Algorithmes asynchrones pour la gestion décentralisée des réseaux électriques soumis aux aléas de communication

Alyssia Dong

DOCTORAT BRETAGNE LOIRE MATHSTIC

ens rennes | école normale supérieure

THÈSE DE DOCTORAT DE

L'ÉCOLE NORMALE SUPÉRIEURE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Électronique - Génie Électrique*

Par
Alyssia DONG

Algorithmes asynchrones pour la gestion décentralisée des réseaux électriques soumis aux aléas de communication

Thèse présentée et soutenue à École Normale Supérieure de Rennes, le 18 juillet 2022
Unité de recherche : Laboratoire SATIE

Rapporteurs avant soutenance :

M. Raphaël CAIRE — Maître de Conférences HDR, Grenoble INP - Ense3, Grenoble
M. Marc PETIT — Professeur des Universités, CentraleSupélec, Paris-Saclay

Composition du Jury :

Président :	M. Benoit ROBYNS	Professeur, Junia, Lille
Examinatrice :	Mme Luce BROTCORNE	Directrice de Recherche INRIA, Lille
Dir. de thèse :	M. Hamid BEN AHMED	Maître de Conférences HDR, ENS Rennes, Bruz
Co-enc. de thèse :	M. Roman LE GOFF LATIMIER	PRAG Docteur, ENS Rennes, Bruz

Invité(s) :

Mme Zita VALE — Professeure, ISEP/IPP, Porto

Remerciements

Ce présent manuscrit est le fruit d'un travail de longue haleine qui a été alimenté par de nombreuses personnes que je souhaite remercier sincèrement.

Je tiens à remercier en premier lieu mes encadrants de thèse, qui m'ont confiée ce sujet et m'ont accompagnée depuis de nombreuses années. Merci Hamid pour ta confiance et ta gentillesse, ainsi que ta précieuse expertise que tu as mise à disposition aux travers de réflexions partagées, toujours pertinentes et toujours inspirantes. J'aimerais aussi remercier chaleureusement Roman, qui s'est montré disponible malgré son emploi du temps chargé pour répondre à mes questions aussi bien techniques que générales. Merci pour ton soutien et ton temps.

Je souhaite adresser mes remerciements aux membres de mon jury de thèse, à commencer par M. Benoît Robyns, pour avoir accepté de juger de la qualité de mon travail en tant que président du jury. Je remercie tout particulièrement M. Raphaël Caire et M. Marc Petit pour avoir pris le temps de lire ce manuscrit avec une rigueur scientifique exemplaire, et pour l'avoir amélioré de manière significative au travers de corrections pertinentes. De même, je remercie Mme Luce Brotcorne pour avoir accepté d'être examinatrice de ma thèse, et Mme Zita Vale pour son intérêt porté à mes travaux.

Je suis aussi reconnaissante envers l'ENS Rennes, qui a fourni le financement de cette thèse, ainsi qu'envers les personnels de l'école qui participent activement à son bon fonctionnement et sans lesquels l'école ne serait pas cet endroit où il y fait bon vivre et que j'ai considéré pendant sept années comme ma seconde maison. Je souhaiterais de même remercier les collègues du département mécatronique, avec lesquels j'ai pu exercer dans une moindre mesure le métier d'enseignante, et qui m'ont guidée et inspirée vers le poste que j'occupe aujourd'hui. J'aimerais aussi remercier les doctorants et stagiaires avec lesquels j'ai pu partager ces trois années dans une ambiance de travail exceptionnelle : Claire, Louise, Damien, Anas, Marvin, Pauline, Briac, Thibault, Béatrice, Lucas, Simon, Thomas, Youen, Alexandre, Zinedine... Merci particulièrement à Guénolé, Hassen et Ibrahim avec qui ça a été un réel plaisir d'échanger sur des sujets communs.

J'aimerais remercier ma famille et mes proches qui m'ont soutenue durant ma thèse.

Enfin, je remercie mon chat Nyx, co-auteur malgré lui de ce manuscrit.

Sommaire court

Table des matières

Notations

$\mathcal{C}$	Ensemble des consomacteurs
$\mathcal{C}_i$	Ensemble des consomacteurs partenaires du consomacteur i
$\mathcal{C}_n$	Ensemble des consomacteurs connectés au nœud n
δ	Paramètre d'asynchronisme
f_i	Fonction coût du consomacteur i
i, j	Indices de consomacteurs
k, h	Indices de régions
$\mathcal{L}_n$	Ensemble de liens entre régions liés au nœud n
n, m	Indices de nœuds
$\mathcal{N}$, N	Ensemble des nœuds du réseau et le cardinal associé
$\mathcal{N}_{ref}$	Ensemble des nœuds de référence
$\mathcal{N}_n$, N_n	Ensemble de nœuds voisins du nœud n et le cardinal associé
$\mathcal{O}_k$	Ensemble des nœuds limitrophes de la région k
P_i	Puissance active du consomacteur i
P_{nm}	Puissance active de la ligne qui relie les nœuds n et m
$\underline{P_i}$, $\overline{P_i}$	Limites inférieure et supérieure pour la puissance active
Q_i	Puissance réactive du consomacteur i
Q_{nm}	Puissance réactive de la ligne qui relie les nœuds n et m
$\underline{Q_i}$, $\overline{Q_i}$	Limites inférieure et supérieure pour la puissance réactive
$\mathcal{R}_k$	Ensemble des nœuds de la région k
S_i	Puissance apparente du consomacteur i
S_{nm}	Puissance apparente dans la ligne qui relie les nœuds n et m
$\overline{S_{nm}}$	Limite de puissance dans la ligne qui relie les nœuds n et m
t	Numéro d'itération
V_n	Tension complexe au nœud n
$\underline{V_n}$, $\overline{V_n}$	Limites inférieure et supérieure pour la tension
$\boldsymbol{Y}$, $Y_{n,m}$	Matrice d'admittance et les valeurs associées

Introduction

L'augmentation massive de la part des énergies renouvelables intermittentes et difficilement prévisibles dans le mix électrique entraîne des défis majeurs à surmonter pour le réseau électrique. Une des pistes étudiées pour relever ces défis est la gestion plus "intelligente" du réseau électrique, notamment au travers du paradigme des Smart Grids. Ce paradigme juxtapose le réseau électrique à un réseau de communication et des algorithmes de gestion avancés, et permet ainsi de pallier les limites historiques liées au remplacement des grandes centrales non renouvelables par un grand nombre de petites centrales renouvelables et distribuées. Les Smart Grids introduisent le concept de gestion décentralisée, qui permet de coordonner un très grand nombre d'agents du réseau électrique, ce qui s'avère difficile à réaliser en version centralisée à cause du goulet d'étranglement des communications vers l'agent central. De plus, les algorithmes décentralisés proposent des avantages comparativement aux alternatives centralisées, tels que la confidentialité des données et les préférences hétérogènes, notamment au travers du fait que les agents décentralisés ont le pouvoir de choisir les agents avec lesquels ils échangent. Les algorithmes décentralisés fragmentent un problème global en plusieurs sous-problèmes locaux résolus par des machines dispersées. Ces machines se coordonnent de manière itérative en communicant leur avancement aux machines voisines à chaque itération. Malgré leurs avantages, les algorithmes décentralisés reposent fortement sur le réseau de communication. En effet, le moindre ralentissement de la communication impacte l'avancement global de l'algorithme car chaque agent doit attendre l'intégralité des messages en provenance de ses voisins à chaque itération. La perte définitive d'un message peut même entraîner l'arrêt complet de la résolution.

Dans les travaux décrits dans ce manuscrit, on cherche à quantifier les effets des aléas du réseau de communication sur la résolution des algorithmes décentralisés de gestion du réseau électrique. Une analyse approfondie de l'état de l'art montre que ces effets sont peu étudiés et analysés, notamment dans le cas de configurations pair à pair avec prise en compte des contraintes physiques. Les travaux présentés ici tentent de combler en partie cette insuffisance. On se penche en particulier sur trois types d'algorithmes : un algorithme de marché pair à pair où chaque agent du marché communique exclusivement avec les agents avec lesquels il négocie, un algorithme d'Optimal Power Flow qui cherche à minimiser les coûts des agents tout en prenant en compte les contraintes physiques du réseau électrique, et enfin un algorithme de marché pair à pair endogène qui ajoute au marché pair à pair un agent prenant en compte les contraintes du réseau électrique. On introduit aussi le principe d'algorithme asynchrone, qui permet de limiter les délais d'attente à chaque itération, et on étudie les effets des aléas de réseau de communication sur la résolution des versions asynchrones de ces algorithmes.

Les contributions de ce manuscrit consistent en :

- l'implémentation asynchrone des algorithmes susmentionnés, disponible en open-source sur Gitlab : `https://gitlab.com/alyssiadong/AsyncDecElec.jl`,
- la mise en place d'une plateforme de simulation prenant en compte les communications lors de la résolution d'algorithmes décentralisés,
- les modèles de délais de communication paramétriques permettant de modéliser les variations de délais de communication, qu'elles soient d'ordre stochastique ou bien dues à la topologie du réseau de communication,
- la prise en compte simultanée des délais de communication et des temps de calcul, ainsi que l'étude de leurs impacts sur le temps de convergence global des algorithmes décentralisés,
- le déploiement opérationnel d'un algorithme de marché pair à pair asynchrone sur une plateforme expérimentale.

Le manuscrit s'organise en cinq chapitres et une annexe. On commence par mettre en place le contexte dans lequel s'inscrivent les travaux de thèse et par fournir un état de l'art sur les algorithmes décentralisés asynchrones (Chapitre 1). Après avoir définit des modèles de délais de communication paramétriques, déterministes ou stochastiques, nous présentons l'architecture de la plateforme de simulation nous permettant de réaliser différentes analyses et comparaisons (Chapitre 2). Ensuite, une première étude est menée sur un marché pair à pair utilisant une résolution synchrone, puis asynchrone. Deux types de modèles de délais de communication stochastiques sont analysés, en particulier via l'impact des délais sur le temps de convergence (Chapitre 3). Les contraintes physique du réseau électrique, telles que le plan de tension et les limites de congestion des lignes, ne sont pas prises en compte dans le marché pair à pair. Elles sont alors intégrées dans une deuxième étude portant sur des algorithmes d'Optimal Power Flow. Là aussi, l'impact des aléas de communication sur le temps de convergence est étudié, ainsi que l'impact du partitionnement du problème qui influe fortement sur les résolutions synchrones et asynchrones des algorithmes (Chapitre 4). Nous cherchons ensuite à revenir sur la topologie d'échanges pair à pair de la première étude, tout en y incorporant un agent à part entière s'occupant des contraintes physiques du réseau : le marché pair à pair endogène. Ce marché permet de séparer les aspects économiques des aspects physiques du réseau. Des études sont réalisées sur l'impact conjoint des aléas de communication et des temps de calcul sur le temps de convergence de ce marché (Chapitre 5). Enfin, l'Annexe décrit les différents cas test sur lesquels nous testons les algorithmes synchrones et asynchrones présentés.

Chapitre 1

Présentation du problème et état de l'art

Sommaire

1.1 Contexte : les réseaux électriques

1.1.1 Les besoins énergétiques et leur évolution future

Les besoins en énergie ont augmenté significativement durant l'ère d'industrialisation, et se stabilisent aujourd'hui. Malgré cette stabilisation, nos besoins mondiaux annuels d'énergie se chiffrent en centaines d'exajoules (EJ). En 2019, la consommation d'énergie primaire s'élevait à 581.51 EJ [1], soit 161 388 TWh, pour une consommation finale de 418 EJ, soit 116 111 TWh, dont 19.7% sous forme d'énergie électrique [2].

Il est nécessaire de pouvoir ancrer la production d'énergie dans une démarche de développement durable, d'autant plus dans le contexte d'urgence climatique dans lequel nous nous trouvons. Les 17 objectifs de développement durable des Nations Unies [3] sont présentés en Figure 1.1. Les changements sur le réseau électrique peuvent s'inscrire dans les objectifs 7, 12 et 13.

— **Objectifs 7 et 13 : énergie propre et d'un coût abordable, mesures relatives à la lutte contre les changements climatiques**. L'électricité est un vecteur d'énergie qui a le potentiel d'atteindre la neutralité carbone, au travers notamment de la production nucléaire et des énergies renouvelables. Atteindre la neutralité carbone permettrait de réduire le réchauffement climatique futur, d'où le terme d'*énergie propre*.

— **Objectif 12 : Consommation et production responsables**. La production d'électricité répond à des besoins en énergie déterminés par les consommateurs. Il est important de questionner ces besoins en énergie, et de se demander s'il est possible de réduire

1. https://www.bp.com/content/dam/bp/business-sites/en/global/corporate/pdfs/energy-economics/statistical-review/bp-stats-review-2021-primary-energy.pdf
2. https://www.iea.org/reports/key-world-energy-statistics-2021
3. https://www.un.org/sustainabledevelopment/sustainable-development-goals/

les consommations d'une part au travers de l'efficacité des infrastructures et des équipements, pour en réduire les pertes, et d'autre part en adoptant des comportements de sobriété énergétique. La gestion intelligente de l'énergie permet aussi de privilégier des sources renouvelables plutôt que des sources carbonées, et mène ainsi à une consommation plus responsable.

FIGURE 1.1 – Les 17 objectifs du développement durable selon les Nations Unies.

Évolution future du mix énergétique On dit que l'on atteint la neutralité carbone lorsque l'émission de CO2 due à l'activité humaine ne dépasse pas ce que les puits de carbone peuvent absorber. Cet objectif de neutralité carbone permet de réduire les conséquences climatiques des émissions de gaz à effet de serre. En France, l'objectif de neutralité carbone d'ici 2050 est inscrit dans la Stratégie Nationale Bas-Carbone [4] (SNBC) depuis 2015 et dans le projet de loi énergie climat [5] depuis 2019.

Plusieurs organismes proposent des scénarios qui permettent de réduire les émissions de gaz à effet de serre tout en répondant à nos besoins énergétiques. On choisit ici de détailler les scénarios de RTE et négaWatt comme deux exemples de stratégies distinctes visant toutes deux la neutralité carbone.

L'association négaWatt propose un scénario [6] se reposant principalement sur les énergies renouvelables et l'arrêt de la production nucléaire d'ici 2050, comme l'illustre la Figure 1.2 représentant la prévision de la consommation d'énergie primaire en France selon leur scénario. La croissance des parts de renouvelable dans le mix énergétique français est atteinte :

— grâce à un développement territorial du photovoltaïque pour atteindre 58 TWh de production en 2030, notamment en misant sur l'installation de panneaux par des acteurs non-professionnels de l'énergie comme les collectivités locales, les PMEs, les agriculteurs, les particuliers et les groupes citoyens ;

4. https://www.ecologie.gouv.fr/strategie-nationale-bas-carbone-snbc
5. https://www.ecologie.gouv.fr/sites/default/files/19060_ProjetloiENERGIE-CLIMAT_BAT.pdf
6. https://www.negawatt.org/IMG/pdf/scenario-negawatt-2022-rapport-complet-partie3.pdf

— au travers d'un programme d'accélération du développement de l'éolien pour atteindre 114 TWh de production en 2030, en misant sur des éoliennes terrestres mieux réparties entre les régions ainsi que sur des éoliennes maritimes installées près des côtes ;
— en accompagnant la sortie du nucléaire, c'est-à-dire en ne construisant plus de nouveaux réacteurs et en utilisant ceux déjà en place jusqu'à leur fin de vie, ce qui correspond à une production de 244 TWh en 2030 soit 50% du mix électrique ;

La croissance des énergies renouvelables se fait en parallèle d'une diminution des besoins énergétiques, notamment au travers du concept de sobriété énergétique et d'augmentation de l'efficacité des équipements énergivores. Une des stratégies proposées par l'association négaWatt pour diminuer les besoins en énergie repose sur un grand programme de rénovation de la performance des bâtiments.

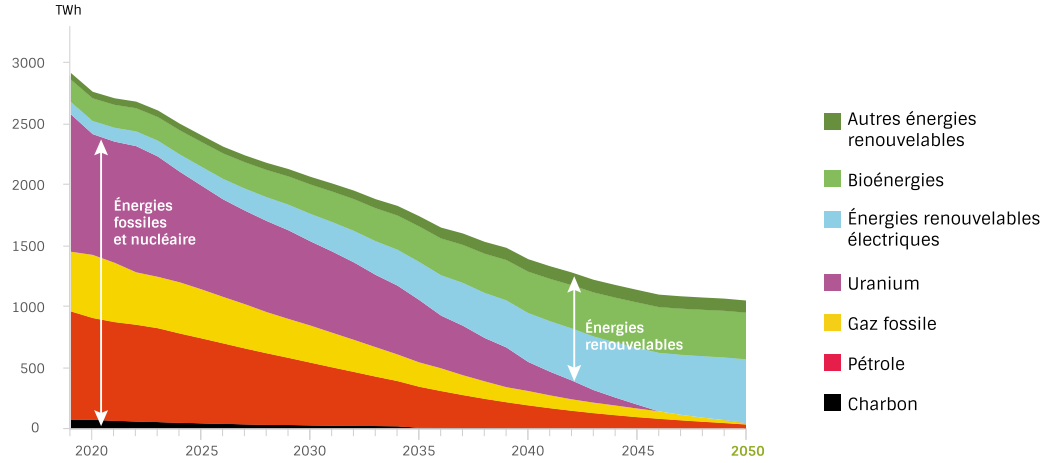


FIGURE 1.2 – Prévision de la consommation d'énergie primaire en France d'ici à 2050 selon le scénario négaWatt 2022. À noter que l'énergie primaire est différente de l'énergie consommée/énergie finale. Elle englobe l'énergie finale ainsi que les pertes de conversion de l'énergie.

Le gestionnaire de réseau électrique RTE propose quant à lui dans son rapport sur les futurs énergétiques 2050 [1] une diminution de la consommation globale d'énergie finale, passant de 1600 TWh d'énergie consommée aujourd'hui à 930 TWh en 2050, accompagné d'une électrification importante (de 25 à 55%) ainsi qu'une augmentation de la part des énergies renouvelables, comme l'illustre la Figure 1.3. Selon les scénarios RTE : le mix de production repose soit à parts égales entre le renouvelable et le nucléaire, ce qui représente 70 GW de capacité solaire installées en 2050, soit totalement sur des énergies renouvelables, ce qui représente 208 GW de capacité solaire installée.

Quelques soient les scénarios considérés, la part des renouvelables est amenée à augmenter significativement, passant de 22% du mix électrique en 2021 à au moins 50% en 2050.

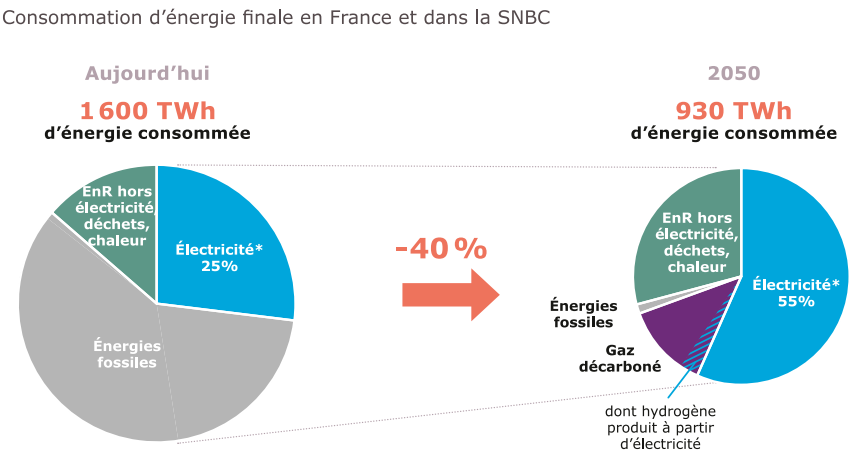


FIGURE 1.3 – Évolution de la production d'électricité d'ici à 2050 selon le rapport 2022 de RTE sur les futurs énergétiques [1].

1.1.2 Principes fondamentaux du réseau électrique

L'électricité est un bien impossible à stocker tel quel, donc qui nécessite sa consommation immédiate une fois produite. Les échanges d'électricité entre les producteurs et les consommateurs nécessitent l'utilisation d'un réseau électrique reliant les producteurs et les consommateurs. Le réseau électrique est un système complexe où de nombreux acteurs interagissent :

- ils utilisent le réseau comme un service pour transporter l'électricité d'un point à un autre,
- ils servent de support au réseau électrique pour s'assurer de la qualité de l'énergie transmise ou pour s'assurer de la bonne santé du réseau, on parle alors de *services système*.

Le plus important pour que le réseau continue de fonctionner nominalement est d'assurer l'équilibre entre les puissances injectées sur le réseau avec les puissances soutirées du réseau. Un déséquilibre trop important et prolongé entraînerait l'effondrement du réseau électrique. Le paradigme historique est de posséder une capacité de production supérieure ou égale à la demande maximale de consommation, qui soit parfaitement pilotable, pour assurer l'équilibre production/consommation. Cependant, plusieurs problèmes subsistent.

- Certaines répartitions de la puissance entre les producteurs ne permettent pas de se conformer aux contraintes physiques du réseau, notamment à cause des limites de puissance des lignes [7].
- La puissance maximale demandée est très élevée par rapport à la valeur moyenne de la puissance consommée dans des conditions normales, ce qui pousse à fortement sur-dimensionner le parc électrique installé. En France, cela est dû aux pics de consommations causés par la thermosensibilité de la production électrique (+ 2400 MW par degrés perdus en hiver à 19h [8]).

7. D'autres contraintes physiques telles que l'angle et l'amplitude des tensions doivent être prises en compte. Des éléments de support du réseau se chargent en partie de résoudre ce problème.

8. `https://bilan-electrique-2020.rte-france.com/consommation-sensibilite-a-la-temperature-et-aux-usages/`

— La production électrique n'est pas systématiquement pilotable en fonction de la source primaire de l'énergie. Les énergies renouvelables fluctuantes, en plus de présenter des incertitudes de puissance, sont intermittentes car elles dépendent de la météo. Elles ne sont donc pilotables qu'en capacité négative seulement. L'énergie nucléaire est peu pilotable car elle possède un temps de réponse très lent et n'est donc pas adaptée pour la régulation en temps réel. Seules les centrales à sources non renouvelables et les stations de pompage-turbinage apparaissent comme facilement pilotables. Ces dernières étant des stockages énergétiques, elles ne peuvent pas être utilisées à volonté.

Les solutions implémentées aujourd'hui pour permettre le bon fonctionnement du réseau électrique sont les suivantes.

— Le réseau électrique installé est bien surdimensionné par rapport à la demande moyenne, mais il existe tout de même des problèmes de surcharge de ligne qu'il faut anticiper pour éviter le blackout. Ces surcharges surviennent notamment au niveau des lignes d'interconnexion entre zones régionales.
— La planification des besoins en énergie et l'investissement dans les infrastructures appropriées se font des années à l'avance, sur des scénarios prospectifs qui anticipent les besoins futurs en énergie.
— Des prévisions des besoins en électricité sur du plus court terme se manifestent sous la forme de contrats sur l'année négociés dans des *marchés à terme* ou *forward markets*, ou bien sur des échanges plus proches de la livraison comme le jour précédent ou le jour même jusqu'à 15 minutes avant au travers de *marchés au comptant* ou *spot markets*.
— La régulation temps réel permet de préserver l'équilibre sur le réseau malgré les écarts avec les prévisions faites auparavant. Elle repose sur un *marché des capacités* où les acteurs du réseau électrique vendent de la capacité d'effacement ou de production au gestionnaire de réseau pour participer à la régulation temps réel.

Bref historique du réseau électrique Au XXe siècle, le réseau électrique et ses infrastructures sont assurées par un monopole, qui s'occupe de la production, du transport, de la distribution, de l'interconnexion de tous les consommateurs industriels ou domestiques. En France, c'est l'État via EDF qui s'occupe de la planification et du développement du réseau électrique, c'est le responsable de la bonne santé du réseau, il prévoit aussi les consommations sur le court terme et décide de la répartition de la production, il commande ses centrales en prévision de la demande, c'est lui qui propose aux consommateurs un forfait d'électricité [9].

À la fin du XXe siècle, la dérégulation du marché électrique ouvre la production et la vente d'électricité à la concurrence et participe alors à la séparation des différents organismes du réseau électrique : on distingue maintenant les producteurs des gestionnaires de réseaux de transport et de distribution, ainsi que des fournisseurs d'électricité et des responsables d'équilibre. Ces acteurs, représentés en Figure 1.4, jouant des rôles distincts doivent se coordonner pour pouvoir assurer la fonction principale du réseau électrique et son bon fonctionnement.

9. On parle de structure verticale intégrée, où les consommateurs n'avaient pas le choix de leur fournisseur d'énergie.

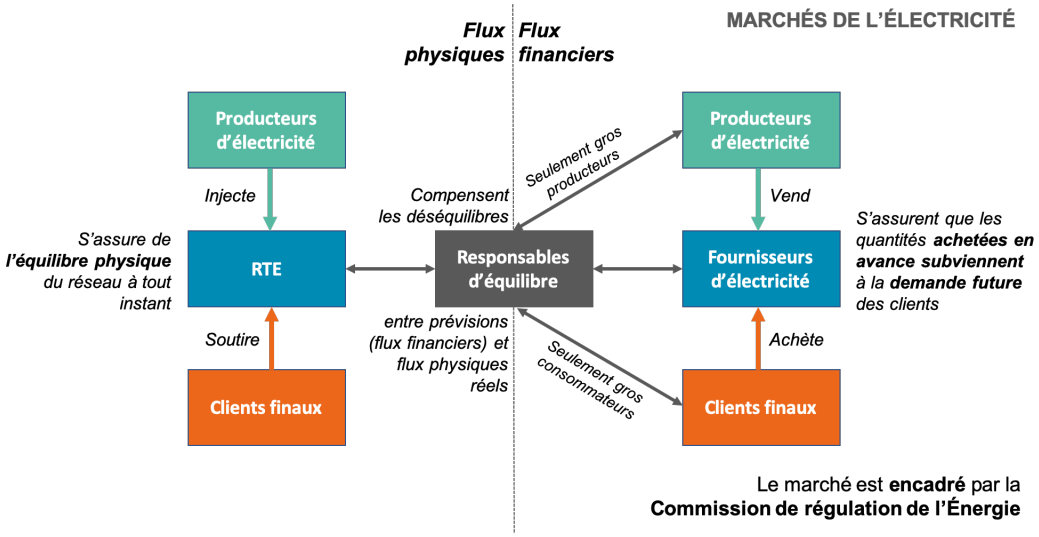


FIGURE 1.4 – Figure modifiée provenant de `https://reseaux.photovoltaique.info/fr/comprendre-le-systeme-electrique/fonctionnement-des-marches-de-lelectricite/acteurs-des-marches-de-lelectricite/`. Représentation des acteurs du réseau électrique, distinguant les flux physiques et les flux financiers, et leurs interactions. Les responsables d'équilibre se trouvent à la frontière entre les flux physiques et les flux financiers en ce qu'ils sont responsables financièrement des écarts entre la puissance injectée et la puissance soutirée du réseau électrique.

Le gestionnaire du **réseau de transport** (GRT) est chargé d'assurer les contraintes physique du réseau lors de son utilisation. Il intervient sur le marché de gros de l'électricité, ainsi que le marché d'ajustement et de capacités. Il doit aussi garantir l'accès non discriminatoire à tous les producteurs souhaitant participer à la production électrique globale. En France, c'est l'entreprise RTE à qui revient le rôle de GRT. Les lignes du réseau de transport présentent des interconnexions importantes. Les centrales de grande puissance sont rattachées au réseau de transport.

Les infrastructures du **réseau de distribution** appartiennent au syndicat départemental d'énergie (SDE ou toute autre entité coïncidente), qui délègue souvent la gestion du réseau de distribution à un gestionnaire tierce. Le gestionnaire du réseau de distribution (GRD) a un but similaire au GRT au niveau du réseau de distribution, mais ce réseau transporte des quantités plus faibles d'électricité, étant donné qu'il est plus proche des consommateurs, industriels et domestiques, ainsi que l'acheminement de l'électricité des postes de transformation jusqu'aux consommateurs. Le GRD est acheteur dans le marché de gros et de capacités pour compenser les pertes. En France, c'est l'entreprise Enedis à qui revient le rôle de GRD dans la plupart des régions.

L'utilisation et la maintenance du réseau électrique entraînent des coûts qu'il faut ensuite répercuter sur les producteurs, consommateurs et utilisateurs du réseau en général.

La **production électrique** reste aujourd'hui encore très centralisée, c'est-à-dire provenant d'un petit nombre de grandes centrales, reliées au réseau de transport et dimensionnées pour répondre aux besoins en énergie nationaux. En 2020 sur le territoire de la France métropolitaine, ce sont 18 centrales nucléaires [10] et 24 centrales thermiques [11] (à charbon ou

10. `https://fr.wikipedia.org/wiki/Centrale_nucléaire_en_France`
11. `https://www.edf.fr/groupe-edf/espaces-dedies/l-energie-de-a-a-z/tout-sur-l-energie/produire-de-l-electricite/le-thermique-a-flamme-en-chiffres`

à gaz) qui représentent 59% du parc électrique installé et 74.6% de la production électrique [12].

Les **fournisseurs d'électricité** sont des organismes intermédiaires qui achètent aux producteurs et revendent aux consommateurs, en ajoutant des services comme l'accès à un service client, des prix fixes sur une période donnée, une atténuation des risques liés à la variation des prix du marché, ou bien l'achat d'électricité provenant de sources renouvelables. Les fournisseurs d'électricité ainsi que les gros consommateurs comme les sites industriels de grande taille participent au marché de gros, de détail, d'ajustement et de capacités. Les autres clients (résidentiels, tertiaires ou petits industriels) interviennent principalement sur le marché de détail.

Enfin, les **responsables d'équilibre** sont responsables des écarts entre la puissance injectée et la puissance soutirée du réseau électrique. Comme illustré en Figure 1.4, ils se trouvent à la limite entre les flux physiques et les flux financiers du réseau électrique car ils doivent compenser les déséquilibres. Un fournisseur d'électricité peut être responsable d'équilibre, mais pas nécessairement. De même, un responsable d'équilibre ne possède pas forcément des moyens de production, il s'arrange pour acheter de la puissance ou de la flexibilité lorsque cela est nécessaire pour assurer l'équilibre.

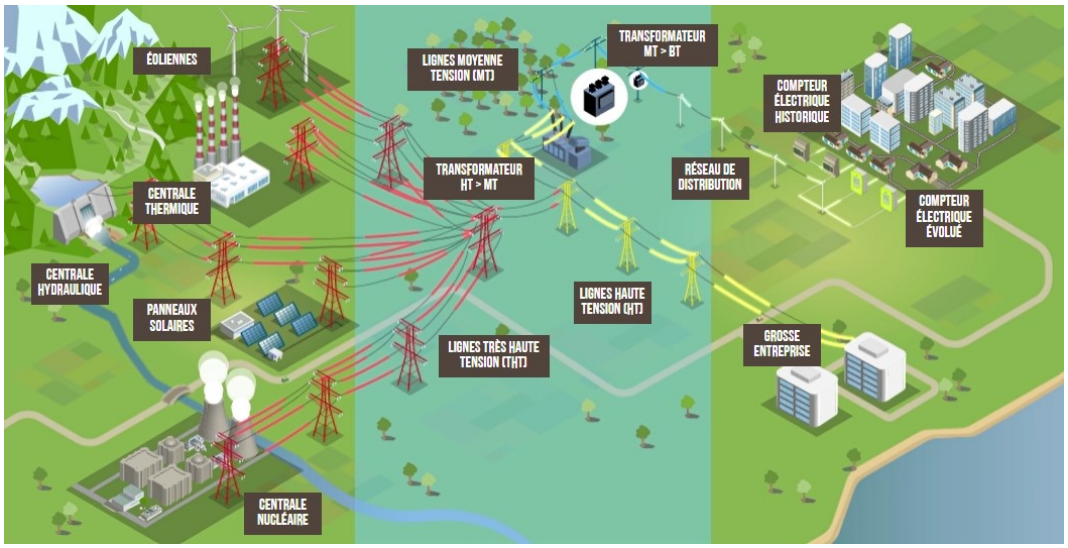


FIGURE 1.5 – Schéma représentant les divers acteurs du réseau électrique, tiré du module pédagogique de la commission de régulation de l'énergie (`http://modules-pedagogiques.cre.fr/m1/`). À gauche, la production d'énergie électrique se fait principalement par de gros producteurs, au milieu les réseaux de transport et de distribution acheminent l'énergie électrique, à droite les consommateurs soutirent de la puissance du réseau électrique.

Les infrastructures du réseau électrique d'aujourd'hui La Figure 1.5 présente les différents éléments constitutifs du réseau électrique. Le réseau de transport est dimensionné pour acheminer de grands volumes d'électricité sur de longues distances. L'énergie est transportée via des lignes à haute et très haute tension, allant de 63 à 400kV. C'est un réseau maillé, possédant de la redondance, ce qui permet d'assurer l'acheminement malgré la panne d'une ou plusieurs lignes. Le réseau de distribution quant à lui est dimensionné pour distribuer l'électricité du réseau de transport en amont directement vers les consommateurs. Les lignes

12. `https://bilan-electrique-2020.rte-france.com/production-production-totale/`

du réseau de distribution sont des lignes moyenne ou basse tension, allant de 230V à 20kV. Ils possèdent une topologie arborescente, avec peu de redondance. Des transformateurs servent à abaisser la tension à mesure que l'on se rapproche des consommateurs.

Des équipements de téléconduite [13], ou *SCADA* pour *Supervisory Control And Data Acquisition*, sont dispersés au sein du réseau électrique et permettent de surveiller et piloter l'ensemble des équipements du réseau. Les SCADAs sont composés de matériels de télécommunications et de fonctions destinées à l'acquisition, au traitement, à la transmission et à la visualisation des informations inhérentes au réseau électrique comme le niveau de tension en un point donné ou bien la puissance traversant une ligne. Ils permettent aussi de contrôler à distance les disjoncteurs pour assurer le bon fonctionnement du réseau électrique. De plus, en France, des compteurs intelligent *Linky* ont été installés ces dernières années au plus proche des consommateurs pour permettre un suivi temporel jusqu'à la demi-heure près de leurs consommations.

La topologie du réseau électrique d'aujourd'hui est verticale, où un petit nombre de grandes centrales de production électrique se placent en amont du réseau de transport, et un grand nombre de petits consommateurs sont reliés en aval du réseau de distribution.

Interactions entre le gestionnaire de marché et le gestionnaire de réseau Depuis la dérégulation de marché, les producteurs et les fournisseurs d'électricité se sont multipliés et entrent en concurrence les uns avec les autres. La concurrence pousse les entreprises à améliorer les services proposés, à diminuer les prix de l'électricité et à proposer un éventail de choix large aux consommateurs. La séparation des rôles des acteurs de l'énergie électrique a imposé le découplage du marché de l'électricité et de la gestion du réseau électrique. Les producteurs et fournisseurs d'électricité signent des contrats en amont de la livraison d'électricité s'engageant à fournir/consommer une certaine quantité de puissance au travers des différents marchés mentionnés plus haut, coordonnés par des gestionnaires de marché [2]. Ils se doivent de communiquer ces contrats au gestionnaire de réseau qui vérifie que les contraintes physiques du réseau sont respectées avec le plan d'injection imposé par le marché. S'il se rend compte de limitations dues à la capacité des lignes, il influe sur les échanges des acteurs concernés pour que la solution finale soit physiquement possible. Il est impératif que les contraintes d'équilibre soient satisfaites, c'est pour cela que face aux incertitudes de consommation et de productions non contrôlables (*non dispatchable*), le gestionnaire de réseau fait en sorte de sécuriser des réserves de productions qu'il pourra utiliser dans le but de maintenir le réseau dans ses limites fonctionnelles.

L'utilisation du réseau électrique entraîne systématiquement des pertes dans les lignes qui ne sont pas prises en compte au niveau des échanges du marché. Le gestionnaire de réseau doit estimer ces pertes et les racheter auprès des producteurs d'électricité. Il fait aussi en sorte de répercuter leurs coûts sur les utilisateurs du réseau électrique.

