, \sigma = 0) = 158\ 400$ messages, which corresponds to an average of 66 messages per commercial link. The number of exchanged messages may be smaller considering other stopping criteria as in [27].

We present in Figure 6 the normalized number of exchanged messages as a function of δ. The gain in convergence time is

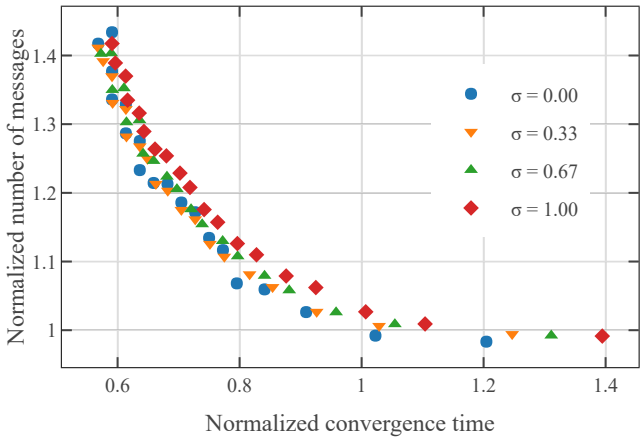


Figure 6: Normalized number of messages to complete the optimization $n(\delta, \sigma)$, as a function of the normalized convergence time $ct(\delta, \sigma)$, and for several values of the stochastic parameter σ.

achieved at the expense of an increase of communication exchanges. There is a balance to find between the number of messages and the convergence time. Moreover, the number of messages increases when the global convergence time decreases, thus resulting in a higher network congestion that might increase the communication delays.

Figure 6 also shows that the stochastic parameter σ, which represents the variations of communication delays, does not affect significantly the number of exchanged messages for a given convergence time.

6.6. *Influence of distance on the tradewise final convergence*

Up to this point we have studied the effect of the communication network state on the global convergence time. We are going to study the local, trade by trade, algorithm progress at the time of convergence.

For every agent $i \in \Omega$ and every partners $j \in \omega_i$, the tradewise primal residual (TR) is computed at the time of convergence:

$$\mathrm{TR}(i,\ j) = \left(t_{i,j}^{k_{i,conv}} + t_{j,i}^{k_{j,conv}}\right)^2 \tag{23}$$

We can note that it corresponds to a single term in the equation (7a). The various values of $\mathrm{TR}(i,\ j)$ are shown in Figure 7 as a function of the normalized distance d_{ij} between the agents i and j:

$$d_{ij} = \frac{D_{ij}}{\max_{i',j'} D_{i'j'}} \tag{24}$$

with D_{ij} as introduced in Eq. (15).

In the case $\delta = 1.0$, i.e. the synchronous case, the distance between agents has no impact on the tradewise residuals at convergence because whatever the communication delays, agents take into account everyone of their trading partners at every iteration.

In the case $\delta = 0.1$, we can see that the agents that are located closer to each other could communicate back and forth more frequently, thus reducing their local residuals faster. We can also notice that the dispersion of the tradewise residuals has greatly increased.

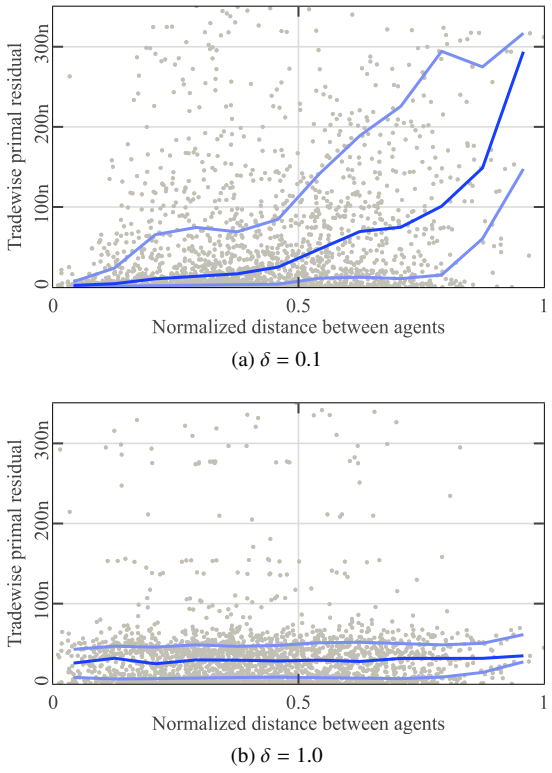


Figure 7: Values of the tradewise primal residuals $\mathrm{TR}(i, j)$ for $i \in \Omega$ and $j \in \omega_i$ at the time of convergence as a function of the normalized distance between agents d_{ij}. The blue lines represent the 25%, 50% and 75% quantiles.

Because we consider the global convergence as presented in Eq. (9), the fact that the closer agents' residuals are small allows for the global algorithm to converge sooner, despite the furthest agents' residuals being higher than in the synchronous case. This shows that a stopping criteria as in [27] could stop exchanges between closely located agents, thus diminishing the total number of messages.

From a market perspective, the participation of an agent to the market could take into account communication delays criteria.

7. Conclusion and future work

The asynchronous version of the peer-to-peer market clearing algorithm presented some challenges concerning the variables' updates, and the non-strictly convex formulation, that were discussed in this paper. We showed that a strictly convex formulation was necessary in order to obtain identical results in the asynchronous formulation as in the synchronous one. A simple gaussian communication delay model has been proposed and used to compare the performance of the asynchronous algorithm to its synchronous version. This model follows the assumption of bounded delays for each communication link. A more realistic communication model has also been used to assess the limits of the gaussian model. The gaussian model provides good estimation of the convergence time for the asynchronous version of the algorithm, however it is slightly more optimistic regarding the convergence time of the

synchronous algorithm. The asynchronous algorithm converges towards the optimal solution, and it does so faster than the synchronous algorithm. It has been shown that the gain in convergence time was due to the frequent communications between agents that are close in regards to the communication network. This gain in time is made at the expense of the number of messages exchanged.

A few points of the proposed algorithm deserve further inspections. This algorithm may be implemented on hardware with actual communication between agents. This will add real computation and processing delays, which can be used to justify, or not, the negligible computation delays assumption made in this paper. Thanks to the asynchronous algorithm, the computation can continue even if one agent is disconnected from the communication network. It can be interesting to demonstrate how to handle such a disconnection in a safe way using the hardware implementation. Also, the arbitrage penalty term may be replaced as in [6] with different tariffs for each trades in order to give preference to some exchanges over other ones. As explained in the introduction, the aim of our proposed asynchronous algorithm is to reduce convergence time in order to get closer to real-time market resolution. The natural continuation of this work would then be to implement a time varying optimization with a warm-starting strategy. Another continuation of this work would be to find the conditions relative to the communication network to allow or not allow a new agent to enter the peer-to-peer market, i.e. to prevent a slow down of the market resolution.

References

[1] P. Panciatici, M. C. Campi, S. Garatti, S. H. Low, D. K. Molzahn, A. X. Sun, L. Wehenkel, Advanced optimization methods for power systems, in: Proceedings - 2014 Power Systems Computation Conference, PSCC 2014, Institute of Electrical and Electronics Engineers Inc., 2014.

[2] L. Baringo, M. Rahimiyan, Virtual power plants, in: Virtual Power Plants and Electricity Markets, Springer, 2020, pp. 1–7.

[3] T. Sousa, T. Soares, P. Pinson, F. Moret, T. Baroche, E. Sorin, Peer-to-peer and community-based markets: A comprehensive review, Renewable and Sustainable Energy Reviews 104 (2019) 367–378.

[4] O. Jogunola, A. Ikpehai, K. Anoh, B. Adebisi, M. Hammoudeh, S.-Y. Son, G. Harris, State-Of-The-Art and Prospects for Peer-To-Peer Transaction-Based Energy System, Energies 10 (2017) 2106.

[5] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, J. Lavaei, A Survey of Distributed Optimization and Control Algorithms for Electric Power Systems, 2017.

[6] T. Baroche, P. Pinson, R. Le Goff Latimier, H. Ben Ahmed, Exogenous Cost Allocation in Peer-to-Peer Electricity Markets, IEEE Transactions on Power Systems 34 (2019) 2553–2564.

[7] F. F. Wu, P. Varaiya, Coordinated multilateral trades for electric power networks: Theory and implementation, International Journal of Electrical Power and Energy Systems 21 (1999) 75–102.