Marchés de l'électricité Les différents marchés de l'électricité sont présentés dans l'infographie de la Figure 1.6, provenant du site du gestionnaire de réseaux RTE [14]. Elle illustre le placement dans le temps des différents marchés de l'électricité par rapport à l'instant de livraison. Les échanges long terme permettent de sécuriser de grands volumes d'électricité à un prix fixe dans des marchés à terme. En Europe, ces échanges se font sur le marché EEX (European Energy Exchange) Power Derivatives. Les échanges court terme permettent

13. `https://fr.wikipedia.org/wiki/T%C3%A9l%C3%A9conduite_d%27un_r%C3%A9seau_%C3%A9lectrique`
14. `https://www.rte-france.com/chaque-seconde-courant-passe/concevoir-et-mettre-en-oeuvre-des-mecanismes-de-marche-innovants-pour-le-systeme-electrique`

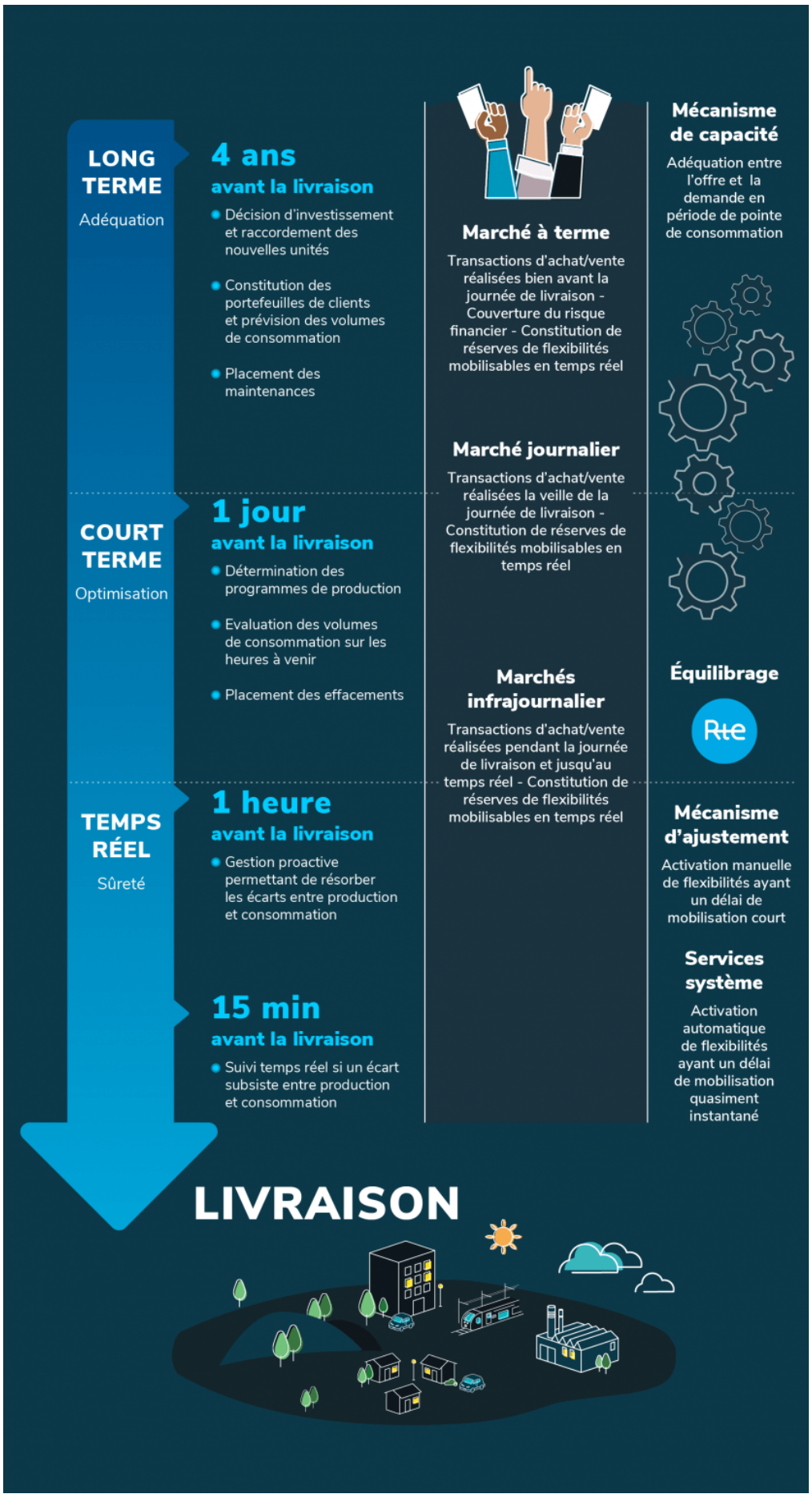


FIGURE 1.6 – Mécanismes de marché décrits par RTE.

d'ajuster les volumes d'électricité à mesure que les incertitudes diminuent. En Europe, ces échanges se font au travers de la bourse européenne de l'électricité EPEX SPOT. Les mécanismes d'équilibrages sont activés en temps réel et gérés par le gestionnaire de réseau. Ils reposent sur des capacités de production ou d'effacement qui ont été négociés au préalable dans un marché des capacités.

1.2 Du réseau vers le Smartgrid

1.2.1 Vers une décentralisation de la gestion du réseau électrique

Comme explicité plus haut, la part de renouvelable dans le mix énergétique est amenée à croître de manière importante dans les années qui suivent, ce qui entraîne des difficultés qu'il faut aborder.

— La nature hautement distribuée des productions renouvelables amène à un changement de paradigme de la production électrique : d'une production principalement centralisée, avec un petit nombre de grandes centrales, à une production distribuée sur tout le territoire avec un grand nombre de petites centrales connectées au réseau de distribution. Les flux d'énergie sont alors amenés à s'inverser par rapport au fonctionnement historique du réseau électrique.

— Les centrales renouvelables sont souvent intermittentes donc ne peuvent pas répondre telles quelles aux exigences de consommation électrique. Il faut les coupler à des éléments ajoutant de la flexibilité comme des batteries, ou bien les coordonner ensemble au travers des centrales virtuelles (*virtual power plant* [3]) de manière à réduire l'incertitude globale.

— La diminution du nombre de grandes centrales mène à une diminution de l'inertie du réseau électrique. En effet, les rotors des alternateurs des grandes centrales tournent à une fréquence proportionnelle à la fréquence du réseau électrique. Ces rotors emmagasinent de l'énergie cinétique $\frac{1}{2}J\Omega^2$, avec J désignant l'inertie mécanique de l'ensemble rotor, arbre et turbine et Ω sa vitesse de rotation. C'est cette énergie que l'on appelle réserve primaire et qui sert de "tampon" lorsque l'équilibre de puissance sur le réseau n'est pas parfaitement atteint. Par exemple, lorsqu'il y a plus de puissance soutirée du réseau que de puissance injectée, alors les rotors ralentissent pour injecter la différence de puissance nécessaire. Par conséquent, la fréquence du réseau électrique diminue jusqu'à ce que le déséquilibre de puissance se résorbe. Or la fréquence du réseau électrique doit rester dans un intervalle donné (autour de 50 ± 0.2 Hz en Europe continentale), ce qui rend cette réserve de puissance primaire très limitée. Le remplacement des grandes centrales par un grand nombre de petites centrales renouvelables [15] aurait pour conséquence de diminuer l'inertie du réseau, donc de diminuer la réserve primaire. Ainsi, l'écart de l'équilibre de puissance permis sur le réseau est amené à diminuer, ce qui engage la stabilité du réseau électrique.

Le principe d'un réseau possédant une topologie décentralisée différente de la topologie verticale actuelle est avancé sous le terme de *Smart Grid*. La commission de régulation de l'énergie en donne la définition suivante [16].

> On désigne par *Smart grid* un réseau d'énergie qui intègre des technologies de l'information et de la communication, ce qui concourt à une amélioration de

15. La plupart des centrales renouvelables ne possèdent pas de parties mobiles connectées au réseau électrique étant donné qu'elles sont interfacées au travers de modules d'électronique de puissance.

16. `https://www.smartgrids-cre.fr/introduction-aux-smart-grids`

> son exploitation et de son dimensionnement et au développement de nouveaux usages tels que l'autoconsommation, le véhicule électrique ou le stockage. Désormais, à la couche physique pour le transit d'énergie des réseaux vient se superposer une couche numérique qui joue un rôle de plus en plus important pour son pilotage.

Ainsi le concept de Smart Grid propose un pilotage du réseau électrique utilisant une gestion plus intelligente permise par la communication et les échanges d'information. Les solutions que permettent les Smart Grids sont les suivantes.

- La consommation plus raisonnée de l'énergie : l'implication des consommateurs dans la gestion de leurs consommations les amènerait à consommer de manière plus raisonnée et à entamer une démarche de sobriété énergétique. Les consommateurs seraient alors amenés à participer aux marchés de l'électricité, notamment s'ils peuvent avoir le choix des producteurs à qui ils achètent leur électricité. Une restructuration des marché serait alors nécessaire pour répondre à la demande des consommateurs et à l'évolution des moyens de production d'électricité [4]. Les consommateurs peuvent aussi faire le choix de mettre leurs flexibilités à disposition d'un agrégateur de flexibilités/nouvel opérateur d'ajustement pour aider à supporter le réseau électrique [5].
- Les divers éléments de flexibilité sont accompagnés d'algorithmes de gestion poussés permettant d'optimiser les échanges d'énergie. Parmi ces éléments, le stockage de l'énergie rendu possible grâce aux batteries, aux centrales de pompage-turbinage, hydrogène, ... permet de stocker de l'énergie lorsqu'elle est produite en surplus et de l'utiliser lorsqu'elle est en déficit. Il faut alors prendre en compte les équations dynamiques qui régissent le comportement de ces stockages, leur rendement et de leur auto-décharge. De plus, la gestion optimale de ce genre de flexibilité nécessite de pouvoir anticiper les besoins futurs en puissance [6]. L'utilisation d'algorithmes poussés prenant en compte des prévisions de production et de consommation permettent d'optimiser ces flexibilités. Le chargement des véhicules électriques peut être vu comme des charges flexibles, voir même peuvent soutenir le réseau en se déchargeant temporairement.
- La prise en compte des besoins multiples tels que les besoins en chauffage qui se présentent comme une autre source de flexibilité : utiliser l'inertie thermique pour le chauffage des habitations, c'est-à-dire chauffer lorsque la production électrique est en surplus et couper le chauffage lorsqu'elle est en déficit, de manière à ce que la température ne soit pas trop impactée.
- Le délestage des charges non essentielles/reprogrammables permet de servir de support de dernière minute du réseau électrique lorsque ce dernier se retrouve en déficit important.
- Enfin, la gestion coordonnée de l'intégralité des acteurs du réseau électrique en utilisant le réseau de communication.

Le challenge du réseau électrique consiste en la transition vers une production zéro carbone tout en continuant de fournir un accès universel et sécurisé à une électricité abordable. Deux technologies permettent de parvenir à une production zéro carbone : les énergies renouvelables distribuées ainsi que la gestion basée sur la communication du côté des consommateurs, notamment des éléments de stockage et des charges flexibles [7]. Dans le paradigme des Smart Grids, les ententes en avance sont d'autant plus cruciales que l'équilibre du réseau est plus fragile. Les mécanismes de marché day-ahead et spot peuvent être décentralisés, pour mieux correspondre à la nouvelle topologie des Smart Grids. Cela nécessite cependant un besoin en communication important, ce qui pourrait freiner le développement des Smart Grids à cause des ralentissements dus aux aléas de communication.

Quelles sont les conséquences des aléas de communication sur la gestion du réseau électrique ? Quelles techniques peut-on utiliser pour en atténuer les conséquences ?

1.2.2 Le réseau électrique cyber-physique et l'importance du réseau de communication

Comme mentionné plus haut, la forte pénétration des énergies renouvelables change fortement l'utilisation du réseau électrique : les échanges de puissance deviennent bi-directionnels et les injections des producteurs distribués deviennent aussi très volatiles, ce qui pousse le réseau électrique à opérer près de ses limites physiques. En plus du grand nombre de producteurs distribués à prendre en compte, beaucoup de charges sont amenées à être connectées au réseau de communication pour que leurs consommations soit coordonnées, de même avec les véhicules électriques dont la charge flexible peut aider à soutenir le réseau électrique. La coordination du réseau électrique est nécessaire pour augmenter l'efficacité, la fiabilité et la sécurité du réseau. Cette coordination repose sur le réseau de communication, il est alors nécessaire que ce dernier soit robuste et performant.

Du point de vue du réseau de transport, dont la stabilité repose sur des intervalles de tension et de fréquence donnés, le moindre retard de communication de l'information pourrait avoir des conséquences graves sur le fonctionnement du réseau, telles que des pannes électriques en cascade ou bien une coupure de l'intégralité du réseau [8]. Des systèmes de gestion de l'énergie dispersés au sein du réseau peuvent servir à des applications de monitoring, d'analyse dynamique à grande échelle, d'estimation d'état du réseau, d'analyse temps-réel des petites perturbations, ou même de répartition économique (*economic dispatch*) [9]. Notamment, le contrôle de la fréquence et du plan de tension du réseau peut être réalisé au travers d'un système multi-agents communicant via un réseau de communication global [10]. Il est nécessaire de garantir une bonne qualité de service du réseau de communication pour assurer le bon fonctionnement du réseau de transport. Pour cela, [11] propose une architecture basée sur un réseau de fibres optiques [17] qui forment l'épine dorsale du réseau de communication en connectant les sous-stations entre elles, là où les éléments de mesure se connectent aux sous-stations via des réseaux locaux (LAN).

Quant au réseau de distribution, sur lequel sont raccordés de nombreux compteurs intelligents, l'importance porte sur la haute couverture du réseau de communication pour permettre à tous les éléments gérant les charges flexibles et les producteurs distribués de se coordonner entre eux. L'utilisation d'un réseau 5G comme moyen physique de communication est en discussion dans la littérature [8]. Notamment, la possibilité de répartir ce réseau 5G en plusieurs sous-réseaux destinés à des applications différentes (*network slicing* [18]) permet de garantir une qualité de service à définir en fonction des besoins de l'application, comme par exemple une garantie de délai de transmission ou de débit, peu importe la charge du réseau de communication par ailleurs.

Étant donné le grand nombre d'équipements amenés à échanger sur le réseau de communication, il est intéressant de limiter le volume des données à envoyer au travers du *edge computing* [19], en traitant les données à la périphérie du réseau, près de la source des données, pour éviter la transmission d'un grand nombre de données non pertinentes. Cette notion de *edge computing* est cohérente avec des approches de gestion décentralisées et des approches

17. Ces fibres optiques sont installées avec les infrastructures électriques, on parle de câble de garde à fibre optique, et servent aussi de raccordement à la terre. `https://fr.wikipedia.org/wiki/OPGW`
18. `https://en.wikipedia.org/wiki/5G_network_slicing`
19. `https://fr.wikipedia.org/wiki/Edge_computing`

de systèmes multi-agents. La réduction des échanges se fait aussi en gérant intelligemment la fréquence des communications [12].

La co-simulation du réseau électrique et du réseau de communication est utilisée pour étudier l'influence de ce dernier sur le bon fonctionnement du réseau électrique au travers de prismes tels que, entre autres, la cyber-sécurité [13, 14] et la résilience face aux attaques informatiques [15], le placement optimal d'éléments de mesure communicants [9], l'influence de la qualité de service sur le contrôle en tension d'un nœud [16], la gestion des défaillances en cascade [17]. Les incertitudes du système de communication sont évaluées dans le cadre des Smart Grids dans [18] : la défaillance d'éléments de l'infrastructure de communication, le taux d'erreurs de transmission, les délais de transmission, les erreurs de routage et la perte d'un message y sont considérés. Ces aléas de communication peuvent aussi être émulés en parallèle d'un simulateur temps réel sur une plateforme constituée d'ordinateurs communicants pour valider des algorithmes de gestion décentralisés [19].

1.2.3 Marchés de l'électricité et production distribuée

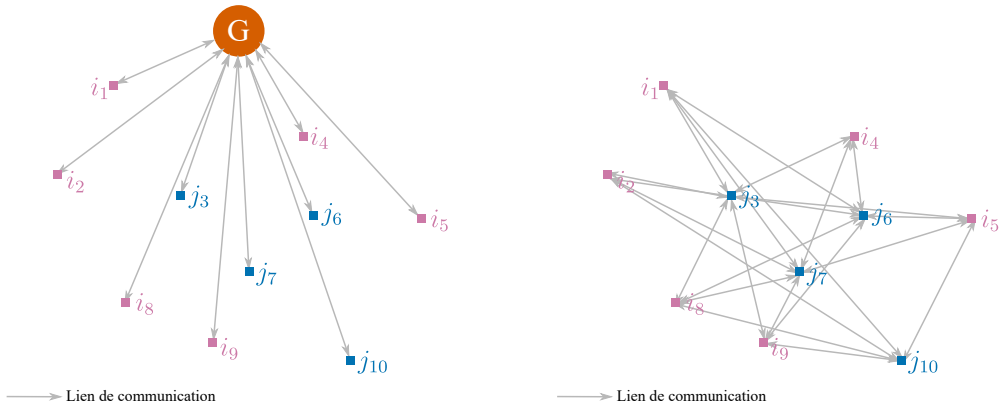
Deux structures de marché sont possibles : la structure centralisée et la structure décentralisée [20, 21]. La première requiert la présence d'un gestionnaire de marché pour rassembler les offres et les demandes d'électricité et en déduire le prix du marché. La seconde opère sans gestionnaire de marché central et laisse les producteurs et consommateurs réaliser des échanges entre eux, on parle alors de marché pair à pair.

Les énergies distribuées ont plusieurs défis à surmonter : leur petites tailles ne leur permettant pas de participer de manière significative dans les marchés de gros car les prix de participation sont trop élevés et ne leur permettent pas de faire des bénéfices. Leurs incertitudes de production est aussi un frein à leur participation. Par ailleurs, le traitement des déséquilibres par l'opérateur de marché, entraînant des coûts supplémentaires pour chaque violation des contrats, n'est pas favorable aux producteurs distribués étant donné que ces derniers sont incertains par nature et donc plus enclins à ne pas respecter les puissances sur lesquels ils se sont engagés précédemment [22]. Une des solutions proposées par [3, 7, 23] est d'agréger ces producteurs distribués pour réduire l'incertitude globale et avoir du poids pour participer au marché de gros.

1.2.4 Marchés de l'électricité pair à pair

Les marchés de l'électricité pair à pair consistent en des marchés sans entité centrale, où chaque échange prend place directement entre deux groupes d'agents du marché (ou entre deux agents du marché lorsque le groupe se limite à un agent). On peut comparer en Figure 1.7 les topologies de communication pour un marché classique avec une gestion centralisée, et pour un marché pair à pair. On voit qu'un marché centralisé présente un goulet d'étranglement des communications car tous les échanges passent par le gestionnaire de marché. Au contraire, chaque groupe du marché pair à pair ne doit gérer que ses propres échanges avec ses partenaires commerciaux, donc il n'y a pas de risque de goulet d'étranglement. Le choix des partenaires peut se faire de différentes manières : proximité géographique ou électrique, partenaires historiques, contractuels, ... Ce choix peut être statique ou bien dynamique en fonction du temps.

Les échanges multilatéraux sont purement économiques tandis que les considérations des contraintes physiques sont prises en compte par le gestionnaire de réseau en dehors du marché [24]. Les différents mécanismes de marché pair à pair sont listés dans [25].



(a) Centralisé (b) Pair à pair

FIGURE 1.7 – Topologies de deux types de marché différents : le marché centralisé avec un gestionnaire de marché noté G qui effectue la résolution du marché, et le marché pair à pair qui fonctionne sans gestionnaire central. Les carrés bleus et roses sont des agents du marché et les flèches grises représentent les liens de communication.

On peut citer parmi les plateformes existantes pair à pair d'échange d'électricité [26, 25] : Piclo au Royaume-Uni [27, 28], Vandebron aux Pays-Bas [29], sonnenCommunity en Allemagne [30] et Brooklyn Microgrid à New York utilisant la technologie blockchain [31, 32].
Les plateformes de marché pair à pair peuvent être étudiées sous des paradigmes différents [33] :

- la **théorie des jeux** : porte sur les comportements coopératifs ou de compétition entre les différents agents dans le but de trouver une solution stable, parfois optimale et mutuellement bénéfique [34, 35, 36] ;
- la **théorie des enchères** : porte sur les interactions entre les acheteurs et les vendeurs qui s'échangent entre eux des volumes d'électricité ;
- l'**optimisation sous contraintes** : porte sur l'aspect mathématique et de programmation pour optimiser différentes fonctions en prenant en compte des contraintes imposées par le marché ou le système de puissance ;
- la **blockchain** : porte sur la structure de données pouvant être répliquée et partagée par les agents pour permettre des échanges sécurisés, décentralisés et transparents.

Biens échangés sur le marché Les biens échangés sur un marché pair à pair peuvent être de même nature que les biens échangés sur un marché centralisé : des biens primaires représentant des volumes d'électricité, ou des biens secondaires associés aux marchés auxiliaires (*ancillary markets*) qui participent au bon fonctionnement du réseau. Le marché primaire se concentre sur des volumes d'électricité étant injectés et soutirés du réseau électrique à une date de livraison donnée. Certains auteurs [37, 38] considèrent aussi des échanges de "negawatts", où les consommateurs vendent de l'effacement de consommation à la date de livraison prévue, c'est-à-dire où les consommateurs vendent à d'autres consommateurs le droit de consommer plus. En ce qui concerne les marchés secondaires, le marché des capacités se concentre sur les réserves de puissances actives qui peuvent être mobilisées ou non à la date de livraison considérée, selon l'état du réseau électrique. Ces réserves peuvent être des

moyens de production ou d'effacement de consommation électriques [39]. Là où aujourd'hui, des contrats sont réalisés entre le gestionnaire de réseau et les fournisseurs d'électricité ou d'effacement, on peut aussi imaginer que ces échanges prennent place sous la forme d'un marché pair à pair. Un autre marché pair à pair est proposé dans le but de soutenir le réseau électrique durant les périodes de pics de consommation en réduisant la demande en puissance durant le pic [40]. Des mesures incitatives peuvent être mises en place sur le marché pour utiliser les flexibilités des ressources distribuées [41].

Avantages et défis du marché pair à pair Le marché pair à pair présente certains avantages par rapport à un marché centralisé classique : (1) tous les acteurs du réseau peuvent être impliqués dans le marché, (2) la structure pair à pair permet le passage à l'échelle du mécanisme de résolution de marché, (3) assure la confidentialité des données des participants, (4) apporte la possibilité de définir des préférences hétérogènes et (5) propose des mécanismes au service du gestionnaire du réseau électrique.

Le contexte de l'inflation énergétique a abouti à une prise de conscience individuelle de la consommation énergétique ainsi que de la provenance de l'énergie consommée. L'**implication** dans le marché de tous les participants au réseau électrique, peu importe leur taille et leur capacité et notamment les producteurs distribués [42] et les consommateurs résidentiels, est une volonté du paradigme des Smart Grids, où la gouvernance est distribuée au niveau de tous les utilisateurs du réseau électrique. Chaque utilisateur doit avoir le pouvoir de régler ses préférences d'achat et de vente indépendamment des autres. Les consomacteurs contrôlent les conditions des transactions et la livraison de la puissance et des services promis [7], ce qui leur permet de s'assurer des revenus substantiels [34]. Aussi, le marché pair à pair peut être un moyen de coordination et d'agrégation de consomacteurs de petite taille pour pouvoir participer à des marchés de gros sous l'identité d'une centrale virtuelle [7]. L'organisation décentralisée des marchés pair à pair permet aussi de respecter la structure fortement décentralisée de la production électrique.

Le **passage à l'échelle** du problème de marché est résolu par la plateforme décentralisée qui permet de prendre en compte un grand nombre d'agents participant au même marché. En effet, le nombre de producteurs distribués a augmenté significativement et est amené à prendre une part importante du mix électrique. De plus, on peut imaginer que les foyers équipés de panneaux photovoltaïques et de batteries puissent participer aux échanges sur le réseau, pour mutualiser la production renouvelable ainsi que la flexibilité apportée par la batterie. Le réseau peut aussi être soutenu par les technologies *vehicule-to-grid* (V2G) [43]. Pour aller plus loin, n'importe quel foyer aux consommations mixtes (flexibles et non flexibles) peut participer au marché en mettant en enchère sa flexibilité. Tous ces acteurs représentent un nombre incommensurable d'agents potentiels du marché, qu'il aurait été impossible de prendre en compte via un mécanisme de résolution centralisée à cause notamment du goulet d'étranglement de communication que cela entraîne ainsi que de l'augmentation de la complexité du problème qui le rend difficile à résoudre de manière centralisée.

La **confidentialité** des données de chaque participant est garantie en ce que ces données n'ont pas besoin d'être partagées avec un gestionnaire de marché. C'est à chaque participant de chercher leurs interlocuteurs et de trouver quel est le meilleur prix en fonction de leurs préférences locales.

L'application de **préférences hétérogènes** autres qu'économiques où les acteurs du marché peuvent privilégier certains producteurs par rapport à d'autres pour des raisons écologiques ou de proximité géographique par exemple [44]. Cela n'est pas possible dans un marché centralisé classique où les volumes d'électricité échangés sont équivalents et ne sont pas différenciés selon leurs provenance renouvelable ou non, par exemple.

Les **gestionnaires de réseaux** peuvent aussi bénéficier du marché pair à pair en ce qu'il permet la réduction des pics de consommations [35], la réduction des coûts d'investissement et d'opération [32], et l'amélioration de la fiabilité du réseau électrique via la participation et la coordination des producteurs distribués [7]. Bien que les contraintes physiques du réseau électrique ne soient pas prises directement en compte dans les marchés pair à pair, des mécanismes ont été proposés pour que la solution du marché respectent ces contraintes, comme l'ajout de prix exogènes décourageant les échanges qui impliquerait une surcharge des lignes [45], la participation dans les échanges pair à pair du gestionnaire de réseau [46], ou bien l'estimation de l'influence d'un échange de puissance sur la tension et les flux du réseau électrique [47].

L'absence d'une entité centrale régulatrice présente en revanche des défis à surmonter, notamment du côté des utilisateurs qui doivent faire confiance à la plateforme pair à pair pour assurer le respect des contrats passés, trouver un moyen de répercuter les conséquences des écarts aux contrats, ainsi que d'assurer que les échanges respectent les contraintes physiques du réseau électrique. De plus, l'anticipation du comportement du marché est complexe car il est difficile de modéliser tous les agents, notamment leurs prises de décision dans un environnement où des agents peuvent avoir des buts antagonistes [34]. En plus de ces contraintes inhérentes à l'absence d'agent de coordination du marché, s'ajoute la complexité apportée par le besoin en communication dû à l'explosion du nombre de participants, mais aussi au paradigme de marché pair à pair où chaque agent possède non plus un seul correspondant, le gestionnaire de marché, mais autant de correspondants que de partenaires commerciaux. Le mécanisme de marché pair à pair reposant complètement sur le réseau de communication, il devient extrêmement sensible aux différents aléas que le réseau de communication peut subir.

1.2.5 Prise en compte des contraintes réseaux : le problème de l'Optimal Power Flow

Aujourd'hui, le gestionnaire du réseau électrique étant responsable de la qualité de service sur le réseau, il lui en incombe la tâche de vérifier que les contraintes techniques soient bien respectées après que les agents du marché se soient mis d'accord sur les échanges de volumes d'électricité. Si certaines contraintes, notamment de congestion de lignes, ne sont pas respectées, le gestionnaire de réseau incite à modifier le point de fonctionnement jusqu'à résolution du problème. L'influence du gestionnaire du réseau est alors exogène sur le comportement des agents du marché.

Une autre manière de faire est d'intégrer les contraintes physiques dans le problème d'optimisation du marché de l'électricité. Historiquement, le réseau électrique français était géré par EDF qui possédait deux caractéristiques : c'était un monopole et c'était un agent intégré, ce qui lui permettait de facilement poser le problème de l'Optimal Power Flow (OPF) qui revient à minimiser les coûts de production tout en respectant les contraintes physiques du réseau. Ce problème, présenté pour la première fois dans [48, 49], se trouve être non linéaire et non convexe [50] dû à ses contraintes physiques, et est considéré comme NP-difficile [51]. Sa complexité rend malheureusement le problème très difficile à résoudre, d'autant plus si l'on considère des réseaux à grande échelle, car le nombre de variables à considérer est beaucoup plus grand que dans le cas d'un simple marché.

La Figure 1.8 présente un schéma d'un réseau typique que l'on peut rencontrer lorsque l'on résout le problème de l'OPF. On considère les tensions alternatives à chaque nœud du réseau (le niveau de tension v_n ainsi que la phase θ_n), les puissances actives et réactives injectées ou soutirées au niveau de chaque nœud (p_n et q_n) ainsi que les puissances actives et réactives traversant les lignes (p_{nm} et q_{nm} pour la ligne reliant les nœuds n et m).

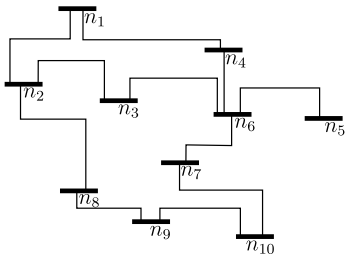


FIGURE 1.8 – Schéma d'un réseau électrique physique incluant les différents nœuds ainsi que les lignes qui les relient entre eux.

La relation entre les tensions et les puissances est non linéaire. Par exemple, pour une ligne modélisée par une simple réactance x_{nm}, la relation entre les tensions aux nœuds n et m et la puissance active traversant la ligne est donnée par (1.1).

$$p_{nm} = v_n v_m \frac{\sin(\theta_n - \theta_m)}{x_{nm}} \tag{1.1}$$

C'est cette relation non linéaire qui rend le problème de l'OPF non convexe et difficile à résoudre. Les autres contraintes consistent en la conservation de la puissance en chaque nœud, ainsi que les limites de puissance traversant les lignes et injectées ou soutirées du réseau électrique. On cherche aussi à limiter les niveaux de tension ainsi que les différences d'angle de part et d'autre de chaque ligne.

La résolution de l'Optimal Power Flow centralisé subit les mêmes inconvénients que celle d'un marché centralisé : d'abord, le passage à l'échelle est d'autant plus compliqué que les contraintes considérées sont non linéaires et le nombre de variables est beaucoup plus conséquent et requiert donc plus de puissance de calcul ; ensuite, la résolution centralisée ne permet pas la confidentialité des données : l'entité effectuant le calcul a besoin des données des agents du marché telles que leurs fonctions coût et leurs limites de puissance, mais a aussi besoin d'informations sur le réseau électrique comme les admittances des lignes, les rapports de transformation. Aujourd'hui, la séparation de la gestion du réseau physique et de la production d'électricité rend difficile la résolution centralisée de l'OPF puisque les informations relatives aux coûts de productions appartiennent aux producteurs et les informations relatives au réseau appartiennent au gestionnaire de réseau.

Différentes méthodes sont possibles pour résoudre le problème d'OPF, notamment des méthodes de décomposition qui permettent de découper le problème global en plusieurs problèmes de petites tailles. La distribution de la résolution sur plusieurs machines permet de la rendre faisable sur le plan informatique. Elle peut se faire de manière locale sur plusieurs cœurs d'une même machine ou bien de manière éclatée sur plusieurs machines se trouvant dans des réseaux de communication différents. Dans le dernier cas, il est possible de garantir une confidentialité locale des données.

L'Optimal Power Flow décentralisé La résolution du problème d'Optimal Power Flow décentralisé [52, 53] peut se faire soit en considérant les contraintes exactes non convexes, soit en réalisant des approximations pour faciliter les calculs.

Dans le premier cas, plusieurs méthodes peuvent être employées comme les méthodes d'*auxiliary problem principle*, *predictor corrector proximal multiplier method*, *alternating directions method of multipliers* ou ADMM [54, 55], *optimality condition decomposition* [56], *augmented Lagrangian alternating direction inexact Newton method* [57, 58].

Dans le second cas, trois types d'approximations principales sont possibles.

— L'approximation DC, où les puissances réactives et les pertes dans les lignes sont négligées : les contraintes sont ainsi linéarisées ce qui rend le problème DC-OPF beaucoup plus simple à résoudre que l'AC-OPF [59].
— La dépendance au carré de la tension par rapport aux puissances permet d'utiliser la méthode *second order cone programming* pour résoudre l'OPF : SOC-OPF [60, 61].
— La méthode plus générale de convexification basée sur l'optimisation SDP (pour *semi-definite programming*) est utilisée pour résoudre le problème SDP-OPF [62, 63].

Ces méthodes d'approximation sont un compromis à faire pour obtenir une simplification des calculs [64] en renonçant à l'exactitude de la solution obtenue. Dans ce manuscrit, nous seront amenés à étudier le problème d'Optimal Power Flow exact, avec une décomposition basée sur la méthode ADMM.

Considérations sur la décomposition et les échanges Comme mentionné plus haut, la décomposition d'un problème consiste à découper le dit problème en plusieurs sous-problèmes de petite taille. Dans le cas du problème d'OPF, il s'agit alors de partitionner les nœuds du réseaux en plusieurs groupes, ou régions, et associer à chaque groupe de nœuds son propre sous-problème à résoudre. Ces régions peuvent être de tailles variables, et on peut aussi considérer le cas où une région ne contient qu'un seul nœud, c'est-à-dire le cas où la décomposition se fait nœud par nœud.

À chaque région est associée un agent de calcul qui s'occupe de résoudre le sous-problème local. Or, cet agent de calcul, bien que local à la région, doit collecter les données des agents associés aux nœuds locaux. Si la région contient plus d'un nœud, alors on se retrouve dans le même problème de partage de données que pour le cas centralisé, à moindre échelle puisque l'agent de calcul ne s'occupe que des agents locaux et ne partage pas leurs données avec les autres régions.

Le choix des différentes régions est aussi une question digne d'intérêt. Les auteurs de [55, 65] proposent des méthodes de clustering spectral pour déterminer la meilleure décomposition qui accélère la résolution d'OPF basés sur la méthode OCD [65] ou sur la méthode ADMM [55]. Cependant, il est possible que la décomposition des régions dépende d'autres facteurs comme la géographie, si une région correspond aux nœuds d'un pays par exemple, la proximité des nœuds en termes de connexion électrique, ou bien du fait de différents gestionnaires de réseaux de distribution.

Lors de la résolution décentralisée du problème d'OPF, les échanges d'information se font de proche en proche du point de vue du réseau électrique étant donné les contraintes réseaux qui lient les tensions de par et d'autre de chaque ligne. En revanche, dans un marché pair à pair, les échanges peuvent se faire entre chaque pair participant au marché, peu importe la distance physique des pairs sur le réseau électrique. La matrice de communication de l'OPF décentralisé est par nature parcimonieuse tandis que celle pour le problème de marché pair à pair a la possibilité d'être remplie. L'influence de chaque agent se répand de proche en proche dans un problème d'Optimal Power Flow décentralisé tandis que la communication directe dans le problème de marché pair à pair permet de trouver rapidement une solution, d'autant plus qu'il n'y a pas de contraintes non linéaire qui rend le problème non convexe.

1.3 Algorithmes asynchrones

Les travaux décrits dans ce manuscrit portent sur l'étude de l'influence des aléas de communication sur la résolution d'algorithmes décentralisés appliqués au réseau électrique et dont la décomposition se base sur la méthode ADMM. Les aléas de communication considérés sont les délais de communication ainsi que les pertes de messages sur le réseau de communication. Nous étudierons l'influence de ces aléas sur un algorithme de marché pair à pair basé sur des travaux antérieurs de l'équipe portés par Baroche [45], sur plusieurs algorithmes décentralisés d'Optimal Power Flow, dont l'un basé sur les travaux de Erseghe [54] sur un algorithme d'OPF décentralisé par régions, ainsi que sur un algorithme de marché pair à pair où le gestionnaire de réseau agit de manière endogène sur le marché, dans la continuité des travaux de Baroche [46].

1.3.1 Introduction de la notion d'algorithme asynchrone

Dans le cadre d'une résolution classique de ces algorithmes décentralisés, la perte d'un message serait préjudiciable à l'avancement de l'algorithme. En effet, puisqu'à chaque itération tous les agents ont besoin de connaître les données de leurs voisins, comme illustré dans l'algorigramme présenté en Figure 1.9, une perte de message entraînerait l'arrêt local de l'algorithme, puis de proche en proche, cet arrêt se propagerait à tous les agents du marché. Dans une moindre mesure, imaginons que la transmission d'un seul message prenne beaucoup plus de temps que tous les autres messages échangés.