[8] H. Pourbabak, Q. Alsafasfeh, W. Su, Fully Distributed AC Optimal Power Flow, IEEE Access 7 (2019) 97594–97603.

[9] B. H. Kim, R. Baldick, A comparison of distributed optimal power flow algorithms, IEEE Transactions on Power Systems 15 (2000) 599–604.

[10] T. Erseghe, Distributed optimal power flow using ADMM, IEEE Transactions on Power Systems 29 (2014) 2370–2380.

[11] F. Iutzeler, P. Bianchi, P. Ciblat, W. Hachem, Asynchronous distributed optimization using a randomized alternating direction method of multipliers, in: Proceedings of the IEEE Conference on Decision and Control.

[12] A. Abboud, Distributed optimization in large interconnected systems using ADMM, Ph.D. thesis, Université Paris-Saclay, 2016.

[13] R. Zhang, J. T. Kwok, Asynchronous distributed ADMM for consensus optimization, 31st International Conference on Machine Learning, ICML 2014 5 (2014) 3689–3697.

[14] T. H. Chang, M. Hong, W. C. Liao, X. Wang, Asynchronous Distributed ADMM for Large-Scale Optimization - Part I: Algorithm and Convergence Analysis, IEEE Transactions on Signal Processing 64 (2016) 3118–3130.

[15] R. Zhu, D. Niu, Z. Li, A Block-wise, Asynchronous and Distributed ADMM Algorithm for General Form Consensus Optimization, arXiv preprint arXiv:1802.08882 (2018).

[16] J. Zhang, Asynchronous decentralized consensus ADMM for distributed machine learning, in: 2019 International Conference on High Performance Big Data and Intelligent Systems, HPBD and IS 2019, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 22–28.

[17] N. Bastianello, R. Carli, L. Schenato, M. Todescato, Asynchronous distributed optimization over lossy networks via relaxed admm: Stability and linear convergence, IEEE Transactions on Automatic Control (2020).

[18] J. Zhang, S. Nabavi, A. Chakrabortty, Y. Xin, ADMM Optimization Strategies for Wide-Area Oscillation Monitoring in Power Systems under Asynchronous Communication Delays, IEEE Transactions on Smart Grid 7 (2016) 2123–2133.

[19] J. Xu, H. Sun, C. J. Dent, ADMM-based Distributed OPF Problem Meets Stochastic Communication Delay, IEEE Transactions on Smart Grid (2018).

[20] J. Guo, G. Hug, O. Tonguz, Impact of communication delay on asynchronous distributed optimal power flow using ADMM, in: 2017 IEEE International Conference on Smart Grid Communications, SmartGridComm 2017.

[21] A. Huang, S. K. Joo, K. B. Song, J. H. Kim, K. Lee, Asynchronous decentralized method for interconnected electricity markets, International Journal of Electrical Power and Energy Systems 30 (2008) 283–290.

[22] M. H. Ullah, J. D. Park, Distributed Energy Optimization in MAS-based Microgrids using Asynchronous ADMM, in: 2019 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference, ISGT 2019, Institute of Electrical and Electronics Engineers Inc., 2019.

[23] F. Moret, T. Baroche, E. Sorin, P. Pinson, Negotiation algorithms for peer-to-peer electricity markets: Computational properties, in: 20th Power Systems Computation Conference, PSCC 2018, Institute of Electrical and Electronics Engineers Inc., 2018.

[24] J. Guo, G. Hug, O. K. Tonguz, Intelligent Partitioning in Distributed Optimization of Electric Power Systems, IEEE Transactions on Smart Grid 7 (2016) 1249–1258.

[25] T. Baroche, F. Moret, P. Pinson, Prosumer markets: A unified formulation, in: 2019 IEEE Milan PowerTech, PowerTech 2019, Institute of Electrical and Electronics Engineers Inc., 2019.

[26] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd, OSQP: An Operator Splitting Solver for Quadratic Programs, Technical Report, 2020.

[27] R. Le Goff Latimier, T. Baroche, H. Ben Ahmed, Mitigation of communication costs in peer-to-peer electricity markets, 2019 IEEE Milan PowerTech, PowerTech 2019 (2019).