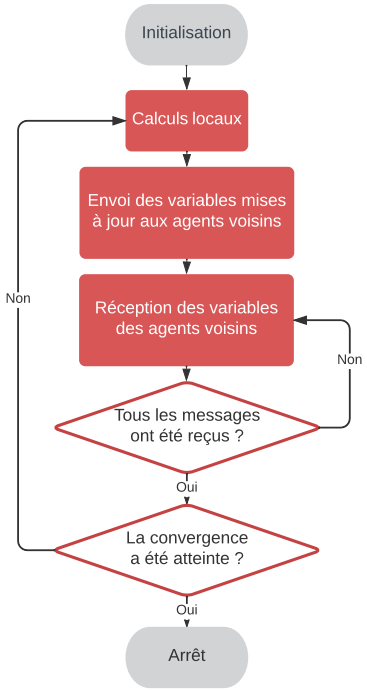
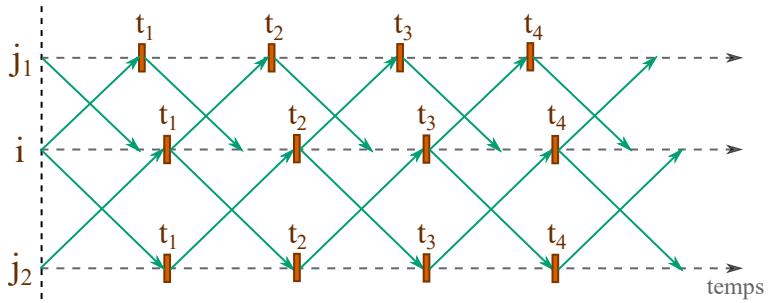


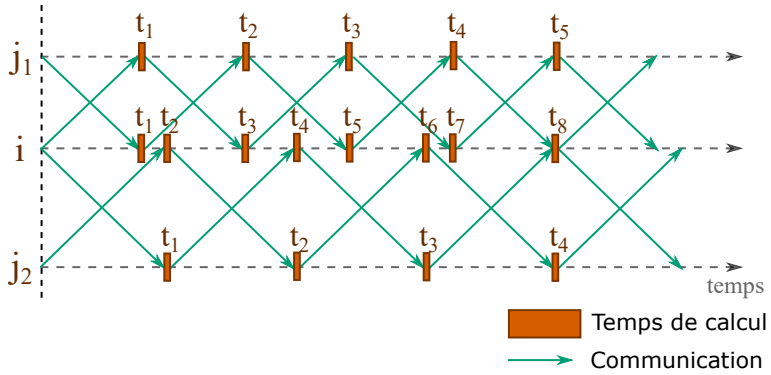
FIGURE 1.9 – Algorigramme simplifié d'un agent de calcul local dans le cadre d'un algorithme de consensus distribué. À chaque itération, l'agent passe par une étape d'attente de message qui ne se termine que lorsque les messages de tous les voisins de l'agent ont été reçus.

La majorité du temps d'attente de l'agent destinataire sera alors consacrée à l'attente de l'arrivée de ce dernier message, décalant les calculs locaux de l'itération qui suit et ainsi, de la même manière que décrit précédemment, le retard d'un seul message se répercute de proche en proche sur l'intégralité des agents du marché.

Dans le cas où le réseau de communication est hétérogène, les disparités dans les délais de communication entraînent alors des temps d'attente élevés, qui pourraient être évités en modifiant la règle d'attente de l'intégralité des messages : un agent réalise les calculs de l'itération après avoir reçu une portion de tous les messages attendus. Le temps d'attente est grandement réduit, et les messages non traités lors d'une itération seront traités lors d'une itération future. Les algorithmes itératifs où l'on ne prend pas en compte l'intégralité des messages à chaque itération s'appellent les algorithmes *asynchrones*, car alors chaque itération locale est indépendante de l'avancement des autres agents. On présente en Figure 1.10 une frise chronologique des échanges entre les agents décentralisés dans le cas synchrone en Figure 1.10a et dans le cas asynchrone en Figure 1.10b, sur un cas théorique à trois agents.



(a) Synchrone



(b) Asynchrone

FIGURE 1.10 – Échanges de messages en fonction du temps pour problème décentralisé constitué de trois agents. La Figure 3.10a présente les échanges dans le cas de l'algorithme synchrone tandis que la Figure 3.10b les présente en asynchrone.

Dans le cas asynchrone, l'agent i n'a pas à attendre les deux messages de ses partenaires avant de continuer ses calculs, ce qui permet à l'agent j_1 de réaliser plus d'itérations durant un même délai. Ainsi, ces figures illustrent le gain de temps qui peut être réalisé en asynchrone par rapport à l'algorithme synchrone de base. Cependant, on peut légitimement

se demander si l'avancement de l'algorithme est plus rapide en asynchrone. En effet, en asynchrone, les agents ne disposent pas de l'intégralité des informations provenant de leurs voisins, ce qui nécessite peut-être de réaliser plus d'itérations pour atteindre le même niveau de convergence que pour l'algorithme synchrone.

Les algorithmes asynchrones ont fait l'objet d'études diverses lors des dernières années. Ces études sont décrites dans la section suivante.

1.3.2 État de l'art sur les algorithmes asynchrones

Bertsekas et Tsitsiklis [66] ont défini deux types d'algorithmes asynchrones, les algorithmes *totalement asynchrones* et les algorithmes *partiellement asynchrones*. Les premiers ont pour hypothèse que chaque lien entre processeurs, ou agent de calcul, recevra toujours un message avant un certain temps T lorsque le temps T tend vers l'infini. Les seconds ont une hypothèse plus forte que le délai de communication est borné. Certains algorithmes asynchrones ne convergent pas dans le cadre totalement asynchrone mais peuvent converger dans le cadre partiellement asynchrone. Cette définition est étoffée dans [67], qui rajoute les cas *partiellement synchrones* et *totalement synchrones*. On dit que la méthode est "synchrone" lorsque les messages ont le même ordre d'arrivée que d'envoi. Ils sont "asynchrones" sinon.

Les algorithmes distribués asynchrones ont d'abord été étudiés dans le domaine de l'informatique. En effet, étant donné que l'algorithme de consensus ADMM est largement utilisé dans les applications de machine learning, plusieurs auteurs se sont penchés sur les différents ralentissements que peuvent subir ces algorithmes distribués dans le but d'améliorer leurs performances. À noter que dans ce type d'application, les machines de calcul sur lesquelles sont distribués les problèmes à résoudre se trouvent très proches les unes des autres : il s'agit parfois même de plusieurs cœurs d'une même machine qui travaillent en parallèle. Les délais de communication sont donc différents que dans une application décentralisée sur des machines se trouvant dans des réseaux de communication différents.

Dans [68, 69] le caractère asynchrone de la résolution est réalisé en choisissant pour chaque itération un agent au hasard. Cet agent est alors le seul à mettre à jour son problème local pour l'itération considérée. Ce type d'algorithme asynchrone entraîne des omissions d'informations. Plusieurs stratégies sont présentées en ce qui concerne la gestion des informations omises : la valeur de la variable omise peut être prédite, rétablie à la valeur précédente ou ignorée complètement. Dans [70, 71], un agent central *master* collecte les informations des travailleurs distribués de manière asynchrone, ce qui signifie qu'il n'attend pas les résultats de chaque travailleur. Le maître lance son calcul après avoir reçu un nombre minimum de mises à jour et avec une condition de délai borné, c'est-à-dire que chaque information variable utilisée par le maître doit avoir au plus τ itérations. En tenant compte de cette condition de délai borné, la preuve de convergence de l'algorithme asynchrone est fournie dans [70]. Dans [72], l'algorithme est distribué par blocs sur plusieurs machines de calcul "workers". L'aspect asynchrone vient du fait que la variable de consensus est mise à jour de manière incrémentale à chaque réception de la mise à jour d'un worker. Les workers utilisent la dernière version de la variable de consensus qu'ils ont reçue. La preuve de convergence est donnée sous l'hypothèse de délais bornés. Dans [73, 74], le problème devient totalement décentralisé. Dans les deux articles, l'asynchronisme provient d'un délai borné donné, et il est montré qu'il permet d'améliorer significativement le temps de convergence total. Cependant, la fonction objective finale diffère de la solution synchrone. De plus, les délais de communication sont soit empiriques soit tirés aléatoirement d'une distribution uniforme, il n'y a donc pas de délai de communication supplémentaire provenant d'un modèle désigné

introduit dans ces articles, ce qui semble raisonnable étant donné que les temps de calcul sont prédominants sur les délais de communication dans le domaine considéré. La preuve mathématique de la convergence d'un consensus décentralisé basé sur ADMM est donnée dans [75]. Dans cet article, deux types de décomposition sont étudiés : la décomposition basée sur les nœuds et la décomposition basée sur les arêtes. En considérant la communication à chaque itération comme un sous-graphe d'un graphe primaire, avec le même nombre de nœuds mais avec moins d'arêtes, la preuve de convergence du premier algorithme requiert que l'intersection de tous les sous-graphes des itérations ne soit pas vide. Cette hypothèse est difficile à réaliser en considérant un niveau d'asynchronisme élevé. Cependant, l'algorithme basé sur les arêtes exige seulement que l'union de tous les sous-graphes soit égale au graphe primaire. Cette hypothèse est facilement réalisable même lorsque l'asynchronisme est important.

Dans [76, 77] , le problème d'estimation de mode, qui consiste en l'estimation des oscillations de phase et de fréquence dans les différents nœuds du réseau électrique, est résolu par un algorithme ADMM asynchrone distribué avec des seuils de retard bornés pour l'agent central et les agents locaux. Les auteurs ont pu constater que l'erreur de convergence n'était plus monotone à mesure que l'algorithme devenait de plus en plus asynchrone. Le modèle de délais de communication utilisé est le même que dans [78].

Le problème d'OPF est également résolu avec des algorithmes asynchrones dans [78, 79, 67, 80]. Dans [79], le problème général d'optimisation sous contraintes non linéaires (AC-OPF) est résolu avec ADMM asynchrone de manière complètement décentralisée. Le déclenchement d'une itération provient du nombre de messages qu'un agent reçoit, et les délais de communication sont générés aléatoirement dans un intervalle donné. L'AC-OPF est également résolu dans [67] en utilisant une méthode de relaxation lagrangienne asynchrone. Le marché global est divisé en plusieurs marchés locaux qui communiquent entre eux de manière asynchrone. L'ordre des messages correspond principalement à la différence des temps de calcul des agents. Le taux de convergence de l'algorithme asynchrone est plus élevé que celui de l'algorithme synchrone s'il y a une différence significative dans la vitesse de calcul des agents. [78] aborde également le problème OPF dans une approche distribuée avec une coordination limitée des nœuds voisins. Les auteurs ont utilisé un modèle avancé de délais de communication et ont testé plusieurs stratégies robustes aux différents délais pour améliorer leurs résultats. La meilleure stratégie qu'ils ont trouvée consistait à estimer l'information manquante due au seuil asynchrone avec un modèle auto-régressif pondéré. Dans [81], un AC-OPF relaxé est résolu en décomposant nœud par nœud le problème via la méthode ADMM, où les calculs de chaque nœud sont planifiés selon la topologie du réseau, ce qui permet une convergence plus rapide. L'étude de l'AC-OPF avec déséquilibre de puissance nul utilisant des multiplicateurs lagrangiens est réalisée dans [82], où l'implémentation asynchrone consiste à ne pas mettre à jour toutes les variables à chaque itération. L'algorithme asynchrone converge vers la solution optimale.

L'approximation linéaire de l'AC-OPF (DC-OPF) susmentionnée a été résolue de manière asynchrone dans [80] d'une manière totalement décentralisée pour les applications de micro-réseaux. Les agents de calculs ignorent délibérément certaines itérations pour accélérer le temps de calcul global. Naturellement, plus le nombre d'itérations ignorées est élevé, plus les résultats sont éloignés de la solution optimale par rapport à l'algorithme synchrone. Le DC-OPF dans [83] est résolu en utilisant des multiplicateurs de Lagrange pour distribuer le problème entre les différents bus du réseau. Les mises à jour sont asynchrones dans le sens où les bus d'une même zone échangent des informations après chaque itération, tandis que la communication entre les zones ne se produit qu'après plusieurs itérations. Cela permet de réduire la communication entre les régions. Cependant, les temps de calcul et les délais

de communication n'ont pas été pris en compte.

Les auteurs de [84] étudient la combinaison d'un système électrique et d'un système de chauffage en utilisant une méthode ADMM relaxée pour répartir l'optimisation entre un système électrique et plusieurs systèmes de chauffage urbain. Le problème considéré est un problème de minimisation de coûts du système global sur plusieurs pas de temps considérés. Les résultats montrent que la méthode ADMM relaxée permet de converger vers la solution optimale en présence de pertes de messages, contrairement aux résultats de la méthode ADMM classique qui ne sont pas optimaux. De plus, [85] étudie un système électrique et de chauffage similaire pour une optimisation combinée de la gestion de l'énergie en temps réel et à l'avance. Ils utilisent une communication basée sur le déclenchement d'événements afin de réduire le nombre de messages échangés. Une décomposition du problème d'*unit commitment* basée sur l'ADMM par région est étudiée dans [86]. Les auteurs ont implémenté l'algorithme en utilisant la communication *message passing interface* (MPI [20]) pour obtenir des temps de calcul et de communication réels. Le caractère asynchrone est ici obtenu en n'attendant pas tous les messages à chaque itération. Ils ont montré que cela permettait une convergence plus rapide en échange d'un écart d'optimalité plus important.

Les modèles de délai de communication dans les travaux de la bibliographie présentés sont variés. Comme les algorithmes de consensus asynchrones ont d'abord été étudiés depuis le domaine de l'informatique, les délais de communication ont été considérés comme uniformes pour tous les agents de calcul. En effet, ces agents de calcul distribués sont soit sur le même ordinateur et utilisent la communication MPI, soit ils sont situés dans le même sous-réseau de communication. Ainsi, des modèles de communication simples tels qu'une probabilité fixe d'arriver à chaque itération [71, 72] ou une probabilité de perte de messages [74] ont été utilisés. Certains ont même utilisé des délais de communication empiriques en implémentant directement leurs algorithmes sur du matériel [70, 73]. En étendant les algorithmes asynchrones aux problèmes liés aux réseaux électriques, certains articles ont choisi de continuer à utiliser des modèles de délais de communication simples : [79] échantillonne un délai à partir d'une distribution uniforme là où [67] utilise un délai fixe unique de 10 ms et s'appuie sur les différences empiriques de temps de calcul. Cependant, compte tenu de l'application d'un algorithme distribué à un réseau électrique où les agents de calcul peuvent ne pas être situés dans les mêmes zones, ces modèles de délais de communication susmentionnés peuvent être limités par rapport à une application réelle. Par conséquent, certains articles ont utilisé des modèles de délai de communication plus complexes, comme dans [87] qui utilise une distribution exponentielle, ou dans [72, 77] qui utilise un modèle de délai avancé pour la communication à grande échelle. Ces modèles tiennent compte des retards de communication plus importants échantillonnés plus fréquemment, comme c'est le cas dans les applications du monde réel.

20. `https://fr.wikipedia.org/wiki/Message_Passing_Interface`

1.4 Contributions et organisation du manuscrit

Les contributions de la thèse développées dans ce manuscrit sont les suivantes.

- L'**implémentation asynchrone** des algorithmes de marché pair à pair ou d'Optimal Power Flow décentralisés présents dans l'état de l'art. À notre connaissance, de telles études exhaustives d'implémentations asynchrones appliquées aux algorithmes présentés sont inédites. Les codes associés sont disponibles sur Gitlab en open-source : `https://gitlab.com/alyssiadong/AsyncDecElec.jl`.
- Le développement d'une **plateforme de simulation** prenant en compte les échanges de messages lors de la résolution d'algorithmes décentralisés. Cette plateforme de simulation est implémentée de manière à être flexible et reconfigurable, et se veut open source.
- Le **modèle de délais de communication paramétrique** permettant de modéliser les variations de délais de communication, qu'elles soient d'ordre stochastique ou bien dues à la topologie du réseau de communication, est un apport primordial qui se distingue de l'état de l'art.
- La prise en compte simultanée des **délais de communication et des temps de calcul**, ainsi que l'étude approfondie de leurs impacts sur le temps de convergence global des algorithmes décentralisés synchrones et asynchrones, sont aussi à notre connaissance, des apports inédits de ce travail de thèse.
- Le **déploiement opérationnel** d'un algorithme asynchrone de marché pair à pair sur une plateforme expérimentale.

Le travail réalisé a conduit à plusieurs communications scientifiques listées ci-après :

- Alyssia Dong, Thomas Baroche, Roman Le Goff Latimier et Hamid Ben Ahmed, « Convergence analysis of an asynchronous peer-to-peer market with communication delays », in : *Sustainable Energy, Grids and Networks* 26 (2021), p. 100475, ISSN : 2352-4677, DOI : 10.1016/J.SEGAN.2021.100475

- Alyssia Dong, Thomas Baroche, Roman Le Goff Latimier, Hamid Ben Ahmed et Roman LE Goff Latimier, « Implémentation asynchrone d'un algorithme de marché de l'électricité décentralisé », in : *Symposium de Génie Électrique*, 2020, URL : `https://hal.archives-ouvertes.fr/hal-03485938`

Les résultats de ces publications seront développés au Chapitre 3.

- Alyssia Dong, Thomas Baroche, Roman Le Goff Latimier et Hamid Ben Ahmed, « Asynchronous algorithm of an endogenous peer-to-peer electricity market », in : *2021 IEEE Madrid PowerTech, PowerTech 2021 - Conference Proceedings* (2021), DOI : 10.1109/POWERTECH46648.2021.9495009

Les résultats de cette publication seront développés au Chapitre 5.

- Alyssia Dong, Roman Le Goff Latimier et Hamid Ben Ahmed, « Asynchronous implementation of a distributed optimal power flow algorithm », 2022, working paper

- Alyssia Dong, Roman Le Goff Latimier et Hamid Ben Ahmed, « Asynchronous ADMM based decentralized optimal power flow with edge decomposition », 2022, working paper

Les résultats de ces publications seront développés au Chapitre 4.

La prise en compte des différents délais influant sur le temps de convergence des algorithmes décentralisés considérés est réalisée au travers d'une simulation numérique. La plateforme de simulation ainsi que les différents modèles de délais de communication et de temps de calculs sont présentés au Chapitre 2 de ce manuscrit. Ensuite, le Chapitre 3 se concentrera sur l'algorithme de marché pair à pair résolu de manière asynchrone. On y étudiera les différentes influences des délais de communication et de calculs sur la version asynchrone de l'algorithme, et on pourra comparer les temps de convergence obtenus avec ceux de la version synchrone. Le Chapitre 4 portera sur la résolution asynchrone d'algorithmes d'Optimal Power Flow décentralisés : deux algorithmes présentant des méthodes de décomposition différentes seront aussi étudiés en asynchrone. Enfin, on s'intéressera dans le Chapitre 5 à un algorithme de marché pair à pair où les contraintes physiques sont prises en compte de manière endogène au marché, c'est-à-dire où un opérateur système vérifie ces contraintes et interagit à chaque itération avec les agents du marché dans le but final d'atteindre la solution du problème d'OPF.

Chapitre 2

Modèles et plateforme de simulation

Sommaire

On présente dans ce chapitre la plateforme de validation des algorithmes et les modèles de délais de communication et de calculs utilisés dans les chapitres suivants. Cette plateforme doit être capable de simuler les différents agents de calculs des algorithmes décentralisés, ainsi que leurs échanges de messages au cours du temps. Il apparaît alors nécessaire de déterminer un modèle de délais de communication entre deux agents, ainsi qu'un modèle de temps de calcul, pour pouvoir étudier l'influence de ces délais sur les algorithmes.

On présente en section 2.1 un modèle de délais de communication paramétrique, prenant en compte des variations stochastiques de la latence, ainsi que les éventuelles pertes de messages. Ensuite, on présente en section 2.2 un modèle de temps de calcul, dépendant du type de problème résolu ainsi que de la taille du problème. Ces modèles sont enfin utilisés en section 2.3 où les différents agents de calculs et leurs échanges sont simulés sur une plateforme de simulation à événements discrets.

2.1 Modélisation de la communication

2.1.1 Transmission d'informations via le réseau de communication

Structure du réseau de communication Un réseau informatique désigne un ensemble d'équipements qui échangent des informations au travers d'interconnexions. Dans le domaine de l'informatique, on ne parle pas d'un réseau unique mais de multiples réseaux interconnectés. Les réseaux informatiques peuvent être de tailles variées : des réseaux locaux à l'échelle d'un même site (Local Area Network, LAN) jusqu'aux réseaux étendus à l'échelle d'un pays ou plus (WAN, Wide Area Network). Le plus célèbre des réseaux étendus WAN est Internet qui couvre la totalité de la planète.

Il existe différentes topologies de réseau informatique selon les applications considérées. La topologie la plus courante pour un réseau local est la topologie en étoile où les équipements sont tous connectés à un concentrateur ou un commutateur central. Les réseaux

locaux peuvent être connectés aux réseaux étendus au travers de routeurs, comme l'illustre la Figure 2.1. Le réseau Internet est un réseau à topologie maillée où tous les hôtes sont connectés pair à pair sans hiérarchie centrale, ce qui évite d'avoir un point critique qui s'il tombe en panne, isole une partie du réseau.

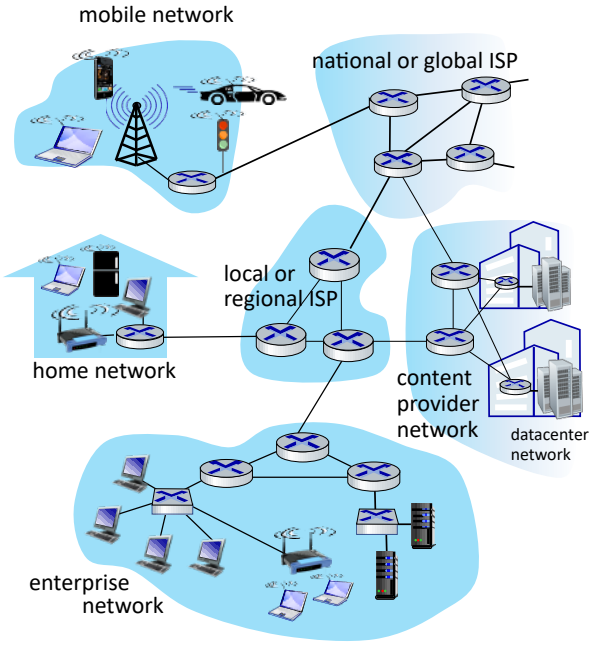


FIGURE 2.1 – Illustration provenant du manuel de Kurose et Ross [93], représentant les connexions entre des réseaux locaux LAN tels que des réseaux domestiques ou institutionnels, à des réseaux régionnaux ou globaux (WAN).

On peut décrire le réseau de communication au travers du modèle OSI, présenté en Figure 2.2, qui décrit les fonctionnalités nécessaires à la communication. Les trois couches inférieures sont orientées communication et sont implémentées sur les équipements du réseau ainsi que sur les machines qui communiquent entre elles. Les quatre couches supérieures sont plutôt orientées application et implémentées seulement sur les machines communicantes. Les différents protocoles utilisés dans le réseau de communication affectent différentes couches du modèle OSI.

Pour la couche 4-Transport, deux types de protocoles principaux peuvent être considérés selon les exigences de l'application : UDP (pour *User Datagram Protocol*) ou TCP (pour *Transmission Control Protocol*). Le premier protocole est adapté à un usage de type voix sur IP ou jeux en ligne où la perte éventuelle d'un paquet est préférée à l'attente de sa retransmission. En effet, comme il n'existe pas de garantie quant à la livraison ou l'ordre d'arrivée des données, on désigne le protocole UDP comme un protocole non fiable. Le second protocole TCP est basé sur un fonctionnement en trois temps : une connexion est d'abord établie avant le transfert de données, et une fois que tous les échanges ont été réalisés, la connexion est interrompue. Un mécanisme d'acquittements est mis en place pour garantir l'arrivée des paquets dans le bon ordre. Une temporisation permet de retransmettre un paquet après un certain temps sans acquittement. Pour cela, on dit que TCP est un

protocole "sans pertes", c'est-à-dire que dès lors qu'un paquet est perdu, il est ré-envoyé autant de fois que nécessaire jusqu'à ce qu'il atteigne sa destination. Ainsi, les pertes réelles de paquets n'induisent du point de vue des couches supérieures que des délais supplémentaire de communication. C'est aussi un protocole fiable car il détecte les éventuelles erreurs de transmission. Étant donné que l'application qui nous concerne dans ce manuscrit a besoin d'une transmission sans pertes et fiable, on considère alors que le protocole de la couche Transport utilisé est le protocole TCP.

Dans les couches inférieures, des mécanismes de vérification des données, comme le bit de parité ou encore les codes correcteurs, permettent de détecter et parfois même corriger les erreurs de transmission. Les liaisons sans fil proposent aussi des codes convolutionnels, des turbo codes ou des codes polaires qui présentent une forte redondance de l'information permettant de corriger un plus fort taux d'erreur binaire.

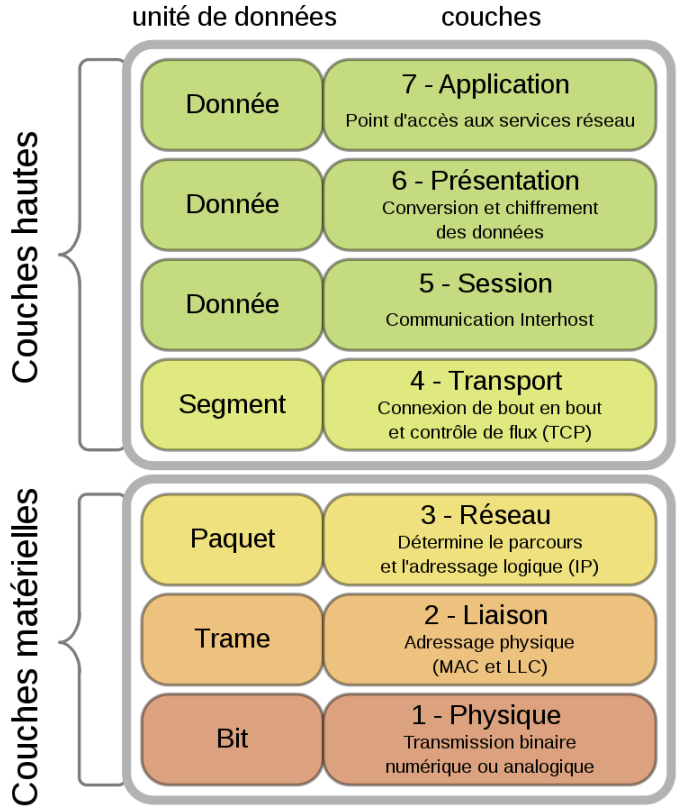


FIGURE 2.2 – Les différentes couches du modèle OSI, illustration provenant de `https://fr.wikipedia.org/wiki/Modèle_OSI`.

Origine des délais de communication On considère qu'un échange consiste en un seul paquet envoyé d'un agent A vers un agent B, c'est-à-dire que le message n'est pas fragmenté en plusieurs paquets. En effet, la longueur maximale d'un segment pour le protocole TCP est de 1500 octets, tandis que les données à communiquer sont de l'ordre de la dizaine d'octets. Étant donné qu'à chaque itération, un agent doit communiquer soit une valeur de

puissance (au Chapitre 3) soit une valeur de tension complexe (au Chapitre 4), ainsi que des informations sur l'itération locale considérée, il est raisonnable de supposer qu'un seul paquet est nécessaire pour l'échange entre deux agents à une itération donnée.

Ce paquet part de l'agent expéditeur et passe par divers équipements du réseau informatique avant d'être réceptionné par l'agent de destination. Si les deux agents communiquant se trouvent dans le même sous-réseau, alors le paquet passera probablement par un concentrateur ou un commutateur avant d'arriver à destination, ce qui rend son trajet relativement court et rapide, sans trop de risque de pertes. En revanche, si les deux agents se trouvent sur des sous-réseaux distincts mais connectés par un réseau étendu, comme par exemple Internet, le paquet devra passer non seulement par les routeurs locaux aux agents, mais aussi par tous les routeurs d'Internet se trouvant sur le chemin.

Lorsqu'un paquet arrive sur un routeur, il est stocké dans une file d'attente. En effet, les routeurs effectuent des traitements pour chaque paquet, ce qui ralentit leur retransmission. Si le réseau est congestionné, alors la file d'attente de chaque routeur rencontré risque d'être longue, ce qui impacte fortement le délai de communication de bout en bout. Le délai total de communication sans pertes peut être décomposé sous la forme suivante :

$$\text{Délai total} = \underbrace{\text{transmission} + \text{propagation} + \text{traitement}}_{\text{délai déterministe}} + \underbrace{\text{attente}}_{\substack{\text{délai} \\ \text{stochastique}}}$$

où le **délai de transmission** désigne le temps des équipements réseau à transmettre les bits constituant le paquet, le **délai de propagation** désigne la distance entre émetteur et récepteur divisé par la vitesse de propagation du milieu, le **délai de traitement** des équipements réseau dépend de leurs capacités et le **délai d'attente** désigne les délais passés dans les files d'attente des routeurs. À trajet fixe, les trois premiers termes sont constants et le quatrième terme dépend de la charge du réseau de communication. On peut alors représenter le délai total comme la somme entre un terme déterministe et un terme aléatoire.

$$\text{Délai total} = \text{Délai}_{det} + \text{Délai}_{sto} \tag{2.1}$$

À noter que le terme de délais stochastiques possède une valeur moyenne.

Origine des pertes de paquets Aussi, si la file d'attente d'un routeur est déjà totalement remplie lors de l'arrivée d'un paquet, alors ce dernier est perdu et on parle de chute de queue [1] (ou *tail drop*). Un paquet peut aussi se perdre lorsqu'il est passé par un nombre élevé de routeurs. On considère alors que le paquet n'a pas réussi à retrouver son chemin et qu'il vaut mieux le supprimer pour ne pas qu'il encombre le réseau. Enfin, une autre cause des pertes de paquets est lorsque le taux d'erreur binaire se montre élevé, notamment au travers de l'utilisation de réseaux sans fil ayant de mauvais taux de couverture comme le Wi-fi.

> Ainsi, différentes circonstances viennent perturber la transmission de messages sur le réseau de communication. Selon le niveau de congestion et la qualité des connexions du réseau, les messages sont soumis à des ralentissements, des erreurs ou même des pertes. Pour notre application, les grandeurs d'intérêt d'une transmission de messages sont : le délai point à point, le taux d'erreurs ainsi que le taux de pertes.

1. `https://fr.wikipedia.org/wiki/Perte_de_paquets`

2.1.2 Modèle de délais de communication

Plusieurs auteurs dans la littérature ont simulé en détails les réseaux de communication pour étudier leurs impacts sur le réseau électrique grâce à des plateformes de co-simulation [13, 14, 16]. Ces méthodes, bien que très précises quant à la simulation des erreurs de transmission ou des pertes de paquets, sont coûteuses en puissance de calcul. Dans le cadre de notre application, on cherche seulement à modéliser le temps que prend un message entre son expéditeur (agent A) et son destinataire (agent B), c'est-à-dire la latence d'un paquet de données, sans tenir compte des différents protocoles de communication utilisés ni des médiums par lesquels ces messages sont transmis.

On pose les hypothèses suivantes :

1. pour chaque envoi d'un message d'un agent A vers un agent B, il existe un chemin dans le réseau de communication entre A et B fonctionnel ;
2. le protocole de transport utilisé est muni d'un **système de détection et de correction d'erreurs**, ou de retransmission lorsqu'une erreur est détectée, on suppose que tout message envoyé est reçu à destination sans erreur ;
3. le protocole de transport utilisé possède un **système d'acquittement avec retransmission**, c'est-à-dire que dans le cas d'une perte de message, l'expéditeur doit renvoyer le message après un certain délai, et ce jusqu'à ce que le message soit effectivement reçu par le destinataire.

Ces hypothèses garantissent que le délai de communication d'un message est toujours borné, ce qui est une hypothèse nécessaire pour prouver la convergence d'un algorithme de consensus se basant sur la méthode ADMM, comme mentionné en section 1.3.2. De plus, les informations contenues dans le message ne sont pas erronées.

Dans un premier temps, on détermine le modèle de délai de communication pour un échange de message sans perte, donc sans retransmission. Le modèle de délais de communication se compose en deux termes : un terme moyen et un terme stochastique centré.

$$T_{AB} = \overline{T_{AB}} + t_{AB} \tag{2.2}$$

Le terme moyen $\overline{T_{AB}}$ est défini de manière à décrire la valeur moyenne du délai de communication entre deux agents A et B. Le terme stochastique t_{AB} est décrit par une variable aléatoire centrée, et représente les variations de délais de communication entre les deux mêmes agents A et B.

Nota

Le modèle présenté est différent du modèle de délai total (2.1) avec une partie déterministe et une partie stochastique. En effet, le terme stochastique de (2.1) possédait une valeur moyenne, tandis que le terme stochastique de notre modèle t_{AB} est centré. La valeur moyenne du terme stochastique en (2.1) est alors inclus dans le terme moyen $\overline{T_{AB}}$.

Partie moyenne Comme précisé plus haut, le délai de communication est affecté par de nombreux facteurs : le nombre d'équipement du réseau informatique par lequel il passe [2], le débit de ces équipements ou la charge moyenne du réseau de communication à un instant donné. On définit une distance informatique entre deux agents A et B : D_{AB}. Cette distance n'est pas directement associée à la distance physique, mais est représentative du

2. Comme par exemple des routeurs, des répéteurs, des commutateurs...

nombre d'équipements informatiques séparant les agents considérés. Le délai de communication moyen entre deux agents A et B est alors défini comme suit :

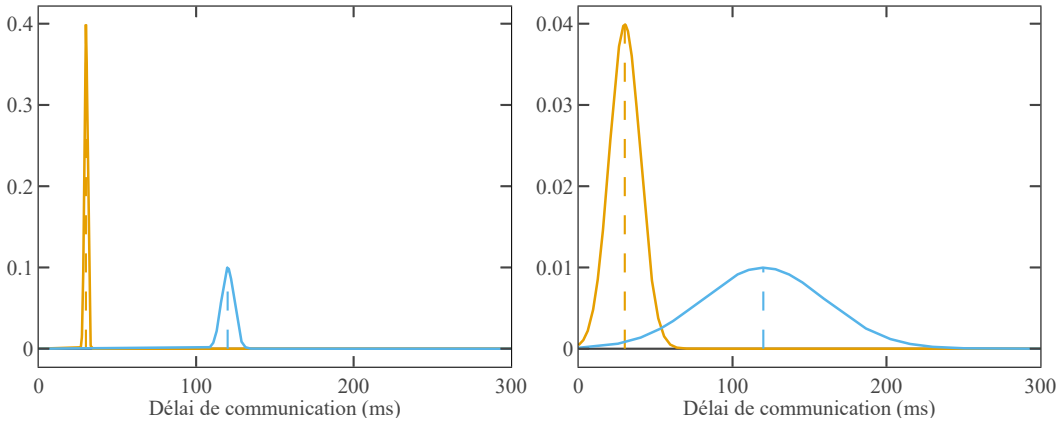
$$\overline{T_{AB}} = \alpha \cdot D_{AB} + \beta \tag{2.3}$$

On considère une relation affine entre la distance D_{AB} et le délai de communication moyen $\overline{T_{AB}}$. Le paramètre β représente le délai minimal et incompressible de communication, dû à l'étape de transmission et de propagation au travers du medium physique ainsi que le temps de traitement du message au niveau des équipements intermédiaires et des machines expéditrice et destinataire.

La partie en $\alpha \cdot D_{AB}$ du délai de communication traduit l'influence du nombre d'équipements par lequel passe le message. Si le réseau est encombré et qu'il ne possède pas une bande passante suffisante par rapport à sa charge, cela se traduit par un délai moyen plus important, donc un paramètre α plus élevé.

Nota — On sera amené à considérer un cas où tous les délais de communication sont égaux. Cela est possible avec notre modèle de délai de communication, il suffit de considérer le paramètre α à zéro.

Partie stochastique La partie stochastique de notre modèle cherche à représenter la gigue des paquets, c'est-à-dire la variation de la latence entre deux mêmes agents. Deux modèles stochastiques sont proposés et seront comparés en section 3.4.5 : un modèle gaussien et un modèle plus avancé prenant en compte les variations causées par le trafic internet ainsi que le routage des paquets.



(a) $\sigma = 0.1$ (b) $\sigma = 1.0$

FIGURE 2.3 – Densités de probabilité des délais de communication pour deux pairs d'agents différents, représentés en orange et en bleu. Ces densités sont représentées pour une valeur du paramètre d'écart-type $\sigma = 0.1$ en Fig. 2.3a et pour $\sigma = 1.0$ en Fig. 2.3b.