Appendix A. Coefficients of the market agents

The following table contains numerical data of the market agents, including the cost function coefficients as in (A.1) and the agent location used in (A.2).

$$f_i(p_i) = 2 \cdot a_i \cdot p_i^2 + b_i \cdot p_i \tag{A.1}$$

$$D_{ij} = \sqrt{(loc_{x,i} - loc_{x,j})^2 + (loc_{y,i} - loc_{y,j})^2} \tag{A.2}$$

i	type	a_i	b_i	$\underline{p_i}$	$\overline{p_i}$	loc_x, i	loc_y, i	i	type	a_i	b_i	$\underline{p_i}$	$\overline{p_i}$	loc_x, i	loc_y, i
		€/kWh²	€/kWh	kWh	kWh					€/kWh²	€/kWh	kWh	kWh		
1	prod	0.0351	29.0	0	775	0,6626	1,2592	56	cons	0.0266	66.0	-1354	0	0,3644	0,0367
2	prod	0.0369	21.0	0	218	1,7075	0,5162	57	cons	0.0360	56.0	-467	0	1,6157	0,4855
3	prod	0.0367	29.0	0	823	0,1216	0,1727	58	cons	0.0388	55.0	-1475	0	0,0552	1,7017
4	prod	0.0347	38.0	0	1084	1,4693	1,1295	59	cons	0.0288	65.0	-1014	0	1,6899	0,6204
5	prod	0.0468	28.0	0	1081	1,2986	1,0040	60	cons	0.0274	66.0	-686	0	0,0102	1,5741
6	prod	0.0408	22.0	0	531	1,3640	0,5622	61	cons	0.0327	65.0	-164	0	1,7724	1,9118
7	prod	0.0346	29.0	0	261	0,9894	1,4688	62	cons	0.0414	81.0	-1183	0	0,4273	1,0198
8	prod	0.0403	9.0	0	900	0,3764	1,0746	63	cons	0.0245	55.0	-150	0	1,7541	1,1871
9	prod	0.0389	38.0	0	614	0,7723	0,4762	64	cons	0.0399	72.0	-1451	0	1,1051	0,8164
10	prod	0.0276	20.0	0	944	1,5919	1,9355	65	cons	0.0423	67.0	-1117	0	1,0255	1,1618
11	prod	0.0420	37.0	0	862	0,9101	0,5004	66	cons	0.0407	86.0	-753	0	0,0936	0,9535
12	prod	0.0321	31.0	0	562	1,3604	0,7221	67	cons	0.0318	81.0	-1368	0	0,6410	0,3118
13	prod	0.0360	23.0	0	929	1,7322	1,0304	68	cons	0.0248	75.0	-622	0	1,5136	1,0441
14	prod	0.0340	29.0	0	216	0,0253	1,8086	69	cons	0.0217	80.0	-472	0	0,3570	0,9966
15	prod	0.0356	32.0	0	251	0,2816	1,6919	70	cons	0.0297	70.0	-309	0	0,2856	0,7119
16	prod	0.0356	23.0	0	657	0,6980	1,8583	71	cons	0.0271	78.0	-409	0	0,7836	1,9152
17	prod	0.0247	34.0	0	70	0,6941	0,8823	72	cons	0.0375	73.0	-725	0	0,4933	0,8265
18	prod	0.0393	33.0	0	682	0,5370	1,6745	73	cons	0.0357	67.0	-868	0	1,8410	1,4669
19	prod	0.0353	29.0	0	514	0,9865	0,7615	74	cons	0.0341	72.0	-1027	0	0,5387	0,3468
20	prod	0.0395	12.0	0	386	1,9675	1,6190	75	cons	0.0236	60.0	-383	0	0,6470	1,2834
21	prod	0.0319	35.0	0	234	1,6393	1,8427	76	cons	0.0353	82.0	-660	0	0,1992	1,2598
22	prod	0.0314	20.0	0	100	0,6857	0,6969	77	cons	0.0359	76.0	-1420	0	1,1034	1,0929
23	prod	0.0296	37.0	0	445	1,7328	1,7711	78	cons	0.0309	65.0	-643	0	1,7707	1,2576
24	prod	0.0405	21.0	0	777	1,1198	0,8357	79	cons	0.0350	76.0	-1289	0	1,2828	1,0441