Modèle gaussien : On note σ_{AB} l'écart-type des délais de communication entre les agents A et B. Cet écart-type est proportionnel au délai moyen de communication T_{AB}. On définit

un paramètre unique σ reflétant les écarts-types des délais de communication. Le paramètre $\sigma \in [0, 1]$ est défini de manière à ce que, lorsque σ est maximal, alors l'amplitude des variations de délais soit du même ordre de grandeur que le délai moyen considéré, soit : $3\,\sigma_{AB} = \overline{T_{AB}}$. Ce principe est illustré en Figure 2.3, où sont représentées les densités de probabilité des délais entre deux paires d'agents différents, la première paire ayant un délai de communication moyen de 30 ms, la seconde de 120 ms. On remarque que les délais sont plus variés lorsque la valeur moyenne est plus grande.

$$t_{AB} \sim \mathcal{N}(0,\, \sigma_{AB}) = \mathcal{N}\left(0,\, \frac{\sigma}{3}\,\overline{T_{AB}}\right) \tag{2.4}$$

Une des limites du modèle gaussien réside dans la symétrie de sa fonction de distribution autour de la valeur moyenne. En effet, dans la réalité, les délais ne sont jamais beaucoup plus rapides que la valeur moyenne, on observe plutôt une longue queue de distribution faisant état des ralentissements cités plus haut. On considère tout de même ce modèle car il a l'avantage d'être très rapide à échantillonner, ce qui n'est pas négligeable étant donné que l'on échantillonne un délai de communication dès qu'un message doit être transmis. C'est un modèle de première approximation dont les limites de validité seront investiguées.

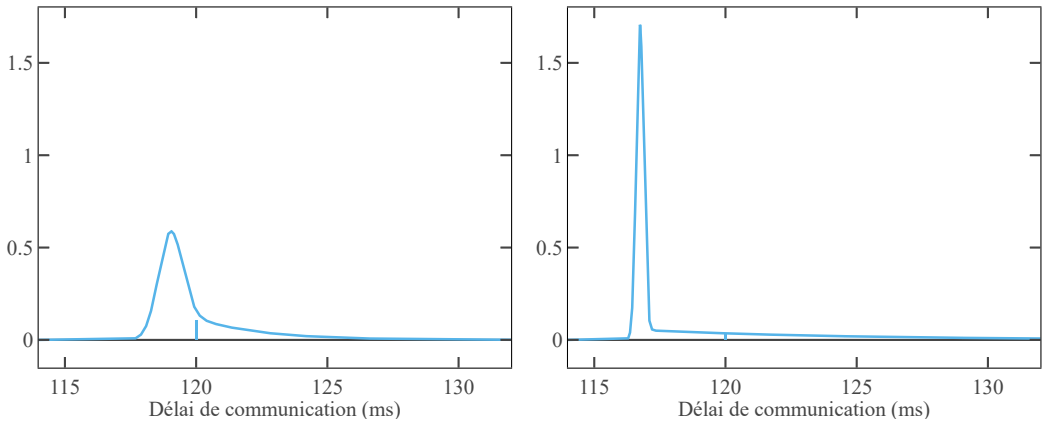
Modèle avancé : ce modèle a été utilisé dans [77, 78] et permet de prendre en compte les délais beaucoup plus longs que la moyenne qui se manifestent à une probabilité plus grande que pour une distribution gaussienne. Ce modèle comporte quatre paramètres qui prennent en compte les délais du trafic Internet ainsi que les délais de traitement des routeurs. C'est un modèle possédant une moyenne non nulle, contrairement au modèle gaussien présenté précédemment, cependant, les paramètres du modèle sont choisis de manière à ce que la moyenne des délais corresponde à la moyenne déterminée précédemment dans la partie déterministe $\overline{T_{AB}}$. La fonction de densité de probabilité $\phi(t)$ du délai total de communication est donnée par :

$$\phi(t) = p'\,\phi_2(t) + (1 - p')\,\phi_1(t) * \phi_2(t), \qquad t \geq 0 \tag{2.5}$$

où $\phi_1(t)$ est la densité de probabilité du délai lié au trafic Internet, $\phi_2(t)$ est la densité de probabilité des délais de traitement des routeurs et $\phi_1(t) * \phi_2(t) = \int_0^t \phi_2(u)\phi_1(t-u)du$. Ici, p' est la probabilité qu'un paquet ne rencontre pas de trafic Internet, ce paramètre est réglé à $p' = 0.58$ pour tous les liens de communication considérés, d'après la valeur donnée dans [77]. Le délai dû au trafic Internet est approximé par une densité de probabilité gaussienne avec une valeur moyenne de μ' et une variance de ${\sigma'}^2$. Le délai de traitement du routeur est modélisé par un processus de renouvellement alterné avec une période de fermeture exponentielle lorsque le trafic Internet est actif, avec la PDF $\phi_2(t) = \lambda'\,e^{-\lambda' t}$ où ${\lambda'}^{-1}$ représente la durée moyenne de la période de fermeture.

À des fins de comparaison, les paramètres $(\lambda',\, \mu',\, \sigma')$ sont réglés pour s'adapter à la valeur moyenne et à la variance du modèle gaussien de délais de communication pour chaque lien de communication :

$$\begin{aligned}(\lambda'_{AB},\, \mu'_{AB},\, \sigma'_{AB}) = \underset{(\lambda',\mu',\sigma')}{\arg\min}\ & \left(\alpha D_{AB} + \beta - \mathbb{E}\left[T_{AB}(\lambda',\, \mu',\, \sigma')\right]\right)^2 \\ & + \left(\sigma_{AB}^2 - \mathbb{V}\left[T_{AB}(\lambda',\, \mu',\, \sigma')\right]\right)^2\end{aligned} \tag{2.6}$$



(a) $\sigma = 0.1$ (b) $\sigma = 1.0$

FIGURE 2.4 – Densités de probabilité des délais de communication pour une paire d'agents dont les délais de communication sont en moyenne de 120 ms (on prend la même moyenne que pour la Figure 2.3). Ces densités sont représentées pour une valeur du paramètre d'écart-type $\sigma = 0.1$ en Fig. 2.4a et pour $\sigma = 1.0$ en Fig. 2.4b.

La fonction de distribution des délais de communication pour deux valeurs du paramètre σ est affichée en Figure 2.4. On peut y distinguer la queue de distribution à droite de la valeur moyenne qui s'étend sur des délais beaucoup plus élevés, et qui rend compte des variations de délais de communication avec plus de réalisme.

Nota

L'ordre de grandeur "réaliste" des délais de communication considérés lors des simulations est d'environ 50 à 100 ms. En effet, c'est l'ordre de grandeur lorsque l'on considère les délais des commandes ping entre différents serveurs mondiaux [a]. Ces délais sont réalistes lorsque l'on considère des communications longue distance. Or, on peut se trouver face à des configurations de réseaux plus favorables où tous les agents appartiennent au même réseau local et ont donc des délais plus courts. Les simulations effectuées dans les chapitres suivants utilisent des délais de communication élevés pour considérer le pire des cas.

a. `https://wondernetwork.com/pings`

Pour ce modèle, les délais de communication sont échantillonnés avec une méthode d'échantillonnage par transformation inverse, en utilisant la fonction de distribution cumulative donnée dans [77]. Comme le reste de la plateforme de simulation, les calculs sont codés en langage Julia et sont effectués sur une machine dotée d'un processeur i7-8850H. Il faut en moyenne 130 ns pour échantillonner un retard avec le modèle gaussien, alors qu'il faut en moyenne 30 μs pour échantillonner un retard avec le modèle avancé, ce qui rend la simulation environ 200 fois plus lente. Nous étudierons en section 3.4.5 dans quelle mesure le modèle gaussien peut être une alternative plus simple mais représentative du modèle avancé en ce qui concerne le temps de convergence de l'algorithme.

Type de modèle	Déterministe	Stochastique gaussien	Stochastique avancé
Moyenne	$\overline{T_{AB}} = \alpha \cdot D_{AB} + \beta$	$\overline{T_{AB}} = \alpha \cdot D_{AB} + \beta$	$\overline{T_{AB}} = \alpha \cdot D_{AB} + \beta$
Écart-type	0	$\sigma_{AB} = \frac{\sigma}{3}\overline{T_{AB}}$	$\sigma_{AB} = \frac{\sigma}{3}\overline{T_{AB}}$
Variable aléatoire	$\emptyset$	$t_{AB} \sim \mathcal{N}(0, \sigma_{AB})$	Densité de probabilité en (2.5)
Temps d'échantillonnage [a]	$\emptyset$	$\sim 100\,ns$	$\sim 30\,\mu s$

Tableau 2.1 – Synthèse des modèles de délais de communication

a. On considère que la partie moyenne est enregistrée en mémoire et n'a pas besoin d'être calculée à l'envoi de chaque message, contrairement à la partie stochastique.

Pertes de messages Le délai déterminé précédemment désigne la latence de transmission sans perte de message. On souhaite cependant aussi considérer l'effet de la perte d'un message et de sa retransmission sur le délai point à point d'un message. On définit une probabilité de perte de message $p_{pertes} \in [0,\, 1\,]$. À chaque nouvel envoi d'un message, si ce dernier est perdu, alors on considère qu'il est retransmis au bout de deux fois le délai de communication sans pertes déterminé précédemment. Ces délais s'ajoutent à chaque fois que le message est perdu, comme explicité dans l'Algorithme 2.1. Dans le cas où les délais sont stochastiques, alors on échantillonne le délai à chaque nouvel envoi au travers de la fonction *échantillonner_délai*().

Algorithme 2.1 Détermination du délai point à point T'_{AB} avec pertes

$T_{AB} \leftarrow$ *échantillonner_délai*()
$T'_{AB} \leftarrow T_{AB}$
tant que rand()<p_{pertes}
  $T_{AB} \leftarrow$ *échantillonner_délai*()
  $T'_{AB} \leftarrow T'_{AB} + 2 \cdot T_{AB}$
fin tant que

retourne T'_{AB}

Conclusion sur le modèle de délais de communication Ce modèle paramétrique permet de prendre en compte différents cas au niveau du réseau de communication : du cas de figure où tous les agents appartiennent au même sous-réseau et où les communications ne subissent quasiment pas de pertes et des délais homogènes, au cas de figure où chaque agent se trouve dans un sous-réseau différent, où les délais de communication sont très variés et qu'ils subissent des ralentissements dus à la charge du réseau de communication à l'instant considéré. C'est un modèle délibérément très simplifié par rapport à ce que proposent certains auteurs [13]-[18] qui simulent finement le réseau de communication. Il a donc l'avantage d'être simple à interpréter et extrêmement rapide à calculer dans le cadre de la simulation décrite en section 2.3, d'être générique et facile à identifier donc de permettre une généricité des résultats.

2.2 Modélisation des calculs

La simulation des algorithmes considérés dans ce manuscrit prend en compte les temps de calculs au même niveau que les délais de communication. Ces temps de calculs diffèrent selon la taille et le type du problème à résoudre ainsi que la méthode utilisée pour le résoudre. Ils dépendent aussi des caractéristiques de la machine de calcul. Sachant que les capacités du matériel informatique est en constante progression à des prix toujours plus abordables, les temps de calcul constatés aujourd'hui ne seront pas les mêmes que ceux constatés d'ici quelques années. Selon le type d'agents considérés, on peut aussi faire le choix économique de les équiper de machines à faibles capacités de calcul au détriment de la rapidité. Les deux situations extrêmes que l'on peut rencontrer sont d'un côté le serveur de calculs dédié, par exemple un serveur appartenant à RTE qui sert exclusivement pour effectuer des calculs de power flow du réseau électrique, et de l'autre côté les machines issues de l'Internet of Things (IoT) d'un agent pair à pair à faibles capacités de calculs. Malgré ces disparités, on utilise un modèle de temps de calcul basé sur des temps réels mesurés sur une machine possédant un processeur Intel Core i7-8850 et une mémoire vive de 16 Go.

Selon le type de problème à résoudre, les différences de temps de calcul sont de plusieurs ordres de grandeur. Dans le cadre de notre application, deux types de problèmes sont considérés :

— la minimisation d'une fonction quadratique sous contraintes linéaires,
— la minimisation d'une fonction non linéaire sous contraintes non linéaires.

Dans le premier cas, que l'on retrouve principalement dans le cas du problème de marché pair à pair décentralisé en Chapitre 3, les temps de calculs sont de l'ordre de la dizaine de microsecondes. Ces délais sont aussi très faibles comparés à l'ordre de grandeur des délais de communication considérés. Ainsi, pour les calculs de marché décentralisé, on fait le choix de négliger les temps de calcul.

Dans le second cas, associé aux calculs d'Optimal Power Flow présentés au Chapitre 4, les temps de calculs sont plutôt de l'ordre de la dizaine de millisecondes, ce qui représente un ordre de grandeur trois fois plus élevé que pour les calculs de marché décentralisé, mais qui arrive à la hauteur des délais de communication. Ainsi, ces délais ne seront pas négligés lorsque l'on considère des calculs d'Optimal Power Flow .

Modèle affine On enregistre le temps que prend la résolution d'un OPF [3] sur plusieurs cas test rendus disponibles par la bibliothèque `MATPOWER` [94], où le nombre de variables [4] varie entre 3 et 1500. On obtient alors les résultats présentés en Figure 2.5. Chaque cas test correspond à un nombre de variables donné. Les variations des temps de calculs sur le même cas test proviennent du fait que le système d'exploitation utilisé (ici, Windows 10) rende le temps de calcul non déterministe. Les fluctuations non monotones des temps de calcul mesurés s'expliquent par le fait que les cas test considérés diffèrent : ils peuvent avoir plus ou moins de lignes à prendre en compte, et peuvent être plus ou moins contraints.

3. Chaque cas test est évalué plusieurs fois grâce au package Julia : BenchmarkTools.
4. Le nombre de variables prend en compte les tensions complexes, les puissances complexes dans les lignes ainsi que les puissances complexes injectées par les consomacteurs.

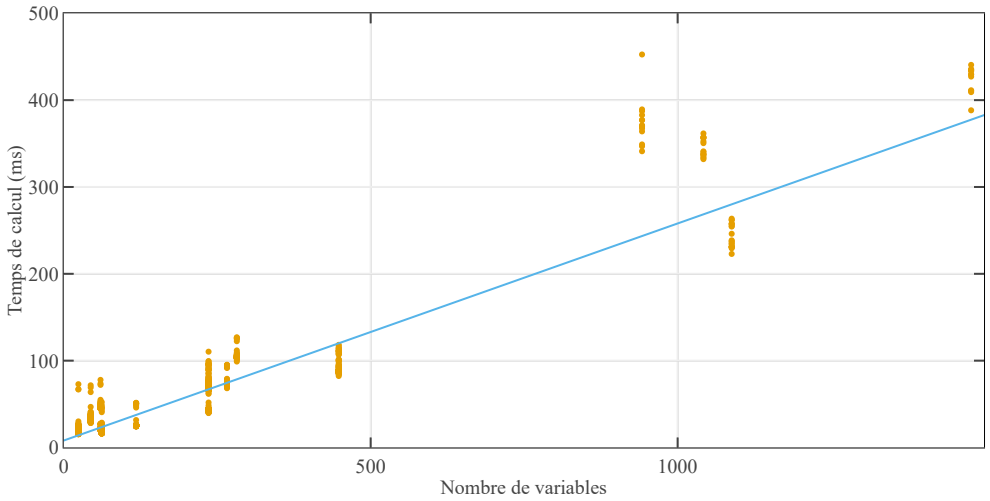


FIGURE 2.5 – Temps de calcul d'Optimal Power Flow en fonction du nombre de variables du problème. La fonction affine retrouvée par régression linéaire est affichée en bleu.

On suppose une relation affine entre le temps de calcul et le nombre de variables du problème. Ainsi, à l'aide d'une régression linéaire, on trouve les coefficients de la fonction affine donnant le temps de calcul en fonction du nombre de variables du problème, donnés en (2.7).

$$\Delta_{comp}(k) = 0.25 \cdot Nvar(k) + 8 \quad \text{(ms)} \tag{2.7}$$

Comme illustré en Figure 2.5, le modèle affine présente des écarts avec les délais réels constatés. Cela n'est pas dérangeant étant donné que l'on s'intéresse surtout à l'ordre de grandeur des temps de calculs et au ratio entre les temps de calcul et les délais de communication.

Synthèse des modèles de temps de calcul

Type de problème	**Marché**	**OPF**
Fonction objectif	Quadratique/convexe	Non linéaire/non convexe
Contraintes	Linéaires	Non linéaires/non convexe
Modèle de temps de calcul	Temps de calcul négligé	Modèle affine (2.7)

TABLEAU 2.2 – Synthèse des modèles de temps de calcul

2.3 Plateforme de simulation à événements discrets

Le terme "plateforme de simulation" englobe l'algorithme, l'environnement logiciel qui centralise les modèles présentés précédemment et le cas test à résoudre ainsi que les paramètres de l'algorithme considéré. La plateforme de simulation doit pouvoir simuler les différents échanges de messages entre les agents décentralisés, ainsi que réaliser les calculs locaux nécessaires à la résolution du problème considéré. Elle doit être flexible et reconfigurable selon les différents modèles et cas test. Pour cela, on a choisi pour notre plateforme de simulation le package `SimJulia` [95], un outil de simulation d'événements discrets basé sur le langage de programmation Julia. Ce package permet de simuler séparément tous les agents de calcul, ainsi que les messages qu'ils s'envoient entre eux. Le programme réalisé doit aussi permettre de faire le suivi de certaines grandeurs importantes telles que le nombre de messages envoyés et les valeurs des différentes variables au cours de la simulation.

La Figure 2.6 présente les différentes actions que la plateforme doit être capable de simuler du point de vue de chaque agent de l'algorithme décentralisé.

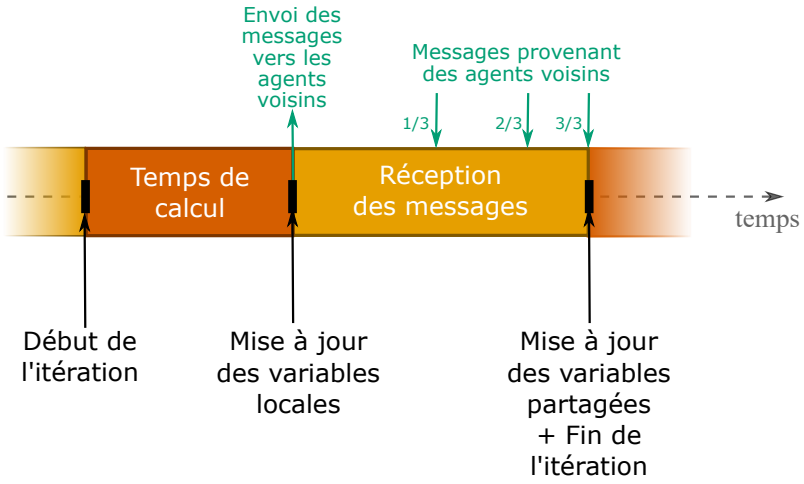


FIGURE 2.6 – Illustration des étapes qui doivent être prises en compte dans la simulation d'un agent local. Dans l'exemple illustré, l'agent envoie des messages à ses voisins après une étape de calcul qui dure un certain délai, puis il entre dans une phase d'attente des messages en provenance de trois de ses voisins. Au bout du troisième message reçu, il met à jour sa copie des variables partagées et peut procéder à l'itération suivante.

La simulation à événements discrets est utilisée en opposition à la simulation continue, où le temps est découpé en tranches égales et où le système est observé à chaque tranche. Dans une simulation à événements discrets, le système est mis à jour à chaque événement, peu importe le temps passé entre l'événement considéré et celui qui le précédait, ce qui en fait un type de simulation parfaitement adaptée à l'envoi/la réception de messages et des délais de calculs, qui sont considérés comme des événements discrets. Dans l'exemple donné en Figure 2.6, les différents événements considérés dans la simulation sont les suivants.

— Le lancement des calculs : on active une pause de l'agent local pendant un temps égal au temps de calcul [5] donné par le modèle de la section 2.2.

5. On distingue bien le temps réel du calcul sur la machine du temps simulé sur la plateforme qui provient du modèle affine décrit plus haut.

— La fin des calculs : on met à jour les variables locales et on envoie des messages contenant ces mises à jour aux agents voisins, la réception de ces messages prend effet au bout d'un certain délai déterminé par le modèle de délais de communication présenté en section 2.1.
— La réception de chaque message : à chaque message reçu, le compteur de messages est incrémenté et comparé au nombre de messages attendu, tant que l'agent local a reçu un nombre de messages inférieur à celui attendu, il reste dans une étape d'attente.
— La réception du dernier message entraîne la fin de l'étape d'attente, la mise à jour des valeurs locales des variables partagées par l'algorithme, ainsi que la fin de l'itération courante.

2.3.1 Structure du programme de simulation

Le package `SimJulia` propose différents éléments de simulation dont :
— des *processus*, ce sont des fonctions qui simulent le comportement d'acteurs dans la simulation, tels que les agents de calcul ;
— des *stores*, ce sont des types de conteneurs à capacité limitée ou illimitée, on les utilise dans notre cas pour stocker les messages reçus en attente de traitement.

Les processus se présentent comme des fonctions classiques, qui peuvent effectuer des actions propres à l'algorithme considéré, mais qui peuvent aussi engendrer des événements ainsi que d'autres processus dont elles sont dépendantes. On associe à chaque agent local son propre processus, ainsi que son propre store qui sert de boîte de réception des messages reçus. La Figure 2.7 est une représentation temporelle des processus associés à deux agents voisins A et B.

Pour simuler le temps de calcul d'un agent, le processus **Agent local A** engendre un processus fille **Temps de calcul A** équivalent à un *timeout* d'une durée déterminée par le modèle de temps de calcul présenté en section 2.2. Tant que le processus **Temps de calcul A** n'est pas terminé, le processus mère **Agent local A** est bloqué et ne peut pas effectuer le reste de son programme. De cette manière on est capable de simuler le temps de calcul de l'agent local A dans la plateforme de simulation.

Pour ce qui est de l'étape d'attente des messages qui suit l'étape de calculs, de la même manière, le processus **Agent local A** engendre un processus fille **Attente de messages A** qui est assimilé à un *timeout* d'une durée très longue par rapport aux délais de communication attendus [6]. En parallèle, le processus associé à un agent voisin de l'agent A **Agent local B** génère des processus filles d'envoi de message, dont un processus **Envoi de B vers A** qui s'active au bout d'un certain délai de communication estimé grâce au modèle de la section 2.1. Lorsque ce processus est activé, il ajoute au store **Boîte de réception A** le message à délivrer, et compare le nombre de messages dans cette boîte de réception au nombre de messages attendus par l'agent A. Si le nombre de messages effectivement reçus est supérieur ou égal au nombre de messages attendus, alors il interrompt le processus **Attente de messages A**. Ainsi, le processus mère **Agent local A** est débloqué et peut passer à la suite de son programme.

À noter que les processus d'envoi de messages ne bloquent pas l'avancement du processus mère qui les a créés. Ils sont constitués d'une partie timeout pour simuler le délai de communication, puis de la partie d'ajout du message dans le store de l'agent destinataire et

6. On utilise ce timeout pour éviter de bloquer l'avancement d'un agent dans le cas où le nombre de messages reçus reste toujours plus petit que le nombre de messages attendus. Cela peut arriver lorsque deux agents s'auto-bloquent c'est-à-dire qu'ils attendent chacun un message de l'autre qui ne peut arriver que si l'un d'entre eux envoie le premier message.

de comparaison du nombre de messages.

Cette organisation du code est très flexible et permet de prendre en compte autant les algorithmes synchrones que les algorithmes asynchrones.

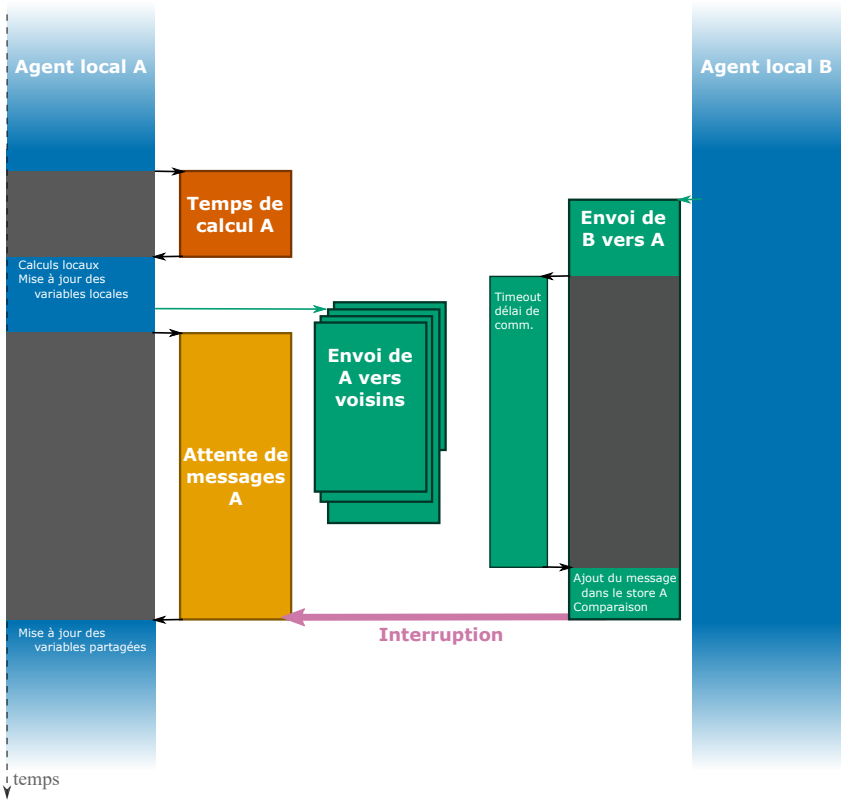


FIGURE 2.7 – Représentation temporelle non exhaustive des différents processus pour une simulation de deux agents A et B. Les processus associés à l'agent local B ne sont pas tous représentés pour simplifier la figure.

2.3.2 Monitoring des valeurs d'intérêt

Le suivi des valeurs d'intérêt est réalisé par un monitoring à pas de temps fixe. Il est effectué au travers d'un processus qui s'active à un pas de temps donné et effectue une copie de l'état de tous les agents participant à l'algorithme décentralisé. Parmi les valeurs enregistrées :

- les valeurs des variables locales, primales et duales,
- les résidus locaux, primal et dual,
- l'écart entre les valeurs des variables et les valeurs optimales attendues [7],
- le nombre d'itérations locales,
- le nombre de messages échangés jusque là,
- le nombre de messages présents dans chaque boîte au lettre.

7. Les valeurs optimales attendues proviennent du résultat obtenu via la résolution centralisée du problème considéré.

Les variations temporelles de ces valeurs sont enregistrées à la fin de la simulation dans un fichier au format .JLD (JuliaData) pour un traitement ultérieur des données.

2.4 Synthèse de la plateforme de simulation

On présente un schéma de synthèse de la plateforme de simulation en Figure 2.8. On s'intéresse en particulier au comportement des algorithmes étudiés face à différentes configurations des modèles de délais de communication et de calcul. Étant donné que l'on a fait le choix de ne pas fixer de topologie du réseau de communication, ni de protocoles de communication, il est nécessaire de tester de nombreux cas de figure pour tester le déroulement des algorithmes en présence de variations de délais ainsi que de pertes de messages.

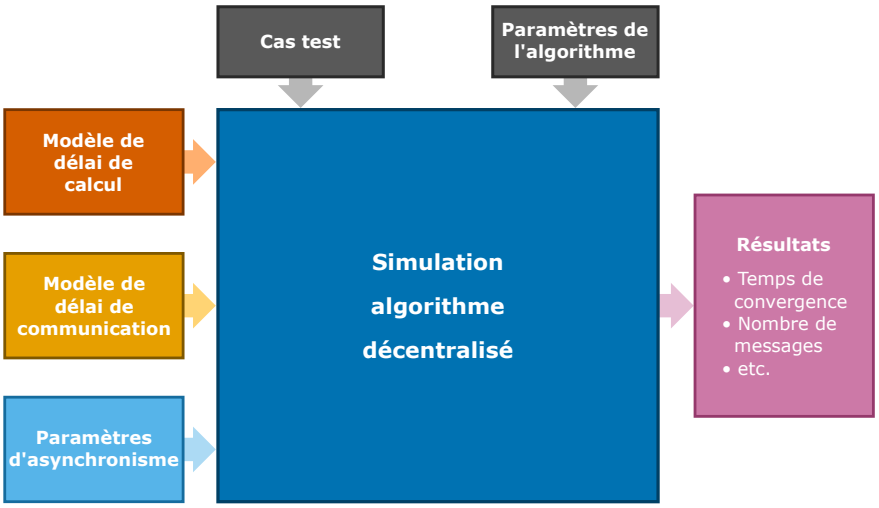


FIGURE 2.8 – Schéma de la plateforme de simulation, avec en entrée les modèles paramétriques de délais de communication et de calcul et les paramètres d'asynchronisme, et en sortie les données enregistrées au long de la simulation ainsi que les résultats des données traitées.

Les paramètres d'asynchronisme, présentés plus en détails dans les chapitres qui suivent, ont aussi un impact sur le comportement de ces algorithmes, nous montrerons qu'un algorithme asynchrone est plus robuste aux variations de communication qu'un algorithme synchrone. Les paramètres de l'algorithme décentralisé, comme le facteur de pénalité ρ utilisé dans la méthode ADMM, sont fixes, et on utilise un cas test donné pour chaque algorithme étudié. Ces cas test sont issus de la librairie `MATPOWER`, et sont systématiquement présentés en Annexe lorsqu'ils sont utilisés.

Enfin, en sortie de simulation, on récupère les données temporelles enregistrées au cours de la simulation, et on en déduit après traitement des résultats significatifs tels que le temps de convergence, le nombre total de messages échangés, mais aussi des mesures de l'écart entre la solution optimale, obtenue par résolution du problème centralisé, et la solution finale à l'instant de convergence.

La plateforme de simulation ainsi que les différents modèles présentés dans ce chapitre sont systématiquement utilisés dans les études d'algorithmes décentralisés présentés dans la

suite de ce manuscrit. On commence par étudier un algorithme de marché pair à pair au Chapitre 3, puis des algorithmes décentralisés d'Optimal Power Flow au Chapitre 4, et enfin au Chapitre 5 un algorithme hybride : un marché pair à pair "endogène" c'est-à-dire prenant directement en compte des contraintes physiques du réseau électrique.

Chapitre 3

Marché de l'électricité pair à pair asynchrone

Sommaire

Nous avons identifié au chapitre introductif que le besoin en communication est amené à augmenter à mesure que la topologie du réseau électrique se modifie, passant d'une organisation centralisée à une organisation décentralisée. Aussi, la gestion du réseau électrique est amenée à reposer fortement sur les échanges d'information entre les différents agents, et donc potentiellement sensible aux différents aléas de communication. Dans ce chapitre, nous nous intéressons aux impacts des aléas de communication sur différents algorithmes de résolution d'un marché de l'électricité. Après avoir fait quelques rappels sur le marché de l'électricité et identifié les limites des algorithmes centralisés, nous nous pencherons sur des algorithmes décentralisés qui ont l'avantage de pouvoir passer à l'échelle et de mieux correspondre à l'organisation future du réseau électrique. Les contributions de ce chapitre résident dans la version asynchrone d'un de ces algorithmes décentralisés, qui permet de rendre plus robuste les temps de convergence en présence d'aléas de communication, ainsi que dans l'implémentation expérimentale du même algorithme asynchrone. L'algorithme de marché de l'électricité étudié en détails est un algorithme de **marché pair à pair** issu de [45] où les agents du marché communiquent directement les uns avec les autres pour se mettre d'accord sur des échanges de puissance et où il n'y a pas besoin de gestionnaire de marché central pour atteindre le résultat optimal.

3.1 Marché de l'électricité

Les différents marchés de l'électricité ont été présentés en Figure 1.6 du chapitre 1. Leur but commun est d'amener les producteurs et les consommateurs à se mettre d'accord sur les volumes d'électricité échangés à un moment donné dans le futur, appelé l'instant de livraison. Là où le marché à terme consiste en des transactions de gros volumes pour une livraison considérée de l'ordre de l'année, les marchés journaliers et infra-journaliers

consistent à affiner les échanges d'électricité pour mieux prendre en compte les prévisions de consommation et de productions, notamment renouvelables. Ces marchés sont considérés en général sur un pas de temps horaire. Les marchés journaliers s'effectuent le jour d'avant la livraison tandis que les marchés infra-journaliers s'effectuent jusqu'à 15 minutes avant la livraison. Les acteurs du marché peuvent être des producteurs, des fournisseurs d'électricité, ou bien des responsables d'équilibre qui sont engagés auprès du gestionnaire de réseaux et qui doivent s'assurer de l'équilibre du réseau électrique sur un périmètre géographique ou contractuel pour le maintenir en fonctionnement. Le but premier du marché de l'électricité est de maximiser le contentement collectif en trouvant le prix du marché qui satisfasse les producteurs et les consommateurs. L'égalité entre l'offre et la demande en électricité est obtenue par construction du mécanisme de résolution de marché, ce qui, en théorie et si tous les acteurs respectent effectivement leur contrat, garantit l'équilibre de puissance sur le réseau électrique. On parle dans ce cas d'acteurs rationnels non stratégiques. On se place de plus dans le cadre où chaque agent possède un poids marginal sur le marché, c'est-à-dire qu'aucun agent n'est assez grand pour influencer à lui seul le prix du marché.

Délimitations du marché Les acteurs constitutifs d'un marché de l'électricité sont en général groupés géographiquement, que ce soit sur le plan national, régional ou local. Les agents d'une même région géographique participent à un unique marché "local" dans lequel on recherche l'équilibre des puissances. Cependant, tant que les agents sont interconnectés physiquement, il est possible qu'il y ait des échanges entre les différents marchés. Cela permet de mutualiser les moyens de production et de pallier les fluctuations de la demande sur le réseau. Ainsi, ces zones d'échange sont traitées dans le marché local comme un agent à part entière pouvant participer au marché.

Ainsi, lorsque l'on considère des marchés à une granularité plus fine qu'à l'échelle d'un pays, il est tout de même possible de prendre en compte des échanges inter-zones. La solution globale diffère du cas où l'on considère l'intégralité des agents réunis en un seul marché, mais encourage la consommation locale de l'énergie. Avec l'avènement des énergies renouvelables distribuées, on considère de plus en plus des marchés locaux où les consommateurs sont directement acteurs du marché et ne sont plus agglomérés derrière un fournisseur d'électricité. Ce changement de paradigme amène une augmentation significative du nombre de consomacteurs participant aux marchés de l'électricité.

Que l'on parle du marché global (européen) ou bien de marchés locaux, le mécanisme de résolution de marché peut être centralisé, c'est-à-dire résolu par un gestionnaire de marché, global ou local, ou bien décentralisé c'est-à-dire dont la résolution n'est pas menée par un unique agent mais répartie sur tous les agents à la fois.

Mécanisme de marché Différents modèles et structures de marché sont disponibles pour effectuer la résolution du marché de l'électricité. On considère dans un premier temps un marché *pool based* dans lequel le prix du marché est déduit à partir des offres et des demandes des consomacteurs participant au marché. Un producteur annonce par exemple qu'il est prêt à vendre n kWh pour y €/kWh, un consommateur annonce qu'il est prêt à payer m kWh à z €/kWh. Après avoir collecté toutes les offres et demandes, le prix du marché est déterminé au travers du mécanisme de *merit-order*, et de ce prix découle la répartition des puissances. Les producteurs trop chers par rapport au prix du marché ne sont pas appelés pour participer à la production électrique. Pour les producteurs appelés, ces derniers sont rémunérés à un prix supérieur au prix auquel ils étaient prêts à vendre. Le même mécanisme s'applique de manière symétrique pour les consommateurs. Ainsi, le contentement collectif est maximisé. Les producteurs qui vendaient leur électricité à un prix supérieur au prix du marché sont

donc exclus des échanges, de même que pour les consommateurs qui demandaient à payer un prix inférieur. Dans ce chapitre, on considère la version continue de ce problème de marché, où les offres et demandes ne sont plus définies de manière discrète, mais sont représentées par des fonctions coût continues à minimiser. Traditionnellement, on considère ces fonctions coût sous une forme quadratique : $f(p) = a \cdot p^2 + b \cdot p + c$ qui a l'avantage d'être convexe et facilite le processus d'optimisation, et qui modélise approximativement les coûts de fonctionnement réels.