25	prod	0.0283	20.0	0	891	1,9156	0,8871	80	cons	0.0246	74.0	-937	0	0,0559	0,7919
26	prod	0.0492	33.0	0	757	1,1352	1,8558	81	cons	0.0341	80.0	-554	0	0,8907	0,5754
27	prod	0.0378	30.0	0	268	0,5235	1,9296	82	cons	0.0289	68.0	-592	0	1,9098	1,0736
28	prod	0.0480	28.0	0	578	0,2445	0,1685	83	cons	0.0410	49.0	-574	0	1,8727	0,5556
29	prod	0.0323	16.0	0	58	1,8959	1,1172	84	cons	0.0380	63.0	-470	0	0,9654	0,5558
30	prod	0.0371	14.0	0	1001	0,4950	1,8816	85	cons	0.0403	83.0	-683	0	1,2576	0,3562
31	cons	0.0390	74.0	-516	0	1,2932	0,1227	86	cons	0.0267	83.0	-10	0	0,1726	0,1066
32	cons	0.0429	66.0	-126	0	1,5768	0,2463	87	cons	0.0343	75.0	-375	0	0,6618	0,2125
33	cons	0.0407	76.0	-1463	0	0,4958	0,0629	88	cons	0.0375	88.0	-239	0	1,7992	0,8815
34	cons	0.0402	65.0	-1479	0	0,3429	1,0859	89	cons	0.0399	65.0	-1372	0	0,3684	0,1629
35	cons	0.0386	77.0	-834	0	0,2645	1,7631	90	cons	0.0393	73.0	-695	0	1,4312	1,6920
36	cons	0.0395	64.0	-765	0	0,0456	1,7721	91	cons	0.0375	53.0	-514	0	0,1660	0,4220
37	cons	0.0361	79.0	-1174	0	0,2507	1,0795	92	cons	0.0301	61.0	-1371	0	1,2704	1,1081
38	cons	0.0467	57.0	-181	0	1,6926	1,3312	93	cons	0.0301	70.0	-299	0	1,7400	0,0339
39	cons	0.0384	61.0	-1160	0	1,3496	1,1711	94	cons	0.0321	73.0	-586	0	1,8077	1,7412
40	cons	0.0241	56.0	-310	0	0,5308	0,8472	95	cons	0.0339	59.0	-1059	0	1,4840	0,6716
41	cons	0.0418	39.0	-1344	0	1,1601	1,2888	96	cons	0.0206	81.0	-293	0	0,6072	1,5895
42	cons	0.0425	70.0	-107	0	1,6866	0,7416	97	cons	0.0345	76.0	-794	0	1,4344	1,6162
43	cons	0.0370	77.0	-623	0	1,8418	1,2733	98	cons	0.0373	52.0	-946	0	0,2828	1,2928
44	cons	0.0330	66.0	-322	0	0,1251	1,1921	99	cons	0.0449	68.0	-522	0	0,6294	0,3967
45	cons	0.0348	54.0	-723	0	1,5502	0,3717	100	cons	0.0403	81.0	-1473	0	1,0658	1,2635
46	cons	0.0295	66.0	-1182	0	0,6214	0,7634	101	cons	0.0240	57.0	-1488	0	1,4196	1,8454
47	cons	0.0243	53.0	-249	0	0,6575	0,8622	102	cons	0.0404	62.0	-491	0	0,4892	0,5767
48	cons	0.0537	73.0	-44	0	0,0781	1,0403	103	cons	0.0428	65.0	-1031	0	0,7602	1,4163
49	cons	0.0355	69.0	-860	0	0,8139	1,3018	104	cons	0.0303	61.0	-121	0	1,2965	0,4389
50	cons	0.0326	69.0	-1394	0	0,4410	0,7359	105	cons	0.0438	69.0	-764	0	1,1921	1,8002
51	cons	0.0382	85.0	-8	0	1,7540	0,3778	106	cons	0.0341	54.0	-1191	0	0,7341	0,0059
52	cons	0.0416	50.0	-1021	0	1,9288	1,2116	107	cons	0.0351	76.0	-1487	0	1,8821	1,7931
53	cons	0.0424	70.0	-99	0	1,2611	1,5065	108	cons	0.0381	61.0	-476	0	1,6693	0,7555
54	cons	0.0402	80.0	-960	0	1,7170	1,3035	109	cons	0.0398	60.0	-1091	0	1,6574	1,8177
55	cons	0.0334	67.0	-117	0	0,1476	0,7530	110	cons	0.0308	76.0	-1035	0	1,4468	0,1260