L'approche privilégiée dans ce manuscrit est celle de l'optimisation sous contraintes. Il existe plusieurs méthodes d'optimisation des marchés, portant notamment sur des techniques de programmation linéaire (ou *linear programming* LP), de programmation linéaire en nombres entiers (ou *mixed integer linear programming* MILP) [39], de programmation basée sur la méthode ADMM *alternating directions methods of multipliers* [96] ou la méthode Consensus+Innovation [44], ou bien de programmation non linéaire (ou *non linear programming* NLP). Nous serons amenés à étudier des algorithmes basés sur la méthode ADMM, cette méthode permettant de converger rapidement vers la solution finale, possédant une preuve de convergence dans le cas convexe et pouvant facilement s'appliquer à des problèmes non convexes [1]. Le but final du mécanisme de résolution de marché est de minimiser la somme des fonctions coûts des différents agents du marché tout en suivant la contrainte d'équilibre des puissances sur le réseau. À noter ici que les contraintes de faisabilité physique des solutions optimales obtenues ne sont pas prises en compte dans le processus d'optimisation du marché en lui-même, mais sont prises en compte a posteriori par le gestionnaire de réseau, de manière exogène. La formulation du problème est donné en (3.1).

Formulation du problème de marché de l'électricité

$$\min \sum_{i \in \mathcal{C}} f_i(P_i) \tag{3.1a}$$

$$\text{selon } P_i \qquad i \in \mathcal{C}$$

$$\text{tel que } \sum_{i \in \mathcal{C}} P_i = 0 \tag{3.1b}$$

$$\underline{P_i} \leq P_i \leq \overline{P_i} \qquad i \in \mathcal{C} \tag{3.1c}$$

avec $f_i(\cdot)$ la fonction coût du consomacteur i, $\underline{P_i}$ et $\overline{P_i}$ respectivement la limite basse et haute de la puissance active injectée ou consommée par le consomacteur i.

- La puissance produite par un producteur et injectée sur le réseau électrique est considérée positive $P_i > 0$.
- La puissance consommée par un consommateur et soutirée du réseau électrique est considérée négative $P_i < 0$.
- Il peut y avoir des consomacteurs qui, selon le prix du marché et leur état local, peuvent décider à tout moment soit de soutirer de la puissance électrique, soit d'en injecter.

1. Le problème de marché est bien un problème convexe mais on utilisera aussi la méthode ADMM pour résoudre le problème d'Optimal Power Flow qui lui est non convexe.

L'équation (3.1b) désigne la contrainte d'équilibre de puissance sur l'ensemble du réseau électrique.

Nota

On ne considère dans ce problème que la puissance active et pas la puissance réactive. En effet, cette dernière a un impact sur d'autres variables du réseau électrique non prises en compte dans ce problème de marché telles que le niveau de tension ainsi que son déphasage. On se penchera plus en détail sur ces contraintes physiques dans le Chapitre 4 traitant du problème d'Optimal Power Flow .

3.2 Résolution centralisée

Résolution centralisée synchrone L'approche classique de la résolution du marché de l'électricité est l'approche centralisée où un agent unique, le *gestionnaire de marché*, regroupe les offres et les demandes des différents acteurs du marché pour trouver la répartition optimale de l'énergie. Sa mise en œuvre est rendue possible par le petit nombre d'acteurs participant au marché, notamment dû à l'agrégation des consommateurs d'un territoire au travers des fournisseurs d'électricité.

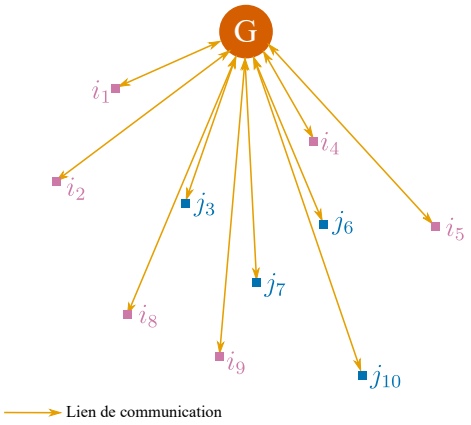


FIGURE 3.1 – Schéma exemple d'un réseau dont la résolution de marché est effectuée par un gestionnaire de marché central. Les flèches jaunes représentent les liens de communication entre les agents du marché et le gestionnaire. Les agents en rose notés i sont des consommateurs et les agents en bleu notés j sont des producteurs. Les lignes de connexion physiques ne sont pas représentées dans ce schéma, étant donné qu'elles ne sont pas prises en compte dans le problème de marché considéré.

D'un pas de temps à l'autre, ce gestionnaire de marché doit collecter l'intégralité des données des agents, c'est-à-dire les coefficients de leur fonction coût ainsi que leurs limites de puissance, avant de réaliser sa résolution. La Figure 3.1 illustre la centralisation des données collectées entre les agents du marché et le gestionnaire. Une fois que le problème d'optimisation a été résolu en utilisant une méthode d'optimisation convexe sous contrainte comme par exemple une méthode de points intérieurs, le gestionnaire communique à son tour aux différents agents du marché le résultat obtenu, c'est-à-dire le prix du marché et la quantité de puissance qu'ils devront injecter ou soutirer du réseau à la livraison. Le fonctionnement du gestionnaire est résumé dans l'Algorithme 3.1, où f_i désigne la fonction

coût de l'agent i, $\underline{P_i}$, $\overline{P_i}$ représentent respectivement la limite de puissance basse et la limite de puissance haute de l'agent i. $\mathcal{C}$ désigne l'ensemble des agents participants au marché.

Algorithme 3.1 Marché centralisé synchrone, agent central

tant que tous les agents n'ont pas communiqué leurs données
  Attendre les données des agents du marché f_i, $\underline{P_i}$, $\overline{P_i}$, $\quad i \in \mathcal{C}$
fin tant que
Effectuer la résolution du marché en résolvant le problème (3.1)
Envoyer les résultats du marché aux agents $i \in \mathcal{C}$

La Figure 3.2 représente le déroulement temporel de la résolution du marché centralisée et son placement en amont de l'instant de livraison considéré par le marché. Traditionnellement, l'avance pour résoudre le marché est très conservatrice et assure l'implication de l'intégralité des agents souhaitant participer au marché, jusqu'à 15 minutes avant l'instant T. Il serait souhaitable de réduire ce temps d'avance ne serait-ce que pour diminuer au mieux les incertitudes des énergies renouvelables. On peut imaginer que dans le cadre d'un marché local, les incertitudes sont plus grandes à cause du faible taux de foisonnement des consommations et des productions incertaines.

Nota

On considère ici un marché *pool* qui nécessite les données de l'intégralité des agents participant au marché, contrairement à un marché *online matching* [97] où des contrats sont négociés de manière bilatérale au fur et à mesure des offres et des demandes. Ces marchés peuvent aussi aller jusqu'au quart d'heure avant la livraison mais ils ne permettent pas d'arriver à une solution optimale, contrairement au marché *pool*.

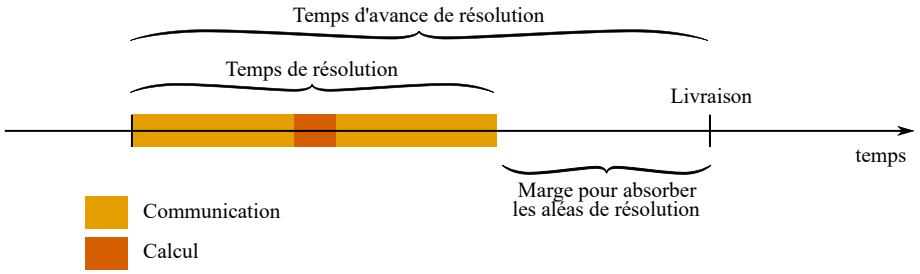


FIGURE 3.2 – Représentation sur l'échelle du temps du déroulement de la résolution de marché centralisée. La résolution, comprenant deux phases de communication et une phase de calcul, est déclenchée en amont de l'instant de livraison considéré. Le temps d'avance de la résolution est traditionnellement très conservateur pour donner le temps à la résolution de se faire, même en présence d'aléas.

Étant donné que l'opérateur de marché doit attendre l'intégralité des données des agents avant de réaliser son calcul en version synchrone, le temps d'attente est alors déterminé par le plus long délai de communication entre un agent du marché et l'opérateur du marché. Cette manière de faire fonctionne dans le cas où tous les délais de communication sont du même ordre de grandeur. Or, le réseau de communication peut être soumis à des aléas

qui amènent à des ralentissement importants, notamment si les messages se perdent en chemin et doivent être alors renvoyés. Les délais de communication [2], calculés en utilisant le modèle de la section 2.1, sont représentés en fonction du pourcentage de perte de messages en Figure 3.3 pour deux cas test : un cas à 100 agents et un cas à 20000 agents. Les coordonnées nécessaires pour calculer la distance "D_{AB}" et donc les délais de communication sont tirées aléatoirement dans l'intervalle $[0, 1]$, et on prend comme paramètres de délais de communication $\alpha = 50$ et $\beta = 10$, ce qui correspond à un délai moyen d'environ 35 ms. L'attente due à la communication augmente avec le taux de pertes de messages et aussi avec le nombre d'agents participant au marché.

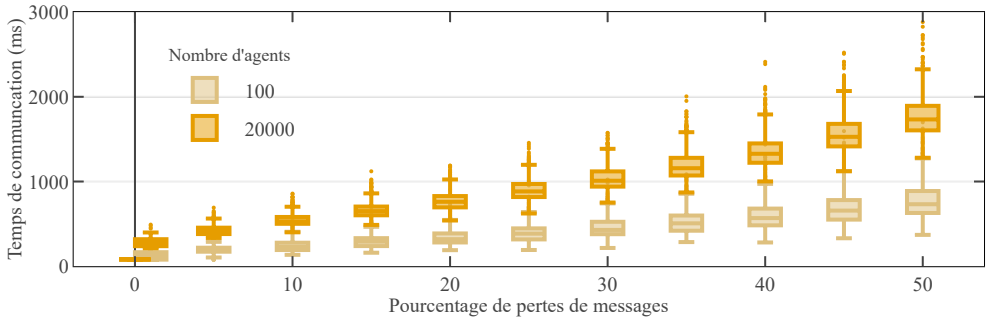


FIGURE 3.3 – Temps de communication entre la totalité des agents du marché et le gestionnaire de marché, en fonction du taux de pertes de messages pour deux cas test : 100 agents et 20000 agents.

Nota

Dans l'exemple utilisé, notre modèle de délais de communication utilise des délais qui sont tous de l'ordre de la dizaine de millisecondes. Dans le cas où ne serait-ce qu'un des délais de communication soit d'un ordre de grandeur supérieur, alors son impact sur les délais de communication globaux serait beaucoup plus visible.

Pour un problème de marché de l'électricité centralisé synchrone, le temps de calcul dépend du nombre d'agents du marché. Le problème d'optimisation est un problème de minimisation d'une fonction convexe sous contraintes linéaires, qui est facilement résolu par des méthodes de descente de gradient par exemple. On utilise pour nos calculs le solveur `OSQP` [98] qui repose sur la méthode ADMM et dont l'implémentation en C rend la résolution très rapide. Les temps de calcul du marché centralisé résolu avec le solveur OSQP et appliqué sur divers cas test générés aléatoirement sont affichés en Figure 3.4. Chaque cas test a été évalué un grand nombre de fois sur un processeur i7-8850H @2.60 GHz en utilisant le package `BenchmarkTools.jl`. On observe que les temps de calcul augmentent de manière polynomiale avec le nombre d'agents participant au marché, on trouve par régression linéaire sur la moyenne en échelle log-log une augmentation du temps de calcul en $N^{1.83}$. Les ressources computationnelles nécessaires au calcul du marché augmentent aussi avec le nombre d'agents. Pour le cas à 20000 agents, la mémoire vive nécessaire au calcul était d'environ 6 Gb.

2. On rappelle que lorsqu'un paquet est perdu sur le réseau de communication, il est renvoyé au bout d'un temps égal au temps d'aller-retour d'un paquet. La durée de communication finale est alors de 3 fois la durée de communication sans perte.

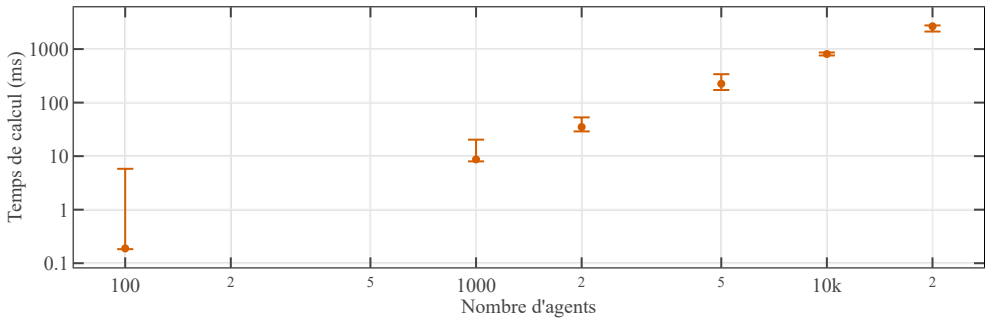


FIGURE 3.4 – Temps de calcul du marché centralisé en fonction du nombre d'agents participant au marché. Les valeurs minimales et maximales sont représentées par les barres d'erreur.

Résolution centralisée asynchrone La réduction du temps d'avance de résolution est limitée par le temps de résolution du marché centralisé, composé de deux phases : une phase de communication et une phase de calcul. La phase de calcul est incompressible car le nombre d'agents à considérer n'est pas modifiable. La phase de communication est aussi incompressible si l'on doit attendre l'intégralité des messages, et ne dépend que de l'état du réseau de communication. On propose alors d'effectuer les calculs de marché au bout d'un nombre arbitraire de messages reçus N_{mes}, de manière à diminuer le temps de résolution malgré les délais. Le fait de ne pas prendre en compte tous les messages justifie le terme de résolution asynchrone. Plusieurs possibilités s'offrent alors.

— Retirer du marché les agents n'ayant pas pu communiquer à temps leur participation. Cette option a comme inconvénient que les productions ou consommations physiques d'électricité continueront malgré le fait qu'elles ne soient pas prises en compte dans l'équilibre du marché, résultant en un déséquilibre des puissances.

— Garder ces agents dans le marché et estimer leurs données en fonction de leurs précédentes participations. Cette option présente aussi l'inconvénient de ne pas prendre en compte avec exactitude les données de ces agents, mais permet de réduire le déséquilibre cité précédemment en comptant sur l'autocorrélation temporelle des productions renouvelables ainsi que des consommations.

On considère le deuxième cas de figure, dont le fonctionnement est décrit dans l'Algorithme 3.2, où f_i^0, $\underline{P_i}^0$, $\overline{P_i}^0$ désignent les données relatives à l'agent i de l'itération précédente du marché [3].

Algorithme 3.2 Marché centralisé asynchrone, agent central

tant que le nombre de messages reçu est inférieur à N_{mes}
  Attendre les données des agents du marché f_i, $\underline{P_i}$, $\overline{P_i}$, $\quad i \in \mathcal{C}$
fin tant que
f_i, $\underline{P_i}$, $\overline{P_i} \leftarrow f_i^0$, $\underline{P_i}^0$, $\overline{P_i}^0$ pour les agents i dont les messages n'ont pas été reçus à temps
Effectuer la résolution du marché en résolvant le problème (3.1)
Envoyer les résultats du marché aux agents $i \in \mathcal{C}$

Cette résolution mène à un déséquilibre sur la puissance globale à produire à la livraison, qui sera compensée par les réserves d'énergie et les mécanismes de régulation en temps réel

3. Cette configuration suppose la résolution séquentielle de plusieurs marchés sur des pas de temps consécutifs.

mis en place sur le réseau. À cela s'ajoute aussi une sous optimalité de la solution trouvée, qui se manifeste sous la forme des écarts entre les puissances injectées de chaque agent et les puissances de la solution optimale trouvée dans le cas où le gestionnaire de marché reçoit toutes les informations nécessaires. L'écart relatif en terme de puissances injectées de la résolution centralisée asynchrone est définie en (3.2).

$$\frac{\sum_{i\in\mathcal{C}} |p_i - p_i^\star|}{\sum_{i\in\mathcal{C}} |p_i^\star|} \tag{3.2}$$

On applique l'algorithme centralisé asynchrone au cas test à 118 nœuds issu de la bibliothèque de cas test IEEE, agrémenté de données temporelles [4] de consommations issues du jeu de données UMass [5] [99]. Les données temporelles ont un pas de temps de la minute, et l'on utilise les 1440 points d'une journée choisie arbitrairement pour nos calculs. Plus d'informations sur le cas test et le jeu de données sont situées en Annexe A.II.1.

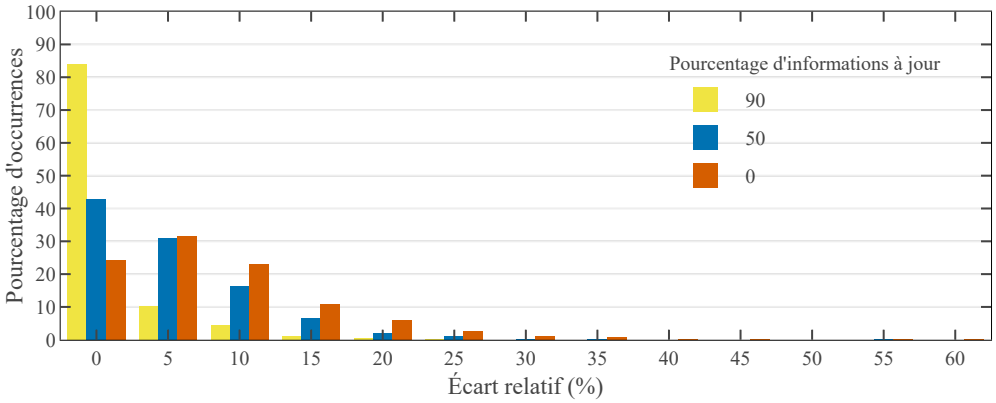


FIGURE 3.5 – Histogramme de l'écart relatif du résultat d'un marché centralisé asynchrone à différents pourcentages d'informations à jour reçues par l'agent central.

L'histogramme de l'écart relatif est affiché en Figure 3.5 pour différents pourcentages d'informations mises à jour. Le cas synchrone revient à 100% des informations mises à jour à temps pour la résolution de marché. Le cas à 0% d'informations mises à jour revient à considérer les données du pas de temps précédent pour l'intégralité des agents. On observe que l'écart relatif est plus faible lorsque le pourcentage d'informations mises à jour est grand. Cependant, cet écart est limité du fait de l'autocorrélation des données temporelles utilisées. En effet, d'un pas de temps à l'autre, la consommation varie peu. L'écart est donc dépendant des données temporelles. Elle sera plus grande si l'on considère des données avec une autocorrélation temporelle plus faible.

4. On a besoin ici de données temporelles car on remplace les données non reçues par celles de l'itération de marché précédente.

5. `https://traces.cs.umass.edu/index.php/smart/smart`

La résolution centralisée, bien qu'étant le type de résolution utilisée historiquement pour le marché de l'électricité, est limitée par les ressources de calculs lorsque le nombre d'agents pris en compte augmente. Pour compenser l'augmentation de l'incertitude due à l'introduction d'un grand nombre de producteurs renouvelables et à la désagrégation des consommateurs, il est souhaitable de réduire le temps d'avance de la résolution. Or, la seule manière de réduire ce temps d'avance est de ne mettre à jour que les données des agents qui ont répondu assez rapidement au gestionnaire de marché. Cette résolution dite asynchrone réduit le temps de résolution du marché mais introduit des écarts avec la solution optimale. De plus, elle est aussi soumise à la limitation du nombre d'agents du fait de sa nature centralisée.

3.3 Résolution décentralisée pair à pair

La décentralisation du problème de marché permet le passage à l'échelle en découpant la résolution du problème global en plusieurs problèmes résolus localement par chaque agent. Il n'y a plus d'opérateur de marché qui centralise toutes les données, les agents sont amenés à communiquer directement les uns avec les autres, comme illustré en Figure 3.6. Les résolutions décentralisées sont itératives, puisque les agents doivent minimiser leur propre fonction coût locale tout en se mettant d'accord avec leurs voisins. Cela implique un besoin en communication beaucoup plus grand que dans la résolution centralisée, donc le temps de résolution est susceptible d'être très impacté par les aléas de communication. Nous allons montrer dans cette section l'impact des aléas de communication sur la version synchrone de l'algorithme pair à pair.

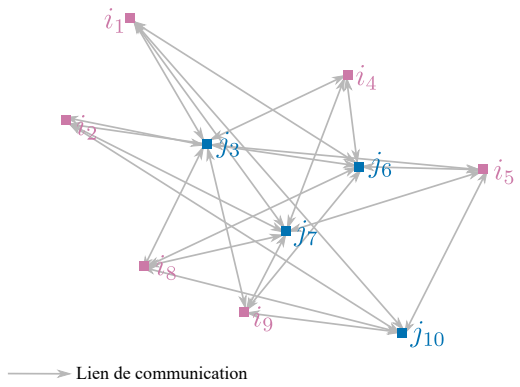


Figure 3.6 – Schéma exemple d'un réseau dont la résolution de marché est effectuée de gré à gré ou pair à pair. Les flèches grises représentent les liens de communication entre les agents du marché. Les agents en rose notés i sont des consommateurs et les agents en bleu notés j sont des producteurs.

Une des formulations décentralisées du problème de marché de l'électricité (3.1) est la formulation de marché pair à pair présentée en [45] et [46]. Elle consiste en la décomposition de la puissance active P_i de chaque consomacteur i en différents échanges P_{ij} entre le consomacteur i et ses partenaires commerciaux j. La formulation pair à pair est donnée en (3.3).

Formulation du marché décentralisé pair à pair

$$\min \sum_{i \in \mathcal{C}} f_i(P_i) \tag{3.3a}$$

$$\begin{aligned}
&\text{selon } P_i && i \in \mathcal{C} \\
&\quad p_{ij} && i \in \mathcal{C},\, j \in \mathcal{C}_i \\
&\text{tel que } P_i = \sum_{j \in \mathcal{C}_i} P_{ij} && i \in \mathcal{C} && \text{(3.3b)} \\
&\quad P_{ij} = -P_{ji} && i \in \mathcal{C},\, j \in \mathcal{C}_i && \text{(3.3c)} \\
&\quad \underline{P_i} \leq P_i \leq \overline{P_i} && i \in \mathcal{C} && \text{(3.3d)} \\
&\quad |P_{ij}| \leq \max(|\underline{P_i}|, |\overline{P_i}|) && i \in \mathcal{C},\, j \in \mathcal{C}_i && \text{(3.3e)}
\end{aligned}$$

La contrainte (3.3b) représente la décomposition de la puissance en termes d'échanges. On observe que la contrainte d'équilibre de puissance (3.1b) n'apparaît plus dans (3.3) : elle a été remplacée par une contrainte sur les échanges en (3.3c). Cette contrainte est plus forte que la contrainte d'équilibre d'origine. En effet, cette dernière portait sur une seule égalité tandis que la nouvelle contrainte impose que chacun des échanges du marché soit réciproque. En d'autres termes, si un consomacteur vend une quantité de puissance à un autre consomacteur, alors ce dernier s'engage à acheter la même quantité, créant ainsi un équilibre de puissance à hauteur de chaque échange.

Décomposition par ADMM La décomposition consiste à séparer le problème (3.3) en plusieurs sous-problèmes locaux. Chaque sous-problème est résolu par un agent de calcul différent en parallèle des autres. La décomposition peut se faire en groupant certains consomacteurs ensembles ou bien en considérant chaque consomacteur séparément. Pour la suite du chapitre, on considère que le problème est partitionné par consomacteur.

Ainsi, en considérant les variables P_i et $(P_{ij})_{j \in \mathcal{C}_i}$ locales à l'agent i, la fonction coût locale à minimiser est alors $f_i(P_i)$, et les contraintes (3.3b), (3.3d) et (3.3e) sont aussi décomposables. Seule la contrainte (3.3c) dépend d'une variable locale P_{ij} mais aussi d'une variable appartenant à un autre agent P_{ji}. Il est nécessaire de réaliser un consensus sur les échanges entre consomacteurs.

Différentes méthodes sont disponibles pour réaliser ce consensus, parmi elles la méthode consensus et innovation (C+I), la méthode du Lagrangien augmenté ou bien la méthode ADMM (pour *Alternating Directions Method of Multipliers*). C'est cette dernière qui a été choisie en [45] pour résoudre ce problème car elle présente des propriétés de convergence rapide, elle fonctionne pour tous les problèmes convexes et elle est largement utilisée dans la littérature.

La résolution des sous-problèmes locaux nécessite qu'ils ne dépendent que des variables propres à chaque agent d'une part, et des valeurs des agents voisins supposées communiquées d'autre part. Pour permettre cette décomposition, on pose une variable globale $\mathbf{W} = [W_{ij}]$ ainsi que la contrainte de consensus :

$$\frac{W_{ij} - W_{ji}}{2} = P_{ij} \tag{3.4}$$

Cette variable $\mathbf{W}$ est remplacée par la suite car il est possible de démontrer que $\frac{W_{ij}^t - W_{ji}^t}{2} = \frac{P_{ij}^t - P_{ji}^t}{2}$ pour toute itération $t \geq 1$. Après avoir appliqué la méthode ADMM sur cette variable, on obtient les équations itératives locales (3.5).

$$(p_i, (p_{ij})_{j\in\mathcal{C}_i})^{t+1} = \underset{(P_i,(P_{ij})_{j\in\mathcal{C}_i})}{\arg\min} f_i(P_i) + \sum_{j\in\mathcal{C}_i} \frac{\rho}{2}\left(\frac{p_{ij}^t - p_{ji}^t}{2} - P_{ij} + \frac{\lambda_{ij}^t}{\rho}\right)^2 \tag{3.5a}$$

$$\text{tel que} \quad P_i = \sum_{j\in\mathcal{C}_i} P_{ij}$$

$$\underline{P_i} \leq P_i \leq \overline{P_i}$$

$$|P_{ij}| \leq \max(|\underline{P_i}|, |\overline{P_i}|) \qquad j \in \mathcal{C}_i$$

$$\lambda_{ij}^{t+1} = \lambda_{ij}^t - \rho\frac{p_{ij}^{t+1} + p_{ji}^{t+1}}{2} \qquad j \in \mathcal{C}_i \tag{3.5b}$$

où λ_{ij} représente la variable duale de la contrainte (3.4). Elle est mise à jour en (3.5b) via une descente de gradient. La mise à jour des variables P_i et $(P_{ij})_{j\in\mathcal{C}_i}$ est réalisée localement par l'agent i en (3.5a). On observe que pour réaliser ces mises à jour, il est nécessaire qu'à chaque itération t les agents $j \in \mathcal{C}_i$ partenaires de i communiquent leurs valeurs p_{ji}^t à l'agent i.

Unicité de la solution Le fait d'avoir rajouté des variables d'échanges P_{ij} a pour conséquence d'avoir une multiplicité des solutions possibles, toutes équivalentes à la solution optimale du problème de marché d'origine (3.1). En effet, si $p_i^\star$ est la solution optimale de l'agent i, alors il existe une infinité de combinaisons des p_{ij} pour répondre à la contrainte $p_i = \sum_{j\in\mathcal{C}_i} p_{ij}$. L'algorithme ADMM synchrone présenté ci-dessus étant déterministe, il trouvera systématiquement la même solution $(p_{ij}^\star)_{j\in\mathcal{C}_i}$. Cependant, l'algorithme asynchrone n'étant pas déterministe, comme expliqué dans la section 3.4, on risque de ne pas trouver la même solution au niveau des variables d'échange p_{ij}. Même si la solution en terme de puissance p_i est bien la solution optimale voulue, on cherche tout de même à obtenir exactement la même solution quel que soit le mode de résolution synchrone ou asynchrone.

Pour obtenir la même solution, on rajoute à la fonction à minimiser un terme de régularisation en $\gamma\sum_{j\in\mathcal{C}_i} P_{ij}^2$. Ce terme permet d'obtenir la solution qui uniformise les valeurs des échanges, en décourageant les agents d'échanger de grandes quantités à un prix plus bas. On appelle γ le facteur de régularisation.

Nota — Il faut choisir une valeur numérique du facteur de régularisation γ assez grande pour qu'il soit pris en compte dans l'optimisation, mais pas trop au risque d'obtenir un déplacement de la solution optimale. Le réglage de ce facteur sera développé en section 3.4.2.

Nota — L'article [45] propose de rajouter des taxes d'utilisation du réseau pour chaque échange en rajoutant un terme $\sum_{j\in\mathcal{C}_i} \gamma_{ij} P_{ij}$ à la fonction locale à minimiser. Ces termes représentent des coûts supplémentaires plus ou moins élevés selon le partenaire avec lequel on échange, et permettent de limiter l'utilisation de certaines lignes qui se retrouveraient sinon congestionnées. L'étude qui suit s'intéresse à la résolution asynchrone de l'algorithme, nous avons donc décidé d'omettre ce terme.

Algorithme local synchrone Tous les agents effectuent leurs calculs en parallèle selon l'Algorithme 3.3. Du point de vue d'un agent i, après avoir mis à jour ses variables locales P_i et $(P_{ij})_{j\in\mathcal{C}_i}$ suivant (3.6), les valeurs mises à jour de ces variables sont alors envoyées à ses partenaires commerciaux. L'agent i envoie, pour chacun de ses partenaires commerciaux $j \in \mathcal{C}_i$, la valeur mise à jour p_{ij}^{t+1}. En retour, il attend toutes les valeurs p_{ji}^{t+1} de tous ses partenaires. À la réception de tous ces messages, l'agent i peut ensuite continuer ses calculs en mettant à jour ses variables duales locales avec (3.7), avant de réitérer les étapes décrites précédemment. La Figure 3.7 représente les différents échanges d'information qui ont lieu à chaque itération pour un marché à trois agents.

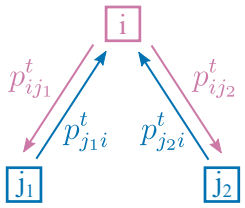


FIGURE 3.7 – Schéma des échanges d'information à l'itération t entre les différents agents du marché pair à pair. Application à un marché à trois agents $\{i, j_1, j_2\}$ où le seul partenaire commercial des agents j_1 et j_2 est l'agent i. Les agents j_1 et j_2 n'étant pas partenaires l'un avec l'autre, ils ne s'échangent pas de messages.

L'algorithme est implémenté sur un cas test à 110 agents explicité en Annexe A.I, et les délais de communication sont simulés et suivent le modèle présenté au chapitre 2 avec des paramètres qui mènent à des délais de communication moyens d'environ 60 ms. La Figure 3.8 présente l'avancement de l'algorithme pair à pair en fonction du temps de la simulation, et compare la solution finale trouvée avec la solution optimale obtenue de manière centralisée et synchrone. L'algorithme converge vers la solution optimale au bout d'environ 4 secondes.

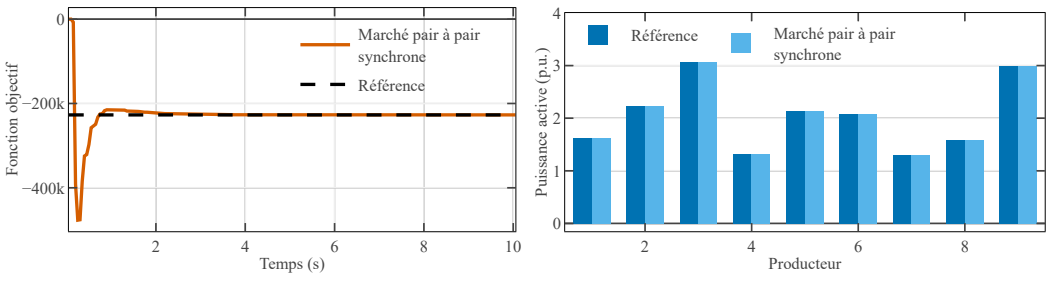


FIGURE 3.8 – Résultat de l'algorithme de marché pair à pair synchrone appliqué au cas test à 110 agents et comparaison par rapport à la solution optimale obtenue de manière centralisée synchrone. En Figure 3.8a, la valeur de la fonction objectif globale est affichée au cours du temps de la simulation. En Figure 3.8b, les puissances actives de 9 producteurs au bout de 10 secondes de calculs sont comparées à la solution optimale.

Algorithme 3.3 Marché pair à pair synchrone local à l'agent i

tant que la convergence n'est pas atteinte
  si $t = 0$ **alors**
    Initialiser les échanges $(p_{ij})_{j\in\mathcal{C}_i}$
    Initialiser les variables duales $(\lambda_{ij})_{j\in\mathcal{C}_i} \leftarrow 0$
  sinon
    Mettre à jour les variables locales via

$$\mathbf{x}^{t+1} \leftarrow \underset{(P_i,\,(p_{ij})_{j\in\mathcal{C}_i})}{\arg\min} f_i(P_i) + \sum_{j\in\mathcal{C}_i} \left[\gamma\, P_{ij}^2 + \frac{\rho}{2}\left(\frac{p_{ij}^t - p_{ji}^t}{2} - P_{ij} + \frac{\lambda_{ij}^t}{\rho}\right)^2\right] \tag{3.6a}$$

$$\text{selon } \mathbf{x} : \begin{cases} P_i \\ P_{ij} \quad j \in \mathcal{C}_i \end{cases}$$

$$\text{tel que } P_i = \sum_{j\in\mathcal{C}_i} P_{ij} \tag{3.6b}$$

$$\underline{P_i} \le P_i \le \overline{P_i} \tag{3.6c}$$

$$|P_{ij}| \le \max(|\underline{P_i}|, |\overline{P_i}|) \qquad j \in \mathcal{C}_i \tag{3.6d}$$

  fin si
  Envoyer les valeurs p_{ij}^{t+1} aux partenaires commerciaux $j \in \mathcal{C}_i$
  tant que les valeurs p_{ji}^{t+1} de $j \in \mathcal{C}_i$ ne sont pas tous reçus
    Attendre
  fin tant que
  Mettre à jour les variables duales pour tout $j \in \mathcal{C}_i$ via

$$\lambda_{ij}^{t+1} \leftarrow \lambda_{ij}^t - \rho\frac{p_{ij}^{t+1} + p_{ji}^{t+1}}{2} \tag{3.7}$$

  $t \leftarrow t + 1$
fin tant que

Convergence de l'algorithme La décomposition étant basée sur la méthode ADMM [96], on utilise aussi les métriques de convergence de cette méthode, à savoir : le résidu primal et le résidu dual. Les résidus de l'itération t locaux à l'agent i sont donnés en (3.8).

$$r_i^t = \sum_{j\in\mathcal{C}_i} (p_{ij}^t + p_{ji}^t)^2 \tag{3.8a}$$

$$s_i^{t+1} = \sum_{j\in\mathcal{C}_i} (p_{ij}^{t+1} - p_{ij}^t)^2 \tag{3.8b}$$

Le résidu primal local (3.8a) est une mesure de l'équilibre des échanges de l'agent i avec ses partenaires commerciaux. Arrivé à convergence, l'équilibre des échanges doit être vérifié donc p_{ij} et p_{ji} tendent vers des valeurs opposées et $r_i^t \underset{t\to\infty}{\longrightarrow} 0$. Le résidu dual local (3.8b) est

une mesure de la variation des échanges de l'agent i entre deux itérations. À convergence, cette variation est amenée à être nulle $s_i^t \underset{t\to\infty}{\longrightarrow} 0$.
Les résidus globaux sont définis en (3.9) comme la somme de tous les résidus locaux.

$$r^t = \sum_{i\in\mathcal{C}} r_i^t \tag{3.9a}$$

$$s^t = \sum_{i\in\mathcal{C}} s_i^t \tag{3.9b}$$

La convergence de l'algorithme est définie en comparant ces résidus à des seuils de tolérance ε_p et ε_d pour le résidu primal et le résidu dual respectivement. On considère que l'algorithme a convergé lorsque ces deux résidus sont en dessous de ces seuils.

$$r^t \leq \varepsilon_p \qquad s^t \leq \varepsilon_d \tag{3.10}$$

Nota

En ce qui concerne le calcul du résidu global dans le cadre d'une résolution décentralisée, plusieurs solutions sont possibles. Baroche [46] propose de communiquer l'ensemble des résidus locaux à tous les agents, ce qui augmenterait fortement les besoins en communication. Un protocole de communication de type *publish-subscribe* serait idéal pour éviter l'explosion du nombre de messages. Cela reviendrait à centraliser les résidus au travers d'un *broker* et de partager les informations à tous les agents du marché. On pourrait aussi considérer un agent organisateur qui réaliserait le calcul de résidu et enverrait un message à tous les participants du marché pour déclarer la convergence. On perdrait dans ce cas l'avantage d'un marché totalement décentralisé en ajoutant un agent "central", mais on réduirait fortement les échanges. Dans l'étude réalisée, on ne prend pas en compte l'envoi de ces messages, on peut imaginer des critères d'arrêt locaux par échange comme proposé dans [100].

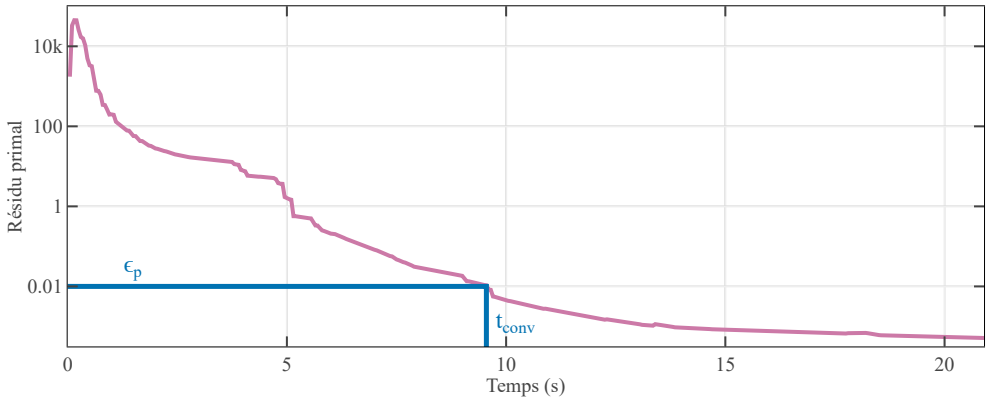


FIGURE 3.9 – Résidu primal en fonction du temps (horloge de la simulation) pour un cas test à 110 agents. La convergence est définie pour $\varepsilon_p = 10^{-2}$, on peut en déduire le temps de convergence de $t_{conv} \simeq 9.5$ secondes.

Pour la suite de nos études, par souci de simplicité, nous considérerons seulement la condition de convergence primale. La Figure 3.9 représente le résidu primal en fonction du

temps, on détermine le temps de convergence à partir d'une certaine valeur du résidu ε_p. Des études similaires peuvent être menées en considérant la condition de convergence duale.

$$r^t \leq \varepsilon_p \tag{3.11}$$

3.4 Résolution décentralisée pair à pair asynchrone

L'algorithme décentralisé synchrone nécessite beaucoup d'échanges entre les agents. À chaque itération, chaque agent envoie autant de messages qu'il possède de partenaires commerciaux et doit attendre de recevoir tous les messages de ses partenaires. Cela rend la résolution très dépendante de l'état du réseau de communication. Dans le cas où, ne serait-ce qu'un message se perd entre deux agents, alors l'agent destinataire serait bloqué à une itération t pendant que tous les autres agents du marché continuent leurs calculs pour l'itération $t+1$. Tous les partenaires de l'agent bloqué seraient à leur tour bloqués dans l'attente de la mise à jour $t+1$. Ainsi, de proche en proche, la perte d'un message conduit à un blocage général de la résolution de l'algorithme.

Plaçons nous dans le cas moins extrême où les communications se font suivant un protocole fiable, c'est-à-dire qu'un message est renvoyé après un certain délai si le destinataire ne l'a pas encore reçu, et que les messages arrivent dans leur ordre d'envoi. Une perte de message se traduit alors comme un rallongement du délai de communication. Même dans ce cas, l'allongement du délai induit un ralentissement général de l'algorithme étant donné que l'agent destinataire doit attendre l'intégralité des messages avant de procéder aux calculs. En d'autres termes, la vitesse d'une itération est systématiquement limitée par l'arrivée du dernier message. Il suffit que le réseau de communication soit très hétérogène pour qu'une grande partie du temps de résolution soit en réalité du temps d'attente.

Dans un souci de résilience de l'algorithme face aux aléas de communication, nous proposons alors une version asynchrone de la résolution décentralisée. Cette version permet de réduire les temps d'attente entre chaque itération, et fournit la même solution optimale que la version synchrone. Elle est qualifiée d'*asynchrone* puisque les différents agents impliqués n'avancent plus au même rythme, certains peuvent avoir une fréquence de calcul plus élevée que d'autres, ils ne partagent donc pas le même numéro d'itération en tout temps.

Déclenchement asynchrone des calculs locaux Dans la version asynchrone de l'algorithme décentralisé, au niveau d'un agent, les calculs de l'itération suivante sont déclenchés malgré le fait que tous les messages n'aient pas encore été reçus. Les données non reçues sont remplacées par les dernières données reçues en date. On peut considérer plusieurs types de déclenchement asynchrone :

- le déclenchement après un certain temps depuis l'itération précédente ou *timeout*,
- le déclenchement au bout d'un certain nombre de messages reçus,
- une combinaison de ces deux déclenchements, celui qui se produit en premier des deux,
- le déclenchement programmé parmi tous les agents de la résolution.

L'attente d'un temps donné présente l'avantage de garantir une certaine fréquence de déclenchement des calculs quelque soit l'état du réseau de communication. Cependant, cette durée de *timeout* peut être définie de manière trop conservatrice, ce qui ralentirait encore plus l'algorithme, ou bien trop optimiste, ce qui ne laisserait même pas le temps à un message d'arriver à destination. De plus, le temps d'attente optimal varie selon l'état du réseau de communication. Il est donc plus simple de considérer le déclenchement par nombre de messages, et de rajouter un délai maximal conservateur pour s'assurer de l'avancement de l'algorithme.

Nous considérons seulement le déclenchement par nombre de messages reçus par souci de concision. Cela nous permet de définir simplement un paramètre d'asynchronisme δ, que l'on retrouvera tout au long des différentes études asynchrones. Ce paramètre représente le taux de messages attendus à chaque itération. Le nombre de message minimum attendu

par chaque agent i est défini à droite de l'inégalité en (3.12), où $\mathcal{A}_i^t$ désigne l'ensemble des partenaires de i dont un message a été reçu par l'agent i à l'itération t, et $\mathcal{C}_i$ désigne l'ensemble des partenaires commerciaux de l'agent i.

$$|\mathcal{A}_i^t| \geq \lceil \delta \cdot |\mathcal{C}_i| \rceil \tag{3.12}$$

Nota

L'inégalité de (3.12) représente le fait que le nombre de messages effectivement reçus à une itération et le nombre de messages déclencheur peuvent être différents. En effet, dans le cas où plusieurs messages sont reçus au même moment, ou bien si les messages sont reçus pendant le temps de calcul de l'agent, il est possible que l'on reçoive un nombre supérieur de messages. Dans ce cas, on prend en compte les informations de tous les messages reçus.

Mise à jour des variables locales et partage de l'information À chaque itération t, l'agent i attend $|\mathcal{A}_i^t|$ messages de ses partenaires commerciaux avant de continuer les calculs de l'itération suivante. Pour $j \in \mathcal{A}_i^t$, il écrase ses copies internes p_{ji} avec les données des nouveaux messages reçus. Les copies internes des p_{ji} pour $j \in \mathcal{C}_i \setminus \mathcal{A}_i^t$, c'est-à-dire pour les partenaires dont l'agent i n'a pas reçu de messages à l'itération t, restent inchangées. En revanche, toutes les variables internes locales p_{ij} pour $j \in \mathcal{C}_i$ sont mises à jour simultanément en synchrone via (3.6). En asynchrone, on définit une variable intermédiaire temporaire p_{ij}^{temp} qui met en tampon le résultat de (3.6) pour tous les $j \in \mathcal{C}_i$, et qui permet de décider a posteriori quelles variables locales p_{ij} sont effectivement mises à jour. En effet le vecteur $(p_{ij})_{j\in\mathcal{C}_i}^{temp}$ possède $|\mathcal{C}_i|$ éléments et on se demande quels éléments doivent effectivement être mis à jour.

Il existe plusieurs possibilités de mise à jour des variables internes et leur communication. Parmi les $|\mathcal{C}_i|$ variables $(p_{ij})_{j\in\mathcal{C}_i}$, doit-on les mettre à jour pour tout $j \in \mathcal{C}_i$ ou bien seulement pour les j dont on a reçu un message avant de commencer les calculs de l'itération ? La même question se pose pour les $|\mathcal{C}_i|$ variables $(\lambda_{ij})_{j\in\mathcal{C}_i}$. De même pour la communication des nouvelles valeurs avec les partenaires commerciaux.

En ce qui concerne les mises à jour des variables locales p_{ij} et λ_{ij}, les quatre possibilités ont été testées et leur résultat compilés dans le Tableau 3.1. Parmi ces schémas, le seul qui permet à l'algorithme de converger vers la solution optimale est celui où les mises à jour des variables p_{ij} et λ_{ij} s'effectuent seulement si $j \in \mathcal{A}_i^t$ c'est-à-dire si l'agent i a reçu un message de j avant de commencer ses calculs de l'itération t. Comme pour les copies de p_{ji}, les copies internes de p_{ij} et λ_{ij} pour $j \in \mathcal{C}_i \setminus \mathcal{A}_i^t$ restent alors inchangées.

p_{ij} \ λ_{ij}	$j \in \mathcal{C}_i$	$j \in \mathcal{A}_i^t$
$j \in \mathcal{C}_i$	Converge mais pas vers la solution optimale	Converge mais pas vers la solution optimale
$j \in \mathcal{A}_i^t$	Ne converge pas	Converge vers la solution optimale

Tableau 3.1 – Schéma de mise à jour et de communication pour l'algorithme asynchrone.

En ce qui concerne la communication avec les autres agents, étant donné que les valeurs internes de p_{ij} ne varient pas pour $j \in \mathcal{C}_i \setminus \mathcal{A}_i^t$, il n'est pas nécessaire de leur renvoyer ces

valeurs. Ainsi, l'agent i ne renvoie un message contenant la valeur de p_{ij}^t à l'agent j seulement si $j \in \mathcal{A}_i^t$, donc seulement si l'agent i a reçu le message p_{ji} avant de commencer ses calculs d'itération.

Pistes pour la preuve de convergence Une preuve de convergence pour l'algorithme asynchrone de consensus ADMM possédant une décomposition par arête dans le cadre d'un réseau de communication avec pertes est donnée en [75]. Un graphe regroupant les processeurs et liens de communication est défini : $\mathscr{G}(\mathscr{V}, \mathscr{E})$ représente le graphe d'origine, avec $\mathscr{V}$ l'ensemble des sommets du graphes représentant les processeurs, et $\mathscr{E}$ les arêtes du graphe représentant les liens de communication entre processeurs. Les communications avec pertes sont modélisées par des sous-graphes $\mathscr{G}^t(\mathscr{V}, \mathscr{E}^t)$ du graphe original $\mathscr{G}$. La convergence se prouve en prenant l'hypothèse qu'à chaque itération, le nombre d'arêtes actives $|\mathscr{E}^t|$ suit une loi binomiale $\mathcal{B}(E, \varepsilon)$ où E représente le nombre total d'arêtes du graphe d'origine et $0 \leq \varepsilon < 1$ représente la probabilité de défaillance de la communication.

Cet algorithme est légèrement différent en ce qu'il concerne le consensus d'une unique variable sur l'ensemble de n processeurs. Cependant, il est facilement transposable à notre problème en considérant autant de variables sur lesquelles effectuer un consensus qu'il y a de liens d'échanges entre les agents du marché. De plus, le consensus sur une variable d'échange n'a besoin de se faire qu'entre les deux agents impliqués dans l'échange, car le reste des agents n'a pas besoin de savoir quel est la quantité échangée entre ces deux agents.

Ensuite, l'hypothèse des sous-graphes aléatoires est réductrice par rapport aux échanges considérés lors de nos simulations asynchrones. En effet, en ne considérant les communications que comme des variables booléennes, on n'autorise que des messages qui proviennent de la même itération, et pas d'itérations précédentes $t-2$, $t-3$, etc. Ainsi la "profondeur d'asynchronisme" est limitée à l'itération $t-1$ dans l'article considéré. De plus, les messages sont considérés comme effectivement perdus tandis que nous considérons un système de renvoi de messages dans le cas d'une perte. Une hypothèse moins forte que notre cas de figure est que l'envoi d'un message d'un agent i vers un agent j est aléatoire et totalement indépendant de la réception d'un message à l'itération précédente.

Malgré les divergences entre notre algorithme asynchrone et celui tiré de [75], il est important de souligner que la preuve de convergence d'un algorithme asynchrone de consensus basé sur de l'ADMM a été donnée dans [75]. Nous nous contenterons de validations expérimentales de convergence le long de ce manuscrit, étant donné que l'on se focalise plutôt sur les temps de convergence ainsi que les avantages de l'asynchrone sur le synchrone.

3.4.1 Formulation et algorithme

L'algorithme asynchrone final est donné dans l'Algorithme 3.4. On y retrouve les mises à jour partielles des variables locales p_{ij} et λ_{ij}, ainsi que le schéma de communication partiel entre les agents, tous mis en évidence en bleu pour mieux faire ressortir les différences avec l'algorithme synchrone. Cette mise à jour partielle requiert l'utilisation d'une variable temporaire $\mathbf{x}^{temp}$, car le résultat de (3.13) renvoie une mise à jour possible pour toutes les variables locales p_{ij} où $j \in \mathcal{C}_i$. Les réelles mises à jour de ces variables se font en (3.13e) et (3.13f). Pour ce qui est du temps d'attente de la réception des messages voisins, l'agent i ne reprend ses calculs d'itération qu'une fois qu'il a reçu les messages de l'ensemble $\mathcal{A}_i^{t+1}$ durant l'itération t. La constitution de l'ensemble $\mathcal{A}_i^t$ est déterminée par l'ordre d'arrivée des messages, qui dépend de l'instant où le message est envoyé et du délai de communication. La notation t désigne bien l'itération locale de l'agent, elle peut être différente d'un agent à un autre durant un instant donné de la simulation.

Algorithme 3.4 Marché pair à pair asynchrone local à l'agent i

tant que la convergence n'est pas atteinte
si $t = 0$ **alors**
Initialiser les échanges $(p_{ij})_{j \in \mathcal{C}_i}$
Initialiser les variables duales $(\lambda_{ij})_{j \in \mathcal{C}_i} \leftarrow 0$
sinon
Mettre à jour les variables locales via

$$\mathbf{x}^{temp} \leftarrow \underset{(P_i, (P_{ij})_{j \in \mathcal{C}_i})}{\arg\min} f_i(P_i) + \sum_{j \in \mathcal{C}_i} \left[\gamma P_{ij}^2 + \frac{\rho}{2} \left(\frac{p_{ij}^t - p_{ji}^t}{2} - P_{ij} + \frac{\lambda_{ij}^t}{\rho} \right)^2 \right] \tag{3.13a}$$

$$\text{selon } \mathbf{x} : \begin{cases} P_i \\ P_{ij} \quad j \in \mathcal{C}_i \end{cases}$$

$$\text{tel que } P_i = \sum_{j \in \mathcal{C}_i} P_{ij} \tag{3.13b}$$

$$\underline{P_i} \leq P_i \leq \overline{P_i} \tag{3.13c}$$

$$|P_{ij}| \leq \max(|\underline{P_i}|, |\overline{P_i}|) \qquad j \in \mathcal{C}_i \tag{3.13d}$$

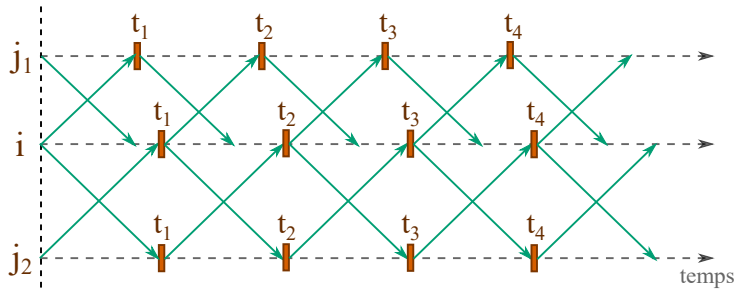
$$p_i \leftarrow P_i^{temp} \tag{3.13e}$$

$$p_{ij} \leftarrow P_{ij}^{temp} \qquad j \in \mathcal{A}_i^t \tag{3.13f}$$

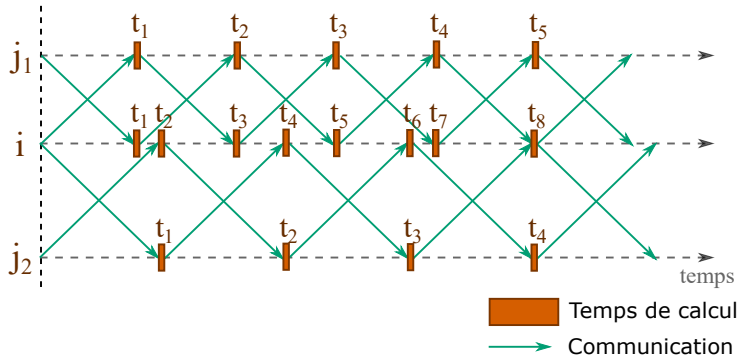
fin si
Envoyer les valeurs p_{ij} aux partenaires commerciaux $j \in \mathcal{A}_i^t$
tant que les valeurs p_{ji} de $j \in \mathcal{A}_i^{t+1}$ ne sont pas tous reçus
Attendre
fin tant que
Mettre à jour les variables duales pour tout $j \in \mathcal{A}_i^{t+1}$ via

$$\lambda_{ij} \leftarrow \lambda_{ij} - \rho \frac{p_{ij} + p_{ji}}{2} \qquad j \in \mathcal{A}_i^{t+1} \tag{3.14}$$

$t \leftarrow t + 1$
fin tant que



(a) Synchrone



(b) Asynchrone

FIGURE 3.10 – Échanges de messages en fonction du temps pour un marché constitué de trois agents, présenté en Figure 3.7. La Figure 3.10a présente les échanges dans le cas de l'algorithme synchrone tandis que la Figure 3.10b les présente en asynchrone. Dans le cas asynchrone, l'agent i n'a pas à attendre les deux messages de ses partenaires avant de continuer ses calculs, ce qui permet à l'agent j_1 de réaliser plus d'itérations durant un même délai.

La Figure 3.10 présente la différence entre une résolution synchrone et une résolution asynchrone sur un exemple à trois agents $\{i, j_1, j_2\}$ où i est le seul agent partenaire commercial de j_1 et j_2. On y distingue en orange les instants de calculs de chaque agent, et les échanges d'informations sont représentés par des flèches vertes. Les délais de communication entre i et j_1 sont sensiblement plus rapides qu'entre i et j_2. Les agents j_1 et j_2 n'ont qu'un seul message à attendre avant de réaliser les calculs de l'itération suivante, tandis que l'agent i doit attendre les messages de j_1 et j_2. Lors de la résolution synchrone en Figure 3.10a, les agents progressent dans leurs itérations au même rythme car ils doivent attendre l'intégralité des messages pour passer à l'itération suivante. Cependant, on voit que l'agent i passe quasiment la moitié de son temps à attendre le deuxième message de j_2 après avoir reçu celui de j_1. Dans la version asynchrone de la résolution en Figure 3.10b, les calculs de l'agent i sont déclenchés seulement après avoir reçu un message, au lieu de deux. Cela a pour effet de presque doubler sa fréquence de calculs en réduisant le temps d'attente entre chaque itération. L'agent j_1 avance aussi plus rapidement dans ses itérations en retour. Seul l'agent j_2, dont les délais de communication avec l'agent i étaient les plus longs, conserve sa fréquence de calcul.

L'algorithme asynchrone pair à pair est appliqué sur le cas test à 110 agents présenté précédemment. Les délais de communication considérés sont de l'ordre de 60 ms. La Figure 3.11

représente l'évolution temporelle de la fonction objectif globale de l'algorithme pair à pair, ainsi que la comparaison avec la solution de référence une fois la convergence atteinte.

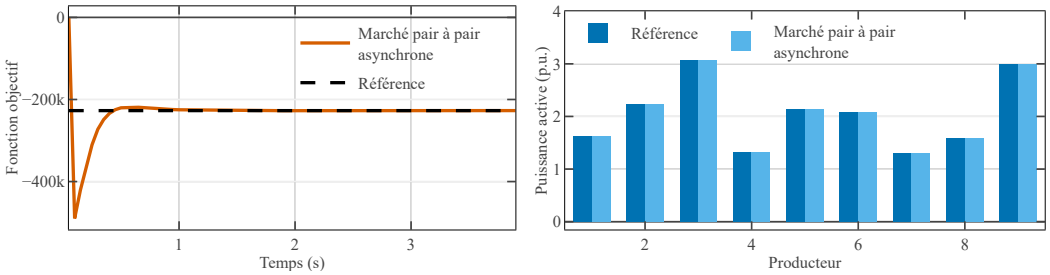


FIGURE 3.11 – Résultat de l'algorithme de marché pair à pair asynchrone appliqué au cas test à 110 agents et comparaison par rapport à la solution optimale centralisée synchrone. Le paramètre d'asynchronisme est réglé ici à $\delta = 10\%$. En Figure 3.11a, la valeur de la fonction objectif globale est affichée au cours du temps. En Figure 3.11b, les puissances actives de 9 producteurs au bout de 4 secondes de calculs sont comparées à la solution optimale.

Les sections suivantes traitent des résultats des simulations effectuées où les algorithmes pair à pair synchrone et asynchrone présentés ci-dessus ont été appliqués sur un cas test à 110 agents explicité en Annexe A.I. Les problèmes d'optimisation locaux des agents du marché sont résolus via le solveur `OSQP` [98] basé sur la méthode ADMM. Beaucoup de paramètres entrent en jeu lorsque l'on considère la résolution de l'algorithme de marché pair à pair. Le premier paramètre à régler est le facteur ρ qui détermine la vitesse d'avancement de l'algorithme. Il est déterminé de manière empirique à $\rho = 15$ pour le cas test considéré.

Section	**Cas**	Δ_{comm}	Δ_{calc}
3.4.2	110 agents $\rho = 15$	$\alpha = 50.0$ $\beta = 10$ ms Déterministe $p_{pertes} = 0\%$	Négligé devant Δ_{comm}
3.4.3	110 agents $\rho = 15$ $\gamma = 1.0$	Déterministe $p_{pertes} = 0\%$	Négligé devant Δ_{comm}
3.4.4 3.4.6	110 agents $\rho = 15$ $\gamma = 1.0$	$\alpha = 50.0$ $\beta = 10$ ms Modèle gaussien $p_{pertes} = 0\%$	Négligé devant Δ_{comm}
3.4.5	110 agents $\rho = 15$ $\gamma = 1.0$	$\alpha = 50.0$ $\beta = 10$ ms Modèles gaussien et avancé $p_{pertes} = 0\%$	Négligé devant Δ_{comm}
3.4.7	110 agents $\rho = 15$ $\gamma = 1.0$	$\alpha = 50.0$ $\beta = 10$ ms Déterministe $p_{pertes} = 0\%$	Négligé devant Δ_{comm}

TABLEAU 3.2 – Paramètres utilisés pour les simulations de l'algorithme de marché pair à pair.

On effectue ensuite le réglage du le facteur de régularisation en section 3.4.2, avant d'étudier les effets du modèle de communication sur le temps de convergence de l'algorithme en section 3.4.3. La section 3.4.4 montre l'influence du paramètre d'asynchronisme sur le temps de convergence. On compare le modèle de délais de communication gaussien au modèle avancé dans la section 3.4.5. Enfin, on finit par évaluer le nombre de messages échangés total en section 3.4.6 ainsi que l'influence de la distance sur le résidu primal en section 3.4.7.

3.4.2 Réglage du facteur de régularisation

Dans un premier temps, il est nécessaire de régler le facteur de régularisation γ présenté en section 3.3 et qui permet d'obtenir l'unicité de la solution en synchrone et en asynchrone. Les résultats des simulations avec $\gamma = 0$, c'est-a-dire en l'absence du terme de régularisation, sont compilées dans le Tableau 3.3. On affiche les valeurs de quelques variables liées à l'agent 32, choisi arbitrairement, lorsque la convergence est atteinte, dont la puissance p_{32}, ainsi que les quatre premières variables d'échanges. À la convergence, les variables de puissance p_{32} sont toutes quasiment égales qu'elles proviennent d'une résolution synchrone ou d'une résolution asynchrone. En revanche, on remarque des disparités dans les valeurs des variables d'échange $p_{32,j}$, ce qui est attendu étant donné la multiplicité des résultats d'une optimisation non strictement convexe.

Variable d'optimisation	Valeur pour $\delta = 1$	$\delta = 0.7$	$\delta = 0.2$
p_{32}	-44.03	-44.03	-44.03
$p_{32,1}$	-1.58	-1.69	-1.73
$p_{32,2}$	-1.73	-1.94	-1.72
$p_{32,3}$	-1.44	-1.31	-1.31
$p_{32,4}$	-0.52	-0.10	-0.00

TABLEAU 3.3 – Valeurs des variables d'optimisation à convergence ($\varepsilon_{rel} = 10^{-12}$) pour différents δ, dans le cas où il n'y a pas de termes de régularisation : $\gamma = 0$.

Le problème de marché d'origine (3.1) est effectivement résolu par les agents décentralisés, que ce soit en synchrone ou en asynchrone. Cependant, il est nécessaire pour permettre des comparaisons valides que les échanges p_{ij} soient aussi égaux pour toute valeur du paramètre d'asynchronisme δ.

On se réfère à la solution synchrone pour comparer les différentes solutions : $p^*_{i,\delta=1}$ et $p^*_{ij,\delta=1}$. Deux quantités sont affichées en Figure 3.12 en fonction du facteur de régularisation γ :

— la valeur de la fonction coût globale à la convergence

$$C_\delta = \sum_{i \in \mathcal{C}} f_i(p^*_{i,\delta}) \tag{3.15}$$

— la différence normalisée des échanges entre la solution de référence synchrone ($\delta = 1$) et la solution étudiée

$$\mathrm{v}_\delta = \mathrm{v}\left(\mathbf{T}^*_\delta, \mathbf{T}^*_{\delta=1}\right) = \frac{\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} |p^*_{ij,\delta} - p^*_{ij,\delta=1}|}{\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}_i} |p^*_{ij,\delta=1}|} \tag{3.16}$$

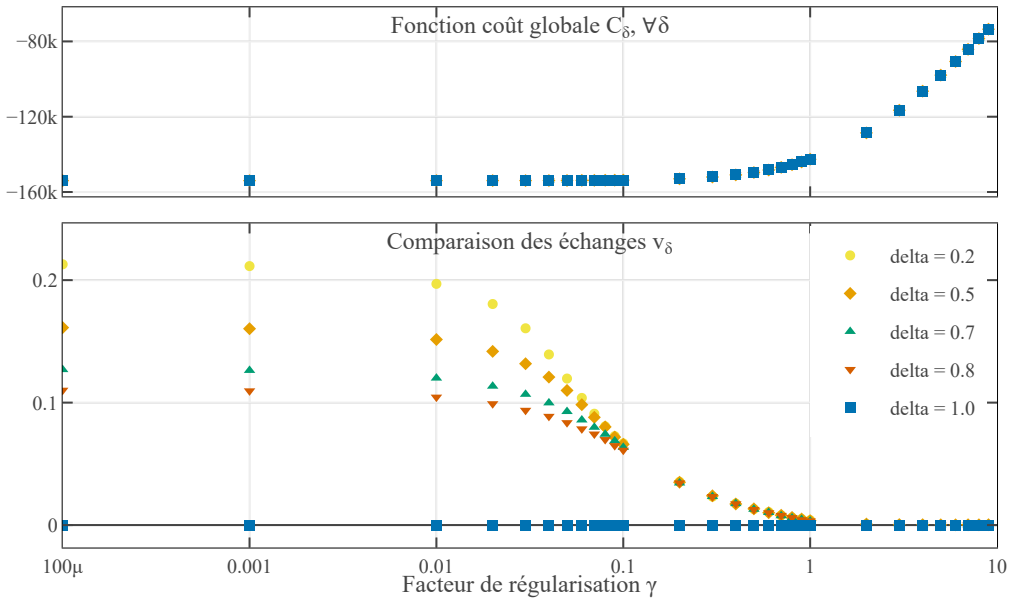


FIGURE 3.12 – Fonction coût globale et comparaison des échanges en fonction du facteur de régularisation γ, pour différentes valeurs du paramètre d'asynchronisme δ. La valeur de la fonction coût globale est indépendante du paramètre d'asynchronisme δ.

On observe alors dans la partie haute de la Figure 3.12 que la fonction coût optimale est indépendant du paramètre d'asynchronisme δ, ce qui est attendu étant donné que la solution en puissance p_i vaut toujours la même et unique valeur optimale. La comparaison des échanges décroît vers zéro lorsque le facteur de régularisation γ augmente, ce qui signifie que les échanges tendent vers la même solution. Cependant, le coût global augmente aussi avec γ selon la relation (3.13), ce qui montre que la solution optimale s'écarte de la solution de référence à $\gamma = 0$. Le choix du facteur γ le plus adapté s'effectue de manière à minimiser la différence entre les échanges $\mathrm{v}\left(\mathbf{T}^*_\delta, \mathbf{T}^*_{\delta=1}\right)$ pour tous les δ tout en conservant une valeur de fonction coût la plus proche de la solution de référence. La lecture de la Figure 3.12 pousse à choisir la valeur, spécifique à chaque cas d'étude, du facteur de régularisation qui répond à ces exigences. Dans le cas considéré, cette valeur est égale à $\gamma = 1$. Elle permet de minimiser les différences entre solutions synchrones et asynchrones jusqu'à $v_{0.2} = 0.004 = 0.4\%$, tout en limitant l'augmentation du coût global à $+8\%$, comme le montre le Tableau 3.4.

γ	Augmentation du coût global	Différence des échanges
0	0	21%
1.0	+8%	0.4%
9.0	+48%	0.03%

TABLEAU 3.4 – Augmentation du coût global par rapport au cas où $\gamma = 0$ et la différence normalisée des échanges associée.

3.4.3 Influence des délais de communication

L'état du réseau de communication est modélisé par les paramètres α et β représentant respectivement le coefficient linéaire et constant en fonction de la distance entre agents. Lorsque les délais de communication augmentent, le temps de convergence de l'algorithme augmente naturellement aussi.

La Figure 3.13 montre l'influence des paramètres de communication α et β sur le temps de convergence pour différentes valeurs du paramètre d'asynchronisme δ, dans le cas où les délais de communication sont déterministes, c'est-à-dire à $\sigma = 0$, et pour un facteur de régularisation $\gamma = 1$. Comme on peut le voir sur les quatre figures, le temps de convergence varie linéairement avec α et β.

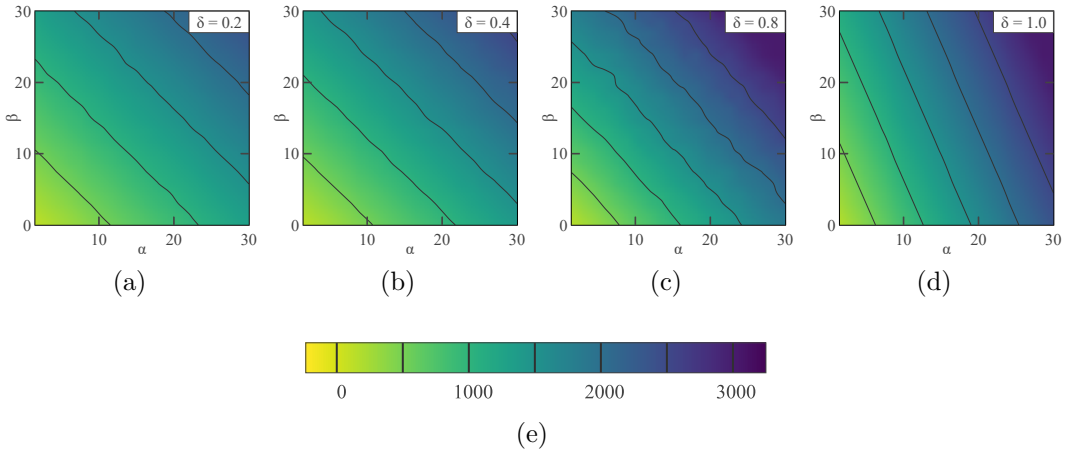


FIGURE 3.13 – Influence de l'état du réseau de communication, les paramètres α et β, sur le temps de convergence de l'algorithme de marché pair à pair asynchrone, pour différentes valeurs du paramètre d'asynchronisme δ. L'échelle des couleurs en 3.13e représente le temps de convergence en millisecondes.

Paramètre d'asynchronisme δ	$\frac{\partial \mathrm{CT}}{\partial \alpha}$ (*)	$\frac{\partial \mathrm{CT}}{\partial \beta}$ (*)	$\frac{\partial \mathrm{CT}}{\partial \alpha} / \frac{\partial \mathrm{TC}}{\partial \beta}$
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

(*) Valeur médiane sur tous les points calculés.

TABLEAU 3.5 – Gradient du temps de convergence (CT) selon les paramètres du modèle de communication α et β.

Le Tableau 3.5 présente les valeurs médianes du gradient du temps de convergence (CT) en fonction des paramètres de communication α et β. On rappelle que le terme moyen d'un délai de communication entre deux agents A et B est décrit comme : $\Delta T_{AB} = \alpha \cdot D_{AB} + \beta$ où D_{AB} désigne la distance entre les agents A et B. On observe dans la quatrième colonne

que le ratio entre les dérivées est à peu près égal à 1 excepté dans le cas synchrone à $\delta = 100\%$. Cela montre que les variations des paramètres α et β ont le même impact sur le temps de convergence final de l'algorithme en asynchrone. En revanche, dans le cas synchrone, le paramètre α a plus d'impact sur le temps de convergence que le paramètre β. Étant donné qu'en version synchrone, le temps d'attente entre chaque itération dépend du délai de réception du message le plus lent, on peut supposer que le temps de convergence global est directement proportionnel au délai de communication le plus élevé, c'est à dire à $\Delta T_{AB,max} = \alpha \cdot D_{AB,max} + \beta$. Ainsi, la variation du temps de convergence selon α est proportionnelle à la plus grande distance entre les agents $D_{AB,max}$, tandis que la variation du temps de convergence selon β est proportionnelle à 1. Si la distance $D_{AB,max}$ est supérieure à 1 alors cela explique la plus forte dépendance avec α, comme l'illustre la Figure 3.14. Dans le cas test utilisé ici, la distance $D_{AB,max}$ est bien supérieure à 1.

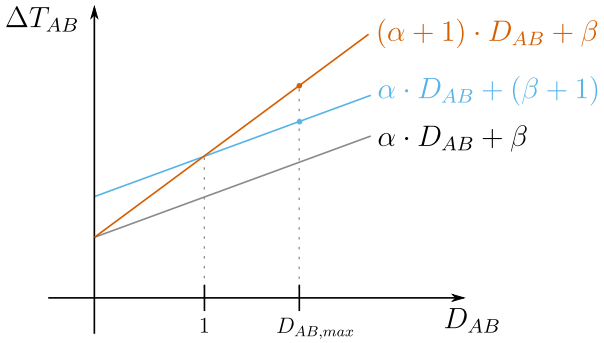


FIGURE 3.14 – Représentation de l'influence des variations des paramètres α et β sur les délais de communication. Dans le cas où la distance maximale entre deux agents du marché est supérieure à 1, la variation de α a alors plus d'impact sur le délai de communication maximal que la variation de β.

3.4.4 Influence du paramètre d'asynchronisme

Le paragraphe suivant se concentre sur l'influence du paramètre d'asynchronisme δ sur le temps de convergence à α et β fixes $\alpha = 5.0$ et $\beta = 1.0$, mais en ajoutant de la stochasticité dans les délais de communication. On se concentre d'abord sur le cas où les délais suivent une loi normale, de moyenne déterminée par α et β, et où l'écart-type σ varie.

Nota

Les valeurs exactes des paramètres de communication α et β ne sont pas pertinentes dans l'étude étant donné que l'on a effectué une étude sur l'influence des variations de ces paramètres précédemment. Les valeurs choisies ici sont arbitraires, et les résultats obtenus sont transposables à n'importe quelle valeur de α et β, étant donné que l'on compare les temps de convergence toujours par rapport au cas de référence synchrone. Les délais de communication sont donnés en unités de temps, et dans cette étude, une unité de temps représente 10 ms. Cette valeur a été choisie arbitrairement pour obtenir des délais de communication moyens de 60 ms, ce qui reste dans le même ordre de grandeur de communications par internet.

Les délais de communication sont maintenant tirés aléatoirement suivant le modèle gaussien décrit en section 2.1.2, ce qui rend le déroulement de l'algorithme stochastique : selon

les délais tirés et puisque chaque agent est déclenché par le nombre de messages qu'il a reçu, deux simulations ne présenteront pas les mêmes séquences de déclenchement ni les mêmes séquences de variables locales mises à jour. Ainsi, pour obtenir des conclusions pertinentes des différentes simulations, il est nécessaire de les lancer un grand nombre de fois et d'en déduire des valeurs statistiques, selon une simulation de Monte-Carlo. Le nombre de tirages pour chaque association de paramètres est de $N_{tirages} = 50$. On rappelle que pour chaque simulation, les délais de communication sont tirés à chaque nouvel envoi d'un message, donc le nombre de tirage des délais de communication est en réalité bien plus élevé que 50. Ce nombre est assez grand pour déterminer des statistiques précises, mais reste raisonnable pour limiter le temps de calcul de l'ensemble des simulations.

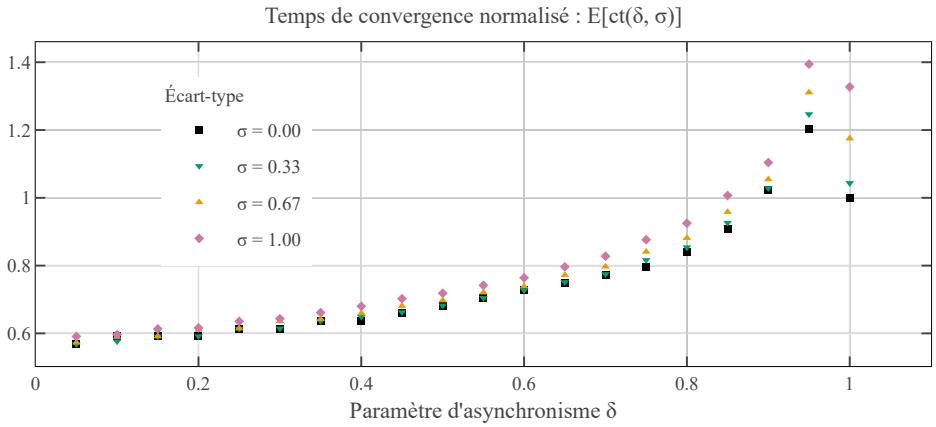


FIGURE 3.15 – Temps de convergence normalisé par rapport au temps de convergence synchrone et déterministe (à $\delta = 100\%$ et $\sigma = 0$), en fonction du paramètre d'asynchronisme δ et pour différentes valeurs de l'écart type sur les délais de communication σ.

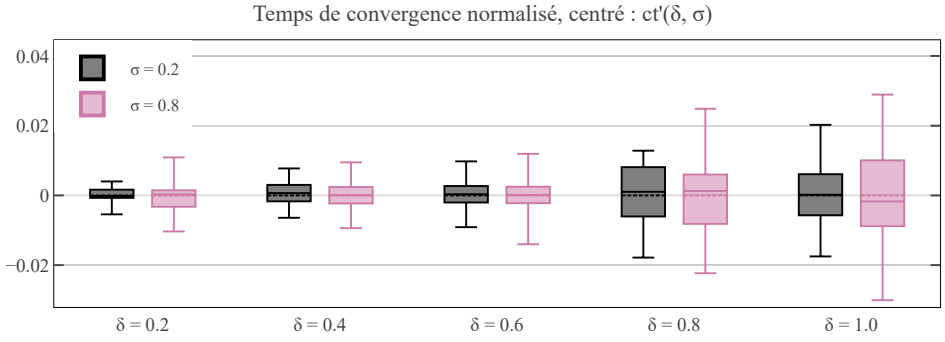


FIGURE 3.16 – Diagramme représentant l'étalement du temps de convergence en fonction du paramètre d'asynchronisme δ et pour deux valeurs de l'écart type sur les délais de communication σ.

On prend pour référence le temps de convergence de l'implémentation synchrone de l'algorithme, à $\delta = 100\%$, dans le cas où les délais de communication sont déterministes $\sigma = 0$. Le temps de convergence normalisé par ce temps de référence est une variable

stochastique définie en (3.17), et sa version centrée en zéro est définie en (3.18).

$$\mathrm{ct}(\delta,\,\sigma) = \frac{\mathrm{CT}(\delta,\,\sigma)}{\mathrm{CT}(\delta = 1,\,\sigma = 0)} \tag{3.17}$$

$$\mathrm{ct}'(\delta,\,\sigma) = \mathrm{ct}(\delta,\,\sigma) - \mathbb{E}\left[\mathrm{ct}(\delta,\,\sigma)\right] \tag{3.18}$$

Le temps de convergence de référence est égal à $\mathrm{CT}(\delta = 1,\,\sigma = 0) = 440$ unités de temps $= 4,4\,\mathrm{s}$. Étant donné que le délai de communication moyen est de 60 ms, et que les temps de calculs ont été négligés, cela équivaut à environ 73 échanges entre les agents avant d'atteindre la convergence de l'algorithme.

L'estimation de l'espérance de $\mathrm{ct}(\delta,\,\sigma)$ à partir des simulations réalisées est affichée en Figure 3.15 pour différentes valeurs du paramètre stochastique de communication σ. La Figure 3.16 représente les boîtes à moustache des variables centrées $\mathrm{ct}'(\delta,\,\sigma)$. On peut tirer plusieurs conclusions de ces figures.

— $\mathbb{E}(\mathrm{ct})$ est strictement croissant en fonction du paramètre d'asynchronisme δ, excepté lorsque l'on se rapproche du cas synchrone $\delta = 100\%$.

— $\mathbb{E}(\mathrm{ct}) < 1$ pour $\delta \leq 80\%$, ce qui signifie que ces cas asynchrones sont plus rapides que le cas synchrone à atteindre le même niveau de convergence. Cependant, $\mathbb{E}(\mathrm{ct}) > 1$ pour les cas où $\delta \geq 90\%$: le cas synchrone est plus rapide. Cela peut être expliqué par le fait que le temps d'attente entre chaque itération est quasiment égal au temps d'attente du cas synchrone, mais il est nécessaire de réaliser plus d'itérations pour compenser le manque d'information à chaque itération. Le temps d'attente gagné n'est pas suffisant pour contrer l'augmentation des itérations et ainsi le temps de convergence global est plus grand.

— $\mathbb{E}[\mathrm{ct}(\delta = 0, \sigma = 0)] \simeq 60\%$, ce qui correspond à une accélération de 40% du temps de convergence par rapport au cas synchrone, obtenu pour la plus petite valeur du paramètre d'asynchronisme δ, qui correspond à un déclenchement des calculs dès le premier message reçu à chaque itération. Une fois le calcul effectué, l'agent renvoie alors la variable mise à jour à son seul voisin dont il vient de recevoir un message.

— Pour un δ donné, $\mathbb{E}(\mathrm{ct})$ augmente avec le paramètre stochastique σ, c'est-à-dire avec l'augmentation des variations des délais de communication entre agents.

— La boîte à moustache de la Figure 3.16 montre que la dispersion du temps de convergence normalisé diminue lorsque δ diminue. Cela implique que le temps de convergence devient plus prévisible lorsque l'algorithme est plus asynchrone, et qu'il est alors moins affecté par les variations des délais de communication.

Une implémentation asynchrone de l'algorithme de marché pair à pair permettrait alors une résolution rapide du marché qui ne soit pas impactée par les variations des délais de communication. Il serait même possible de garantir un certain temps de convergence malgré ces variations étant donné qu'il n'y a pas beaucoup de dispersion sur le temps de convergence final lorsque le paramètre d'asynchronisme est proche de zéro, c'est-à-dire lorsque l'on déclenche une mise à jour locale dès la réception d'un message.

3.4.5 Modèle avancé de délais de communication

Le modèle de délai de communication utilisé jusqu'ici était le modèle gaussien, qui est plus rapide à calculer que le modèle plus réaliste présenté dans la section 2.1.2. Dans cette section, nous voulons comparer les résultats du temps de convergence et discuter des avantages et des inconvénients du modèle gaussien et du modèle plus avancé.

Pour ce faire, le paramètre de délai stochastique est fixé à $\sigma = 0,2$. Comme décrit précé-

demment, les paramètres du modèle sont choisis de manière à ce que la valeur moyenne et la variance des deux modèles correspondent. Les valeurs moyennes du temps de convergence normalisé sont affichées dans le tableau 3.6 pour différentes valeurs du paramètre asynchrone δ. Nous observons que les modèles de délai donnent des résultats similaires pour la version asynchrone de l'algorithme. Tout d'abord, le temps de convergence est largement sous-estimé lorsque l'on utilise le modèle gaussien au lieu du modèle avancé pour la version synchrone ($\delta = 1$). Cela peut s'expliquer par le fait que le modèle avancé a une longue queue de distribution, ce qui signifie que les délais plus longs ont plus de chance d'être échantillonnés qu'avec le modèle gaussien. Cela ralentit la résolution synchrone puisque les agents doivent attendre l'arrivée de tous les messages avant de pouvoir reprendre leurs calculs. Cela n'affecte pas autant la résolution asynchrone, car les délais les plus longs sont "coupés" par le passage à l'itération suivante, donc nous ne voyons pas de grande différence dans le temps de convergence en utilisant le modèle gaussien ou le modèle avancé.

δ	Modèle gaussien $\mathbb{E}[ct]$	Modèle avancé $\mathbb{E}[ct]$
20%	0.588	0.593
60%	0.714	0.722
100%	1.015	1.348

TABLEAU 3.6 – Comparaison du temps de convergence moyen normalisé entre les deux modèles de retard de communication, pour un paramètre de communication stochastique fixé $\sigma = 0.2$.

On affiche en Figure 3.17 la boîte à moustache du temps de convergence normalisé centré $\mathrm{ct}'(\delta, \sigma = 0.2)$ comparant les deux modèles de communication. On peut en tirer les mêmes conclusions que pour le Tableau 3.6 : l'utilisation du modèle gaussien donne des résultats similaires par rapport au modèle avancé, cependant il sous-estime la dispersion du temps de convergence dans le cas synchrone.

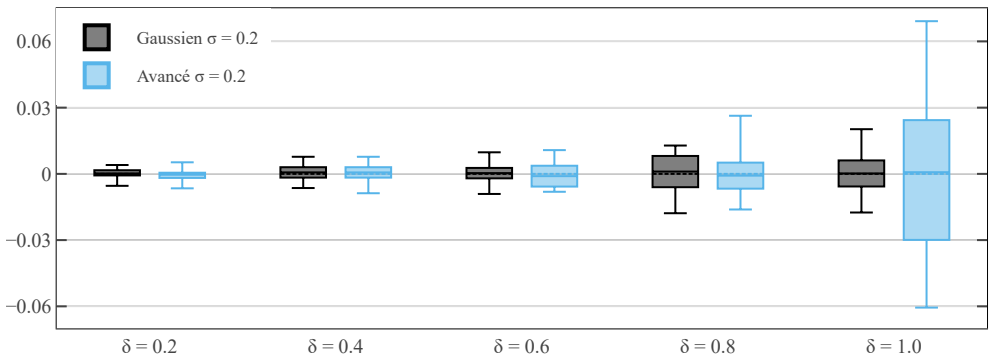


FIGURE 3.17 – Boîte à moustache du temps de convergence normalisé centré $\mathrm{ct}'(\delta, \sigma = 0.2)$ comparant le modèle gaussien et le modèle avancé.

En conclusion, le modèle gaussien de retard de communication est une bonne alternative au modèle plus avancé lors de la simulation de la résolution asynchrone de l'algorithme. Cependant, il sous-estime à la fois la valeur moyenne et la dispersion du temps de convergence de la résolution synchrone. Pour le reste de ce manuscrit, le modèle gaussien sera utilisé

mais nous devons garder à l'esprit que le temps de convergence plus réaliste de la résolution synchrone sera plus grand que celui approximé par le modèle gaussien.

3.4.6 Considérations sur le nombre de messages

On a montré en Figure 3.15 que plus l'algorithme était asynchrone, plus le temps de convergence diminuait par rapport au temps de convergence synchrone. Cependant, cette diminution se fait en contrepartie d'une augmentation du nombre total de messages échangés.

On définit de la même manière que précédemment le nombre de messages de référence comme étant :

$$n(\delta, \sigma) = \frac{N_{mes}(\delta, \sigma)}{N_{mes}(\delta = 1, \sigma = 0)} \tag{3.19}$$

avec N_{mes} le nombre de messages échangés pour atteindre la convergence de l'algorithme, et $N_{mes}(\delta = 1, \sigma = 0) = 158\,400$ messages, ce qui correspond à une moyenne de 66 messages par lien commercial. Le nombre de messages échangés peut être réduit en considérant d'autres critères de convergence comme dans [100].

On représente en Figure 3.18 le gain de temps de convergence en fonction de l'augmentation du nombre de messages échangés par rapport à la version synchrone. On remarque que le gain de temps en asynchrone vient au prix d'une augmentation des besoins en communication. A posteriori de cette étude, il est possible de définir un équilibre à trouver entre le temps de convergence et le nombre de messages échangés. En effet, le nombre de messages augmente considérablement à mesure que le temps de convergence diminue. Selon les capacités du réseau de communication, il est peut être préférable de choisir une valeur de δ plus élevée pour limiter le nombre de messages à échanger. Selon la Figure 3.18 par exemple, une diminution de 40% du temps de convergence se paie au prix d'une augmentation de 40% du nombre de messages échangés. On constate un palier de gain de temps de convergence à partir de 30% de messages échangés en plus par rapport à la version synchrone. Ainsi, il n'est pas nécessaire d'échanger plus de messages car le gain de temps de convergence est minime après 30%.

La Figure 3.18 montre aussi que l'augmentation du paramètre stochastique σ, qui représente la variation des délais de communication autour de leur valeur déterministe, n'augmente pas significativement le nombre de messages échangés.

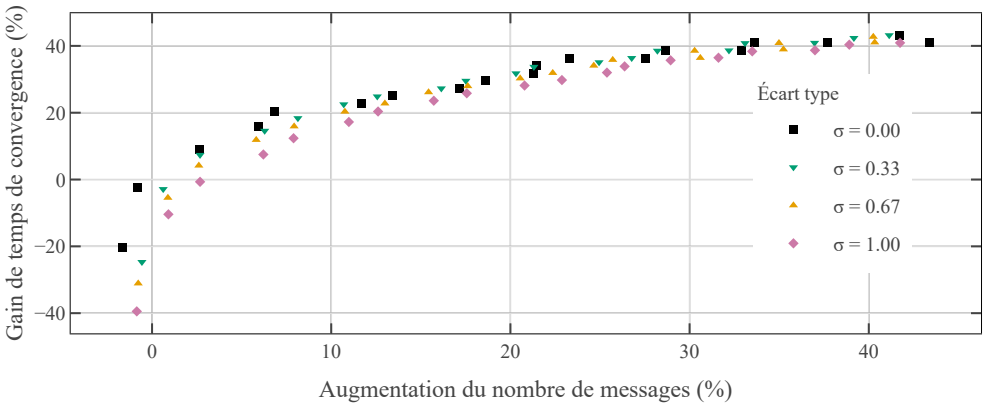


FIGURE 3.18 – Gain de temps de convergence en fonction de l'augmentation du nombre de messages échangés (par rapport à la solution synchrone déterministe). Le paramètre d'asynchronisme décroît lorsqu'on se déplace vers la droite de la figure.

3.4.7 Influence de la distance

Jusqu'ici nous avons étudié l'effet de l'état du réseau de communication sur le temps de convergence global seulement. Nous allons dans cette partie nous intéresser plus précisément aux valeurs locales des différentes variables d'échanges P_{ij} au moment de la convergence de l'algorithme.

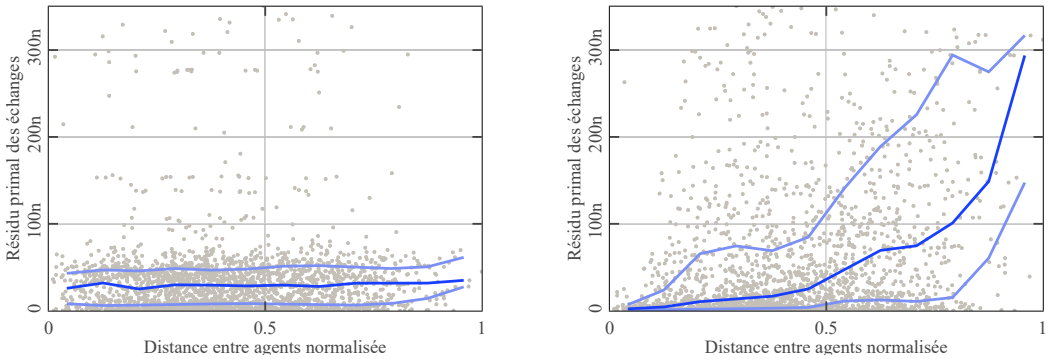
Pour chaque agent $i \in \mathcal{C}$ et leur partenaires commerciaux $j \in \mathcal{C}_i$, le résidu primal par échange (TR) est calculé au moment de la convergence globale de l'algorithme :

$$\mathrm{TR}(i,\, j) = \left(p_{i,j}^{t_{conv}} + p_{j,i}^{t_{conv}}\right)^2 \tag{3.20}$$

Cela correspond à un des termes du résidu primal donné en (3.8). Les différentes valeurs de $\mathrm{TR}(i,\, j)$ sont affichées en Figure 3.19 en fonction de la distance normalisée entre les agents i et j considérés :

$$d_{ij} = \frac{D_{ij}}{\max_{i',j'} D_{i'j'}} \tag{3.21}$$

avec D_{ij} présenté dans la section 2.1, représentant la distance entre les agents i et j [6].



(a) $\delta = 100\%$ (b) $\delta = 10\%$

FIGURE 3.19 – Résidu primal des échanges en fonction de la distance entre agents, comparaison entre la version synchrone $\delta = 100\%$ et la version asynchrone à $\delta = 10\%$. Les lignes bleues représentent le premier quartile, la médiane et le troisième quartile des résidus.

Dans le cas synchrone $\delta = 100\%$, la distance entre les agents n'a pas d'impact sur le résidu des échanges car quelque soit le délai de communication, tous les agents prennent en compte l'intégralité des messages à chaque itération.

Dans le cas asynchrone à $\delta = 10\%$, on observe que les agents plus proches ont des résidus plus petits que les agents plus éloignés. Cela s'explique par le fait que, parce qu'ils sont proches, leurs délais de communication sont les plus rapides, ils peuvent alors échanger plus de messages entre eux et atteindre un consensus plus rapidement. On remarque que la dispersion de ces résidus est aussi plus grande que dans le cas synchrone.

Puisque l'on considère que la convergence globale est atteinte lorsque la condition (3.11) est vérifiée, le fait que les agents les plus proches réduisent leurs résidus rapidement permet

6. On rappelle que la distance entre agents n'est pas forcément une distance géographique, mais peut aussi représenter le nombre d'équipements du réseau de communication qui se trouve entre les agents communicants.

à l'algorithme global de converger plus rapidement malgré le fait que les agents plus éloignés ont des résidus plus élevés. Cela montre que le critère d'arrêt tel que présenté dans [100] pourrait permettre aux agents plus proches d'arrêter de communiquer avant l'arrêt global de l'algorithme, et ainsi réduire le nombre de messages échangés.

Du point de vue du marché de l'électricité, on pourrait définir un critère de distance maximale avec les autres agents pour pouvoir participer au marché dans le but de garantir que les résidus soient en dessous d'un certain seuil.

3.4.8 Conclusions sur l'algorithme de marché pair à pair asynchrone simulé

La résolution du marché pair à pair, où les agents du marché échangent directement et de manière itérative les uns avec les autres, nécessite de nombreux échanges d'information. Or, ces échanges se font via un réseau de communication qui est susceptible de présenter des aléas de communication. Ainsi, l'implémentation réelle d'un tel algorithme à grande échelle pourrait être inenvisageable du fait des aléas de communication. On propose d'étudier une version asynchrone de cet algorithme qui permet de le rendre plus robuste aux aléas de communication. On a montré que l'asynchronisme permet de réduire les temps de convergence de 40% par rapport à la version synchrone de l'algorithme, en contrepartie d'une augmentation du nombre de messages échangés. De plus, le temps de convergence en asynchrone est moins impacté par les variations des délais de communication qu'en synchrone. Ceci permet de valider que le déploiement d'un marché pair à pair serait opérationnellement viable dans une situation réelle, sous son implémentation asynchrone.

Ces travaux peuvent s'appliquer à n'importe quel marché pair à pair. Or, le marché de l'énergie est opérationnellement couplé à un marché des capacités. Une des perspectives de ces travaux serait donc d'implémenter les méthodes proposées à un marché des capacités pair à pair [46] qui serait résolu en même temps que le marché classique.

3.5 Validation sur plateforme expérimentale

Le déploiement opérationnel des algorithmes proposés dans ce manuscrit nécessite une validation expérimentale. L'implémentation de l'algorithme de marché décentralisé pair à pair nécessite une plateforme matérielle pouvant représenter les différents agents de calcul impliqués dans le marché. Sur chacun de ces agents, il est nécessaire d'implémenter les différents protocoles qui entretiendront le bon déroulement de l'algorithme, de son initialisation à sa convergence, en prenant en compte les différentes phases itératives de calcul et de communication.

3.5.1 Plateforme opENS

Une plateforme expérimentale a été développée pour implémenter l'algorithme pair à pair asynchrone sur différentes machines communiquant entre elles. L'objectif final est de pouvoir implémenter l'algorithme sur le système *opENS* [101] en cours de développement sur le site du SATIE situé à l'ENS Rennes, qui consiste en une plateforme open source d'onduleurs triphasés destinée à simuler le comportement d'un smart-grid. Les différents constituants d'un smart-grid sont émulés par des nœuds opENS, composés chacun d'éléments de contrôle haut niveau (ordinateur Raspberry Pi) et bas niveau (DSP pour *Digital Signal Processor*, un microprocesseur optimisé pour les traitements numériques des signaux), ainsi que d'un onduleur triphasé qui s'occupe de l'échange physique de puissance avec le reste du réseau, comme illustré sur le schéma de la Figure 3.20.

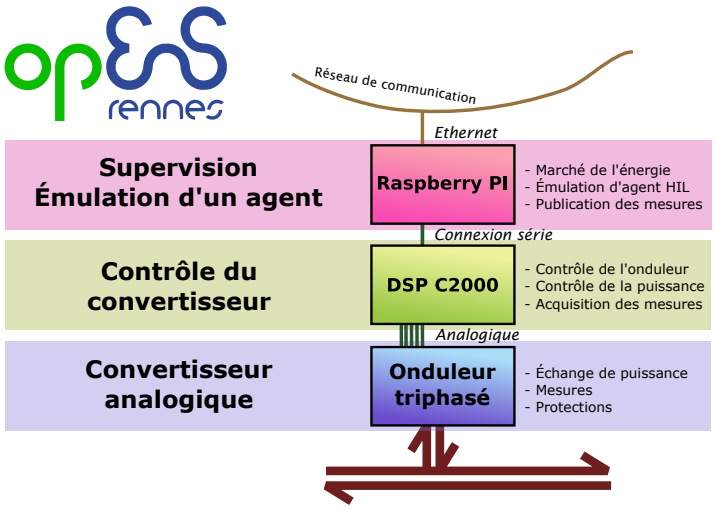


FIGURE 3.20 – Représentation hiérarchique d'un nœud opENS, de la partie supervision et émulation haut niveau prise en charge par le Raspberry Pi, à la partie d'échange de puissance physique assurée par l'onduleur triphasé.

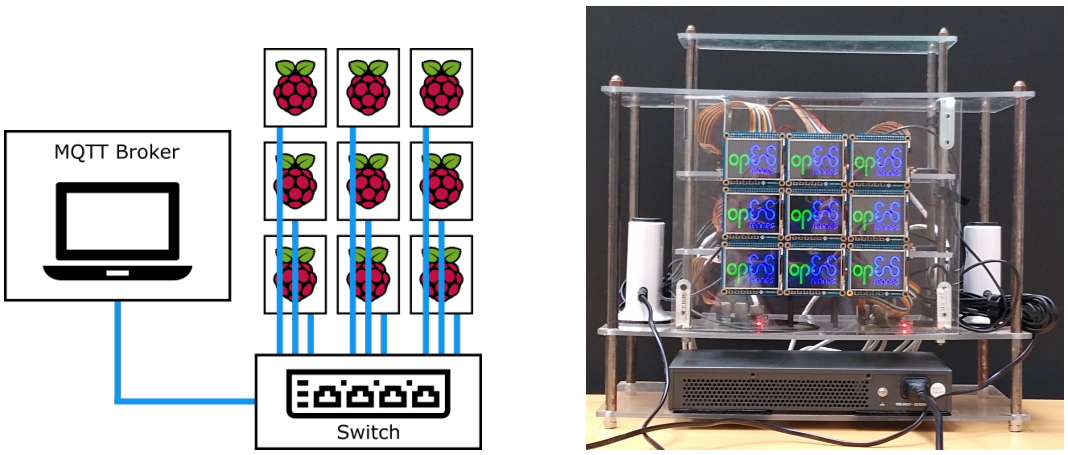
L'émulation d'un réseau électrique se fait en connectant plusieurs agents opENS entre eux, au niveau électrique en connectant les sorties triphasées ensemble, et au niveau applicatif en reliant les Raspberry Pis au même réseau Ethernet.

La plateforme ainsi constituée permet d'étudier différentes manières de gérer un réseau électrique via une approche transversale et pluri-disciplinaire. En effet, ils autorisent le

contrôle à la fois de variables de haut niveau telles que la puissance injectée sur le réseau, mais aussi des variables de très bas niveau telles que le courant, la tension ainsi que la fréquence de découpage de l'onduleur.

Le niveau supervision et émulation est géré par un ordinateur Raspberry Pi. Il est capable de se coordonner avec les autres nœuds ou bien un superviseur via un réseau Ethernet d'un côté, et communique avec la couche inférieure via une liaison série/USB. Il est aussi relié à un écran tactile permettant l'affichage en temps réel de diverses valeurs, ainsi que d'une série de boutons poussoirs qui permettent l'interaction avec l'utilisateur. Le DSP, contrôlant la couche inférieure, réalise le contrôle bas niveau ainsi que les mesures des valeurs de l'onduleur. Il remonte les valeurs filtrées au Raspberry Pi, et applique les boucles de contrôle pour obtenir des valeurs physiques commandés par la couche gestion.

L'implémentation de notre algorithme de marché décentralisé s'insère au niveau de la couche "Supervision et émulation d'un agent" et est implémenté en langage `Python`. Nous allons donc dans un premier temps nous concentrer sur l'implémentation logicielle de l'algorithme comprenant les phases de calcul et les échanges d'informations entre agents via la connexion Ethernet.



(a) (b)

FIGURE 3.21 – Schéma des éléments de la plateforme expérimentale à gauche, photo de la plateforme à droite. La photo n'inclue pas le broker MQTT.

3.5.2 Implémentation du marché pair à pair

On n'implémente que la partie gestion entre les Raspberry Pis, sans descendre jusqu'aux échanges réels de puissance entre les onduleurs, étant donné que ce n'est pas le but de l'étude. L'algorithme asynchrone de marché pair à pair a été implémenté sur 9 Raspberry Pis, représentant chacun un agent du marché. La communication entre agents est assurée par une liaison Ethernet, un commutateur ainsi qu'une machine servant de superviseur, représentés en Figure 3.21. Normalement, il n'y a pas besoin de cette machine étant donné que l'algorithme est totalement décentralisé, et n'a donc pas besoin d'un agent central. Elle sert cependant à faire du monitoring dans le contexte expérimental dans lequel nous nous trouvons.

Les codes du projet sont disponibles en accès libre sur Gitlab : [102]. Le Wiki du projet rentre en détails sur les outils et logiciels requis pour faire tourner la plateforme.

L'algorithme pair à pair asynchrone est implémenté de manière décentralisée sur 9 Raspberry Pis représentant chacun un agent du marché. Il y a trois producteurs et six consommateurs, illustrés en Figure 3.22.

— Les trois producteurs sont flexibles : $\underline{P_i} \leq P_i \leq \overline{P_i}$ avec $\underline{P_i} = 0$ pour les producteurs renouvelables et $\overline{P_i}$ étant égal à la puissance maximale pouvant être produite à l'instant de livraison considéré.
— Les deux voitures électriques à charger sont considérées comme des consommateurs totalement flexibles : $\underline{P_i} \leq P_i \leq 0$.
— Les trois habitations sont considérées comme des charges fixes : $\underline{P_i} = P_i = \overline{P_i} < 0$.
— L'agent 9 représentant un consommateur de type industriel possède une part de consommation fixe et une part flexible selon le prix de l'électricité : $\underline{P_i} \leq P_i \leq \overline{P_i} < 0$.

Le solveur utilisé pour résoudre les problèmes locaux des agents du marché est le solveur `OSQP` [98], identique à celui utilisé dans la partie simulation.

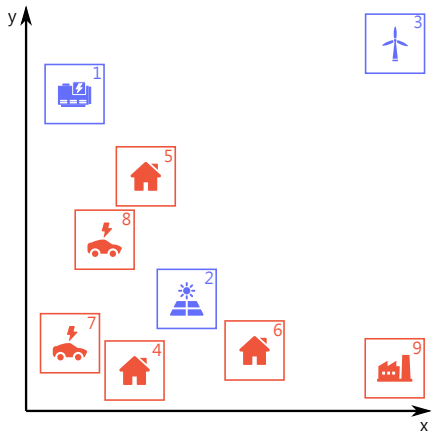


FIGURE 3.22 – Représentation géographique des agents implémentés sur la plateforme expérimentale.

Tous les producteurs ont pour partenaires commerciaux l'ensemble des consommateurs et vice-versa. Le paramètre d'asynchronisme δ est fixé avant de lancer les calculs. Les calculs continuent indéfiniment peu importe l'avancement de l'algorithme. Ainsi le marché tourne en continu malgré les possibles variations des points de fonctionnement des agents, telles que les puissances minimales et maximales ainsi que leurs fonction coût. On pourra vérifier par la suite que le marché global s'adapte à ces changements.

On présente par la suite le protocole de communication utilisé, ainsi que les délais de communication ajoutés pour simuler des distances plus ou moins grandes entre agents. Le monitoring de l'avancement de l'algorithme ainsi que l'enregistrement des données temporelles seront expliquées, de même que pour l'agencement global des différents scripts utiles au bon fonctionnement de la plateforme de résolution de marché.

Protocole de communication Le protocole de communication est imposé par la plateforme opENS : le protocole MQTT [103]. Il s'agit d'un protocole standard pour l'internet des objets (IoT) de type *publish-subscribe*, c'est-à-dire reposant sur des mécanismes de publications et d'abonnements, conçu pour être le plus léger possible et ne nécessiter qu'une faible bande passante.

Le mécanisme publish-subscribe (pub/sub) est une alternative au traditionnel mécanisme de client-serveur dans lequel le serveur communique directement avec le client. Ici, le modèle pub/sub permet de découpler le client qui envoie un message (appelé *publisher*) du ou des clients qui reçoivent le message (appelés *subscribers*). Les publishers et subscribers ne sont pas au courant de l'existence des uns et des autres et ne se contactent jamais directement. En effet, tous les messages passent par une entité centrale appelée *broker* qui collecte les messages des publishers et les redistribue aux subscribers. Les messages sont envoyés sous des *topics* différents. Les clients peuvent s'abonner à certains topics et pas d'autre et ainsi recevoir tous les messages des topics qui les intéresse. Le schéma de la Figure 3.23, issue de [104], représente les clients MQTT ainsi que le broker, échangeant sur un topic "speed".

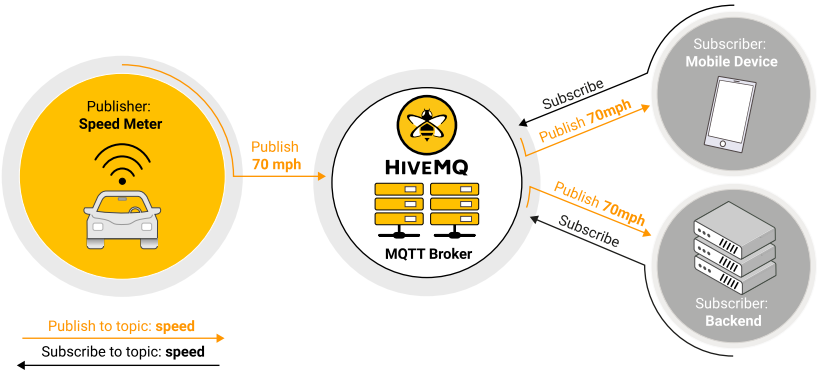


FIGURE 3.23 – Schéma représentant les échanges entre les clients et le broker dans le cadre du protocole MQTT. L'image est issue de [104]. Ici, les clients "Mobile Device" et "Backend" s'abonnent au topic "speed", tandis que le client publisher "Speed Meter" envoie un message dans le topic "speed" vers le broker. Le broker transfère alors le messages aux clients subscribers.

La centralisation des messages via le broker peut sembler aller à l'encontre de la philosophie de l'algorithme pair-à-pair où les pairs communiquent directement entre eux et n'ont pas besoin d'agent coordinateur central. Or, cette plateforme a un but de validation de concept, et nécessite donc des moyens de vérification et de surveillance de l'avancement de l'algorithme. Le fait que tous les messages échangés passent par une seule et même machine permet de suivre l'évolution des grandeurs de l'algorithme, notamment grâce à des outils de visualisation en temps réel tels que la suite InfluxDB.

Délais de communication Des délais sont rajoutés juste avant l'envoi effectif des messages pour simuler des délais de communications qui varient selon le destinataire, et qui suivent le modèle explicité en section 2.1. Les paramètres de délais de communication α et β sont réglés de manière à obtenir une valeur moyenne des délais de communication à 500 ms. À ce délai simulé s'ajoute le réel délai de communication de l'ordre de 20 ms [7]. On teste alors trois valeurs du paramètre de délais de communication $\alpha \in \{0, 0.5, 1\}$. L'histogramme des délais de communication est affiché en Figure 3.24

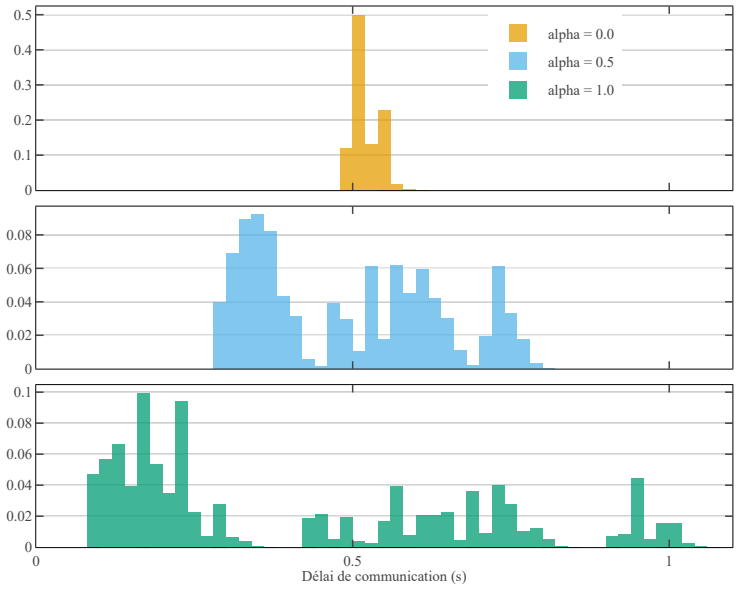


FIGURE 3.24 – Distribution empirique des délais de communication pour trois valeurs différentes du paramètre de délai de communication α.

Monitoring de l'avancement de l'algorithme Le monitoring des différentes grandeurs échangées lors des différentes itérations de l'algorithme pair à pair est effectué grâce à la suite d'outils proposés par InfluxData [105] et dont le fonctionnement est représenté en Figure 3.25.

L'outil Telegraf est utilisé comme client MQTT connecté au broker et qui s'abonne à certains topics configurés au préalable. Les données ainsi collectées sont enregistrées dans la base de données temporelle InfluxDB. Il est possible d'effectuer des calculs sur ces données à l'aide de l'outil Kapacitor si nécessaire. Enfin, la visualisation graphique et en temps réel [8] des données collectées se fait au travers de l'outil Chronograf. Une capture d'écran de la fenêtre Chronograf est affichée en Figure 3.26. L'interface graphique permet aussi d'enregistrer les données temporelles en format CSV pour pouvoir ensuite les traiter.

7. Cette valeur peut paraître élevée pour un délai de communication ayant lieu sur le même sous réseau, mais il englobe l'intégralité des délais ajoutés par le système d'exploitation et les temps de traitement du programme python.
8. Les données peuvent être mises à jour toutes les 5 secondes.

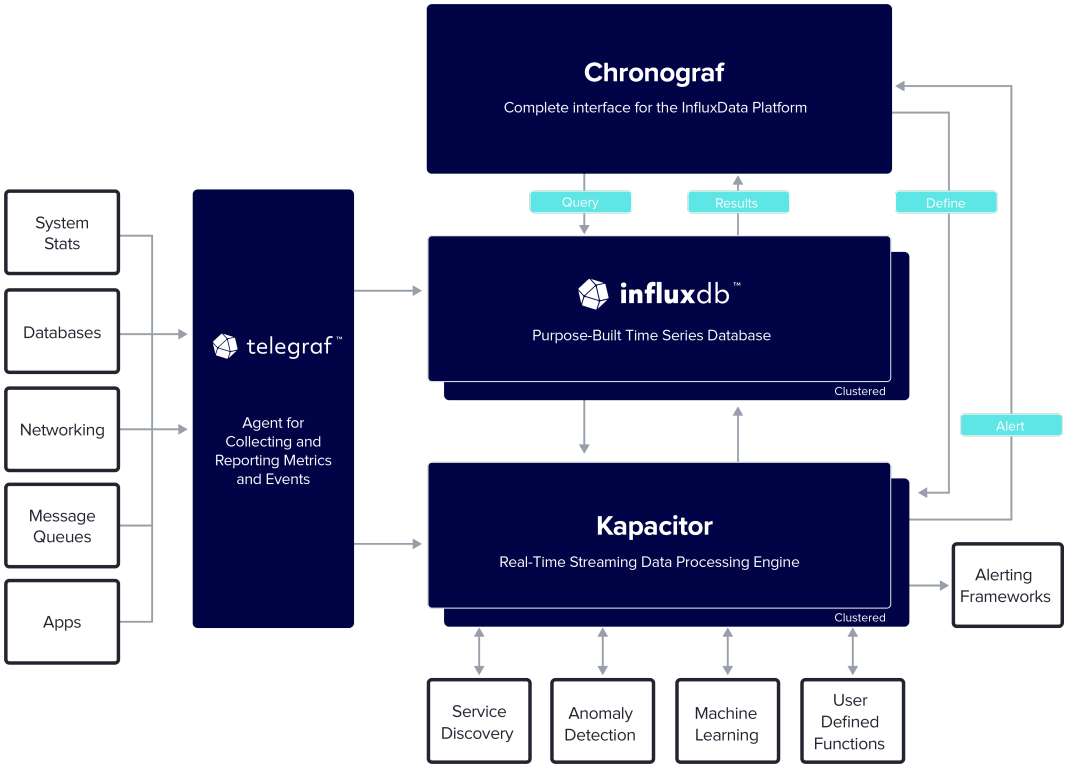


FIGURE 3.25 – Schéma de l'organisation entre les différents outils de la suite InfluxData [105].

FIGURE 3.26 – Capture d'écran de la fenêtre Chronograf, paramétrée pour afficher les valeurs de l'avancement global de l'algorithme en fonction du temps : la puissance de chaque agent, les variables duales "price", le coût global, ainsi que les différents résidus de l'algorithme ADMM. Dans cette capture, la fonction coût d'un des agents a été modifiée pendant la résolution, avant d'être rétablie à sa valeur initiale. On observe que le marché s'adapte rapidement à ce changement.

Mécanisme de détection de présence Pour pouvoir vérifier qu'un agent est toujours présent dans le marché pendant les calculs itératifs, indépendamment de sa fréquence d'envoi de messages, on instaure un système de "battements de cœur/heartbeat". Chaque agent envoie à fréquence fixe un message sur un topic global *heartbeat* écouté par l'intégralité des agents du marché. Cela permet aux agents de vérifier que leurs partenaires commerciaux sont toujours connectés au réseau de communication. Au bout d'un certain timeout, supérieur à plusieurs fois la période du battement de cœur, si un agent n'a toujours pas envoyé son message sur le topic "heartbeat", ses partenaires commerciaux peuvent en déduire qu'il est déconnecté du réseau de communication.

La détection de l'absence d'un partenaire commercial peut entraîner plusieurs réactions au choix.

— Soit l'algorithme continue sans prendre en compte les messages du partenaire absent : cela revient alors à un fonctionnement asynchrone, et n'a d'impact que si l'on considère l'algorithme synchrone. La puissance échangée avec le partenaire absent est estimée comme étant la dernière valeur échangée avant que l'agent ne se déconnecte.
— Soit l'on considère que l'agent ne participe plus au marché et n'injecte ni ne consomme plus de puissance. Les agents partenaires modifient alors leurs points de fonctionnement pour trouver le nouvel optimum. Si un agent absent communique à nouveau via le topic "heartbeat", alors les échanges avec ses partenaires commerciaux peuvent reprendre là où ils s'étaient arrêtés.

Dans le second cas, on pourrait parler d'un comportement "plug and play" [9].

Machine à états Chaque agent implémenté sur un Raspberry Pi fait tourner son propre script python permettant d'initialiser et de faire avancer l'algorithme de marché pair à pair. Une machine à état, représentée en Figure 3.27, est implémentée pour gérer l'initialisation ainsi que le déroulement des différentes phases itératives de l'algorithme.

— **GlobalInit** : initialisation des différents buffers locaux servant à la réception et à l'envoi de messages via le protocole MQTT, initialisation des flags et des variables locaux.
— **BrokerConnection** : création de la connexion avec le broker, inscription aux différents topics associés aux variables nécessaires à la résolution du problème local.
— **AgentInit** : initialisation de la communication entre agents, notamment via le topic "heartbeat". Chaque agent envoie son propre heartbeat et l'algorithme peut commencer une fois que tous les agents ont détecté la présence de l'ensemble des participants au marché. Cela évite que des messages soient envoyés avant que les agents ne soient abonnés aux bons topics. Ensuite, calcul de la première itération du problème local et envoi des résultats aux partenaires commerciaux concernés.
— **Stall** : attente de l'arrivée des messages des partenaires commerciaux. Dans le cas synchrone, on attend autant de messages que de partenaires commerciaux. Dans le cas asynchrone, on n'attend un nombre donné de messages.
— **Computation** : résolution du problème local avec les données mises à jour provenant des messages reçus.
— **Sendback** : envoi des nouveaux résultats aux partenaires commerciaux.
— **Pause** : l'appui sur un bouton de l'écran provoque un changement d'état du flag "pause_flag" qui peut mener à l'état "Pause". Dans cet état, on désactive le heartbeat

9. La façon dont chaque agent est codé ne permet pas pour l'instant le vrai comportement "plug and play" dans le sens où tous les agents participant au marché sont déjà connus à l'avance. L'ajout d'un protocole d'annonce de participation au marché pourrait permettre à un agent extérieur au marché d'y rentrer en cours de route.

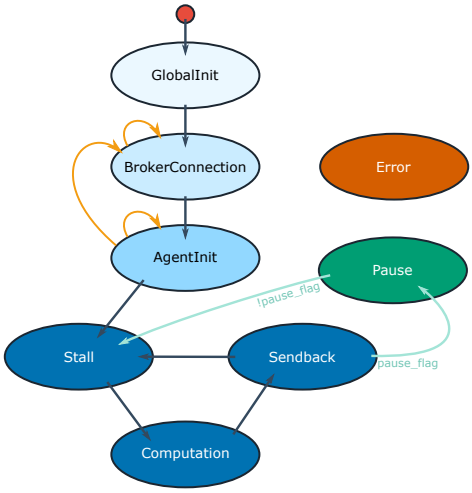


FIGURE 3.27 – Machine à état implémentée dans chaque machine de calcul. Les trois premiers états *GlobalInit*, *BrokerConnection* et *AgentInit* sont des états d'initialisation, les états *Stall*, *Computation* et *Sendback* sont les états itératifs de l'algorithme ADMM. Il est possible via une commande externe d'entrer dans un état *Pause* qui met en pause les calculs de l'agent local. Toute erreur interne au code amène la machine à état dans l'état *Error*.

et on attend que le flag "pause_flag" repasse à zéro de nouveau par l'appui sur un bouton.

— **Error** : depuis n'importe quel autre état, si une erreur logicielle survient, la machine à états se place dans l'état "Error".

Structuration du code La structure du code de chaque ordinateur est affichée en Figure 3.28. On y représente les différents scripts et leurs interactions.

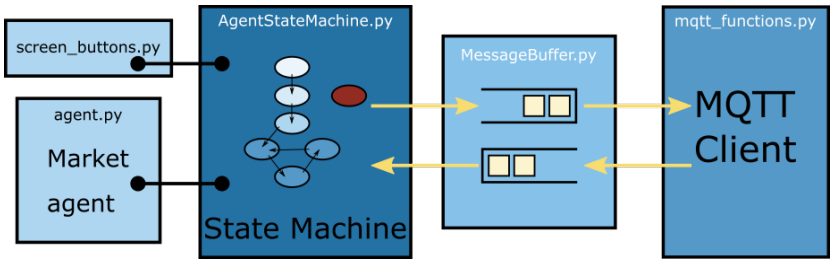


FIGURE 3.28 – Structure des différentes fonctions implémentées sur un Raspberry Pi. Le code propre à la résolution du problème local est inclus dans "Market agent". Lorsqu'un message est reçu ou bien un message est prêt à être envoyé, il passe par un buffer dans "MessageBuffer". "MQTT Client" s'occupe d'encapsuler/désencapsuler les messages à envoyer/reçus au travers du protocole MQTT. L'appui sur les boutons de l'écran est géré par le script "screen_buttons.py".

La machine à états présentée plus haut fonctionne de pair avec les fonctions propres aux calculs locaux de marché contenus dans le "Market agent" et les fonctions prenant en compte l'appui sur les différents boutons de l'écran sont contenus dans le script "screen_buttons.py". C'est via l'appui des boutons que l'on peut contrôler en temps réel les points de fonctionne-

ment des agents du marché. L'interface de communication se trouve dans le script "MQTT Client". À la réception d'un message, il s'occupe de désencapsuler le message de son protocole MQTT, et le place dans le buffer entrant. Au contraire, pour envoyer un message placé dans le buffer sortant, il l'encapsule pour ensuite l'envoyer au broker.

3.5.3 Résultats temporels

Influence du paramètre d'asynchronisme sur la vitesse de convergence L'algorithme a été lancé pour plusieurs cas de figures différents : en faisant varier les délais de communication, donc le paramètre α comme précisé dans un paragraphe précédent et illustré en Figure 3.24, et en faisant varier le paramètre d'asynchronisme δ.

Nota Bien que la fonctionnalité permettant de modifier les valeurs des fonctions coût locales durant la résolution du marché ait été implémentée, on ne l'utilise pas dans les essais présentés.

Le résidu primal en fonction du temps est affiché en Figure 3.29. On y observe une décroissance du résidu primal, peut importe les paramètres choisis. Cependant, certains cas de figure sont plus rapides à converger que d'autres.

Tout d'abord, la vitesse de convergence de l'algorithme est plus lente lorsque le paramètre de délais de communication α est plus grand c'est-à-dire lorsque les délais de communication sont plus variés les uns par rapport aux autres. On rappelle que le délai de communication moyen est constant et égal à 500 ms quelque soit la valeur de α. Cela signifie qu'à délai de communication moyen donné, la disparité des délais de communication influence le temps de convergence global. Lorsque les délais sont homogènes, l'algorithme converge plus rapidement.

Ensuite, la vitesse de convergence dépend aussi du paramètre d'asynchronisme δ. Dans le cas observé, la solution synchrone à $\delta = 100\%$ est plus rapide que n'importe quelle solution asynchrone lorsque les délais de communication sont homogènes ($\alpha = 0$). Elle est cependant rattrapée lorsque les délais de communication deviennent hétérogènes par les cas asynchrones à $\delta = 10\%$ pour $\alpha = 0.5$ et $\delta \in \{10, 40\}\%$ pour $\alpha = 1.0$ lorsque l'on considère la convergence pour un résidu primal de 10^{-6}.

Le gain en temps de convergence de la version asynchrone par rapport à la version synchrone n'est pas aussi visible que dans le cas simulé à 110 agents de la section 3.4.4. Cela est dû au faible nombre d'agents sur le marché, donc du faible nombre d'agents partenaires commerciaux. La version asynchrone permet de réduire les délais d'attente des messages en n'attendant qu'une partie donnée de ces messages. Cependant, si le nombre maximal de messages attendus est petit, alors cela limite le gain de temps apporté dans la version asynchrone.

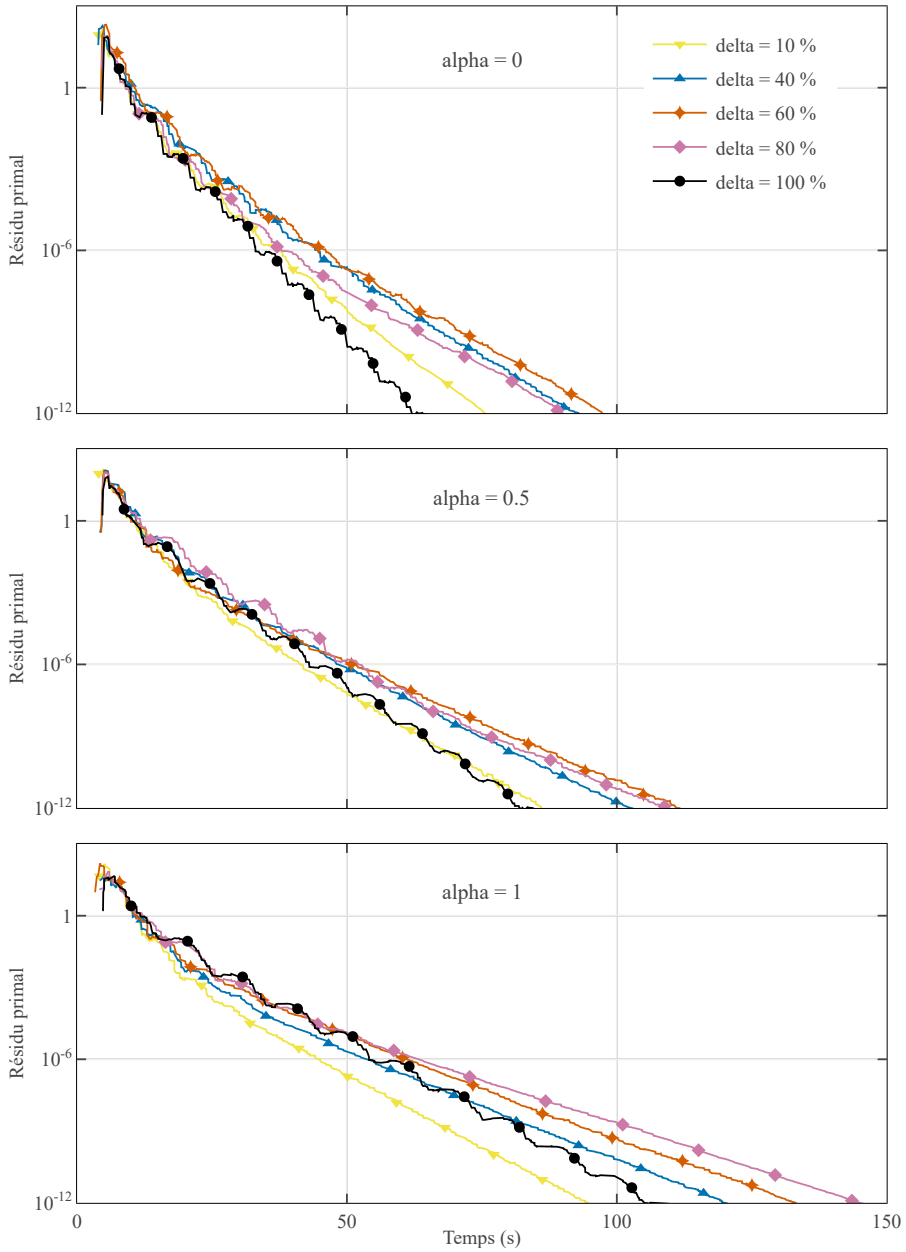


FIGURE 3.29 – Résidu primal global en fonction du temps, pour trois valeurs du paramètre de délais de communication α, et pour différentes valeurs du paramètre d'asynchronisme δ. Le cas synchrone $\delta = 1$ est affiché en noir.

Mise en pause d'un agent Cette partie s'intéresse au cas où un agent est momentanément déconnecté du réseau de communication. Le cas d'étude considéré consiste en la déconnexion de l'agent 7 arbitrairement choisi au bout de trente secondes après le commencement de la résolution, et sa reconnexion quinze secondes plus tard. On se place dans le cas où $\alpha = 0$, c'est-à-dire où les délais de communication ajoutés sont tous égaux à 500 ms.

Les cas synchrone et asynchrone pour différentes valeurs du paramètre d'asynchronisme δ est considéré. La Figure 3.30 représente le résidu primal en fonction du temps.

Pour le cas synchrone (en noir à marqueur rond sur la figure), le résidu primal global se fige pendant toute la durée de déconnexion, tandis que pour les cas asynchrones, les agents ne communiquent plus avec l'agent déconnecté mais peuvent toujours communiquer entre eux et continuer à faire décroître le résidu primal global. On remarque cependant un ralentissement de cette diminution, qui peut être expliqué par le fait que la partie du résidu primal de chaque agent partenaire de l'agent 7 : $t_{i7} + t_{7i}$ ne décroît pas.

À la reprise normale des communications, le cas synchrone reprend alors là où il s'était arrêté. Dans cette configuration du réseau de communication, la version synchrone est bien plus rapide que les versions asynchrones ce qui explique que le résidu primal reste en général plus petit qu'en asynchrone. Cependant, on remarque que le cas asynchrone $\delta = 0.1$ est équivalent au cas synchrone là où il était plus lent en Figure 3.29 là où il n'y avait pas de déconnexion d'un agent.

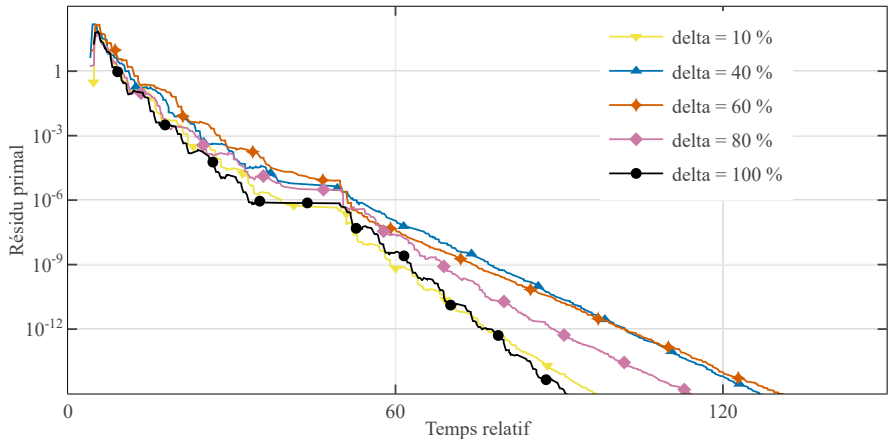


FIGURE 3.30 – Résidu primal global en fonction du temps, et pour différentes valeurs du paramètre d'asynchronisme δ, avec $\alpha = 0$. L'agent 7 est mis en pause au bout de 30 secondes après le début des calculs, et repart 15 secondes plus tard.

3.5.4 Conclusions sur la plateforme expérimentale

L'implémentation expérimentale de l'algorithme de marché pair à pair est essentielle pour déterminer la faisabilité empirique de ce genre d'algorithme décentralisé. Nous avons mis en place une plateforme expérimentale limitée à 9 agents communiquant via un protocole MQTT sur un même réseau local, et implémentant l'algorithme de marché pair à pair en versions synchrone et asynchrone. Cette plateforme a permis de mettre en œuvre des tests empiriques de résolution de marché ainsi que la supervision de l'avancement de l'algorithme au travers d'un logiciel de visualisation temps réel des données échangées.

On a pu montrer que la version asynchrone de l'algorithme permet de pallier les ralen-

tissements de l'algorithme dus aux variations de délais de communication, même dans un cas limité à 9 agents. La version asynchrone permet aussi à l'algorithme de continuer sa résolution malgré la déconnexion d'un des agents du marché. En revanche, la diminution du résidu primal est limitée car les termes liés à l'agent déconnecté ne sont pas mis à jour durant la période de déconnexion.

En perspectives de ces travaux, il serait intéressant d'implémenter un marché "en continu" c'est-à-dire un marché où les fonctions coût et les limites de puissance des agents peuvent changer au cours de la résolution. Il serait alors nécessaire de déterminer des phases de "régime transitoire" juste après un changement de point de fonctionnement où le marché est encore en train de retrouver un équilibre et de "régime permanent" une fois que les résidus se retrouvent en dessous d'une certaine valeur et que les puissances se trouvent en équilibre. Une telle implémentation ressemblerait à une résolution en *warm start*, c'est-à-dire une résolution ayant pour conditions initiales le point d'équilibre de l'instant précédent.

L'implémentation logicielle pourrait aussi bénéficier d'un couplage à un réseau de puissance réel, comme le permet la plateforme opENS présentée en début de section. De cette manière, on pourrait étudier l'influence des résultats du marché sur la partie puissance, mais aussi l'influence des mesures physiques de puissance sur le marché électrique.

La faisabilité empirique de l'algorithme décentralisé considéré pourrait bénéficier d'une implémentation sur un grand nombre d'agents. Il serait alors nécessaire de connecter beaucoup plus de Raspberry Pis via un réseau de communication, qui lui ne serait pas forcément limité à un réseau local. En effet, répartir les agents sur plusieurs sous-réseaux connectés via Internet pourrait mettre en avant des topologies de communication plus réalistes et faire varier d'autant plus les délais de communication. Ainsi l'on pourrait étudier les effets de l'asynchronisme sur un tel cas d'étude.

3.6 Conclusions et perspectives du chapitre

La résolution du marché pair à pair, où les agents du marché échangent directement et de manière itérative les uns avec les autres, nécessite de nombreux échanges d'information. Or, ces échanges se font via un réseau de communication qui est susceptible de présenter des aléas de communication. Ainsi, l'implémentation réelle d'un tel algorithme à grande échelle pourrait être inenvisageable du fait des aléas de communication. La version asynchrone de cet algorithme, qui permet de le rendre plus robuste aux aléas de communictation a été étudié dans ce chapitre au travers de deux méthodes : une méthode de simulation utilisant la plateforme présentée au Chapitre 2, et une méthode expérimentale.

On a montré au travers de la plateforme de simulation que l'asynchronisme permet de réduire les temps de convergence de 40% par rapport à la version synchrone de l'algorithme, en contrepartie d'une augmentation du nombre de messages échangés. De plus, le temps de convergence en asynchrone est moins impacté par les variations des délais de communication qu'en synchrone. Ceci permet de valider que le déploiement d'un marché pair à pair serait opérationnellement viable dans une situation réelle, sous son implémentation asynchrone.

Nous avons mis en place dans un second temps une plateforme expérimentale limitée à 9 agents communiquant via un protocole MQTT sur un même réseau local, et implémentant l'algorithme de marché pair à pair en versions synchrone et asynchrone. Cette plateforme a permis de mettre en œuvre des tests empiriques de résolution de marché ainsi que la supervision de l'avancement de l'algorithme au travers d'un logiciel de visualisation temps réel des données échangées. On a pu montrer que la version asynchrone de l'algorithme permet de pallier les ralentissements de l'algorithme dus aux variations de délais de communication,

même dans un cas limité à 9 agents. La version asynchrone permet aussi à l'algorithme de continuer sa résolution malgré la déconnexion d'un des agents du marché. En revanche, la diminution du résidu primal est limitée car les termes liés à l'agent déconnecté ne sont pas mis à jour durant la période de déconnexion.

En perspective de ces travaux, on pourrait appliquer le principe d'asynchronisme au marché des capacités pair à pair [46] qui serait résolu en même temps que le marché classique. Il serait aussi intéressant d'implémenter un marché "en continu", c'est-à-dire un marché dont la résolution ne finit jamais, où les fonctions coût et les limites de puissance des agents peuvent changer au cours de la résolution. Il serait alors nécessaire de déterminer des phases de "régime transitoire" juste après un changement de point de fonctionnement où le marché est encore en train de retrouver un équilibre et de "régime permanent" une fois que les résidus se retrouvent en dessous d'une certaine valeur et que les puissances se trouvent en équilibre. Une telle implémentation ressemblerait à une résolution en *warm start*, c'est-à-dire une résolution ayant pour conditions initiales le point d'équilibre de l'instant précédent.

L'implémentation logicielle pourrait aussi bénéficier d'un couplage à un réseau de puissance réel, comme le permet la plateforme opENS par exemple. De cette manière, on pourrait étudier l'influence des résultats du marché sur la partie puissance, mais aussi l'influence des mesures physiques de puissance sur le marché électrique.

La faisabilité empirique de l'algorithme décentralisé considéré pourrait bénéficier d'une implémentation sur un grand nombre d'agents. Il serait alors nécessaire de connecter beaucoup plus de Raspberry Pis via un réseau de communication, qui lui ne serait pas forcément limité à un réseau local. En effet, répartir les agents sur plusieurs sous-réseaux connectés via Internet pourrait mettre en avant des topologies de communication plus réalistes et faire varier d'autant plus les délais de communication. Ainsi l'on pourrait étudier les effets de l'asynchronisme sur un tel cas d'étude.

Chapitre 4

Optimal Power Flow décentralisé asynchrone

Sommaire

Le problème du marché de l'électricité étudié dans le chapitre précédent ne prend pas en compte les contraintes physiques du réseau électrique. Le calcul de *power flow* est historiquement réalisé par l'opérateur réseau après la résolution de marché de l'électricité pour vérifier que toutes les contraintes physiques sont respectées. Si ce n'est pas le cas, ce dernier incite les agents du marché à déplacer leurs points de fonctionnement jusqu'à ce que la solution respecte les contraintes réseaux. Cette méthode nécessite donc plusieurs allers-retours entre l'opérateur de marché et l'opérateur réseau pour trouver un point de fonctionnement faisable pour un instant futur donné.

Le calcul de l'Optimal Power Flow (OPF) permet en une seule étape de trouver la solution optimale tant du point de vue économique des agents du marché, mais aussi du point de vue du réseau électrique en s'assurant de la faisabilité de la solution. Il introduit les variables de tensions complexes ainsi que de puissances complexes dans les lignes, et s'assure que ces variables restent dans des intervalles imposés.

L'OPF est jusqu'ici réalisé de manière centralisée, c'est-à-dire par un unique agent. Depuis la séparation du marché de l'électricité et des gestionnaires de réseaux, il n'est plus possible de considérer qu'une unique entité possède l'ensemble des informations nécessaires à la résolution du problème d'OPF. En effet, les données des fonctions coût ainsi que des limites de puissances des producteurs et consommateurs doivent être communiquées si l'on veut résoudre l'OPF de manière centralisée, ce qui porte atteinte à la confidentialité des données de chaque agent. De plus, avec l'avènement des énergies renouvelables distribuées et de communautés énergétiques diverses, le nombre d'acteurs pouvant interagir au travers du réseau électrique est amené à augmenter de manière significative, ce qui rend le calcul d'OPF centralisé plus compliqué. La décentralisation de l'OPF permet de pallier ces problématiques en rendant possible le passage à l'échelle, pour prendre en compte un réseau électrique à grande échelle, ainsi que la confidentialité des données de chaque agent. En revanche, elle né-

cessite de nombreuses itérations et de la communication entre les agents à chaque itération. Cela rend la résolution décentralisée très sensible aux aléas de communication.

Une version asynchrone est alors proposée pour chaque algorithme d'OPF décentralisé, qui rend la résolution plus robuste aux aléas de communication. Ainsi, les versions asynchrones des algorithmes présentés permettent une implémentation réelle moins sensible aux différents aléas de communication.

Après avoir présenté le problème d'Optimal Power Flow en section 4.1, la résolution centralisée sera explicitée en section 4.2 en soulignant ses insuffisances pour une implémentation à grande échelle. Le principe de décomposition du problème pour sa résolution décentralisée sera présenté en section 4.3. Nous présenterons ensuite deux algorithmes décentralisés, l'un décomposé par nœuds en section 4.4 et l'autre décomposé par frontières en section 4.5. Pour chacun de ces algorithmes décentralisés, les versions asynchrones seront introduites et comparées aux versions synchrones pour montrer l'intérêt des implémentations asynchrones dans diverses conditions du réseau de communication et divers cas de figure en ce qui concerne le partitionnement du réseau.

4.1 Formalisation du problème

Le problème de l'Optimal Power Flow consiste à minimiser les coûts de tous les consomacteurs du réseau, comme dans le problème du marché de l'électricité, tout en déterminant les valeurs des variables physiques du réseau électrique et en s'assurant de la faisabilité de la solution.

Ainsi, les variables considérées dans la résolution de l'OPF sont les suivantes :
— les tensions complexes en chaque noeud n du réseau, V_n,
— les puissances complexes des consomacteurs i, $S_i = P_i + jQ_i$.
La formulation de l'OPF est décrite en (4.1).

Formulation de l'Optimal Power Flow

$$\min \sum_{i \in \mathcal{C}} f_i(P_i) \tag{4.1a}$$

$$\begin{aligned}
&\text{selon } V_n \in \mathbb{C}, && n \in \mathcal{N} \\
&\quad S_i = P_i + jQ_i \in \mathbb{C} && i \in \mathcal{C} \\
&\quad S_{nm} \in \mathbb{C} && n \in \mathcal{N},\ m \in \mathcal{N}_n \\
&\text{tel que } \underline{V_n} \leq |V_n| \leq \overline{V_n} && n \in \mathcal{N} && \text{(4.1b)} \\
&\quad \underline{S_i} \leq S_i \leq \overline{S_i} && i \in \mathcal{C} && \text{(4.1c)} \\
&\quad \sum_{i \in \mathcal{C}_n} S_i = \sum_{m \in \mathcal{N}_n} S_{nm} && n \in \mathcal{N} && \text{(4.1d)} \\
&\quad S_{nm} = Y_{nm}^* V_n (V_n - V_m)^* && n \in \mathcal{N},\ m \in \mathcal{N}_n && \text{(4.1e)} \\
&\quad \underline{S_{nm}} \leq |S_{nm}| \leq \overline{S_{nm}} && n \in \mathcal{N},\ m \in \mathcal{N}_n && \text{(4.1f)} \\
&\quad \underline{\theta_{nm}} \leq \angle(V_n V_m^*) \leq \overline{\theta_{nm}} && n \in \mathcal{N},\ m \in \mathcal{N}_n && \text{(4.1g)} \\
&\quad \angle V_n = 0 && n \in \mathcal{N}_{ref} && \text{(4.1h)}
\end{aligned}$$

Nota

Il existe des formulations diverses selon le niveau de modélisation du réseau. Ici, on considère un modèle de ligne avec une simple admittance, mais on pourrait aussi prendre en compte un modèle en pi comme dans `matpower`[94], ou bien un transformateur suivi d'un modèle en pi asymétrique comme dans `PowerModels`[106].

Les équations (4.1b)-(4.1c) représentent respectivement les contraintes sur l'amplitude de la tension à chaque nœud et sur la puissance apparente complexe de chaque consomacteur. L'équation (4.1d) décrit la contrainte de conservation du flux en chaque nœud, avec S_{nm} représentant la puissance complexe s'écoulant depuis le nœud n vers le nœud m, dont la définition est donnée en (4.1e), et dont l'amplitude est limitée en (4.1f). La différence d'angle de la tension entre deux nœuds reliés par une ligne est limitée en (4.1g). Enfin, l'angle du nœud de référence est fixé en (4.1h).

À noter que dans cette formulation, nous avons choisi de représenter les producteurs et les consommateurs sous la même forme de "consomacteurs", et de les distinguer des nœuds auxquels ils sont connectés. Classiquement, on retrouve par exemple l'équation (4.1d) sous

la forme de $S_n^g - S_n^d = \sum_{m \in \mathcal{N}_n} S_{nm}$ où S_n^g et S_n^d représentent respectivement la puissance complexe du générateur au nœud n et celle du consommateur au nœud n. Cette formulation ne permet pas d'avoir plusieurs consommateurs ou producteurs sur le même nœud, et ne permet pas non plus d'avoir un consomacteur qui puisse produire pendant un moment de la journée, puis consommer pendant un autre.

Il est également à noter que dans la littérature, on considère traditionnellement les consommations comme étant non flexibles, d'où la formulation donnée dans le paragraphe précédent. Dans notre formulation, les consommations non flexibles sont représentées par des consomacteurs dont les bornes inférieures et supérieures sont égales $\underline{S_i} = \overline{S_i}$. Cela peut aussi être transposé dans le cas des producteurs non flexibles, tels que les producteurs renouvelables. Cependant, dans ce cas, on peut toujours imaginer un degré de flexibilité de délestage, où l'on aurait alors $\underline{S_i} = 0$ et $\overline{S_i} = S_{i,max}$.

Le problème de l'OPF contient des contraintes non convexes, notamment (4.1e), (4.1g) et (4.1h). Ainsi, il se peut qu'il existe plusieurs minima locaux à un problème donné.

Il est possible de relaxer le problème de l'OPF en utilisant la programmation semi-définie positive (SDP), ou bien de le linéariser autour d'un point de fonctionnement donné. En utilisant les équations DC du flux de puissance, on obtient ce qu'on appelle le problème DC-OPF qui est linéaire et très rapide à résoudre, mais qui, en contrepartie, est moins précis [107].

Dans la suite de ce chapitre, nous allons nous concentrer sur la version non linéaire du problème, en utilisant des solveurs non linéaires basés sur la méthode de points intérieurs comme le solveur `Ipopt` [108].

4.2 Résolution centralisée

4.2.1 Résolution centralisée synchrone

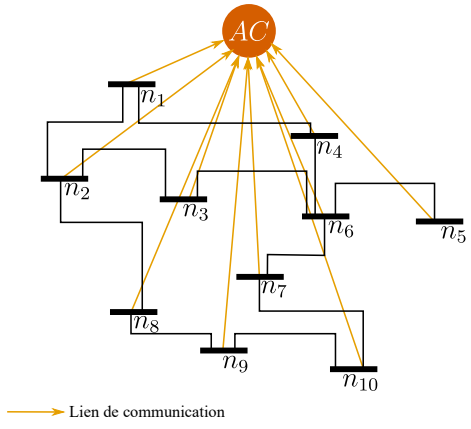


FIGURE 4.1 – Schéma exemple d'un réseau dont le problème d'OPF est résolu par un agent central *AC*. Les flèches jaunes représentent les liens de communication entre les nœuds du réseau et l'agent central.

La résolution centralisée de l'OPF se fait par un agent central, sur une unique machine de calcul possédant toutes les données nécessaires pour résoudre le problème, en particulier les informations portant sur :

— l'ensemble des nœuds $\mathcal{N}$ du réseau électrique, les lignes reliant ces nœuds et la matrice d'admittance $\mathbf{Y}$ associée ;
— l'ensemble des consomacteurs $\mathcal{C}_n$ associés à chaque nœud $n \in \mathcal{N}$, ainsi que leurs fonctions coût et leurs limites de puissance ;
— les valeurs limites des puissances dans les lignes, des différences d'angle de la tension de par et d'autre de chaque ligne et de la valeur absolue de la tension de chaque nœud.

L'agent central "AC" ainsi que ses liens de communication avec les nœuds d'un cas test sont représentés en Figure 4.1. Une résolution "synchrone" dans le schéma centralisé correspondrait au fait qu'avant chaque résolution de l'OPF, tous les consomacteurs communiquent avec l'agent central leurs données, c'est-à-dire leur fonction coût et leurs limites de production et consommation pour un instant de livraison futur. La durée de la résolution centralisée se compose alors dans un premier temps d'une durée de collecte des informations, dans un deuxième temps de la durée de calcul de l'OPF, et enfin, de la durée de renvoi de la solution à tous les agents. Une étude similaire a été effectuée en section 3.2 du Chapitre précédent pour la résolution centralisée de marché.

De la même manière que pour un marché de l'électricité, la résolution d'un OPF se fait pour un instant futur de livraison (instant T) donné et permet de déterminer les valeurs des puissances à injecter et à consommer des agents flexibles du réseau électrique. Il est alors nécessaire que le calcul de l'OPF et l'étape de communication de l'agent central vers les consomacteurs du marché se termine avant l'instant T pour lequel le calcul a été fait en premier lieu.

Or, les mêmes difficultés que pour le marché centralisé se présentent lorsque l'on considère la résolution centralisée de l'OPF. D'abord, le fait que l'agent central collecte l'intégralité des données du marché provoque un goulet d'étranglement de communication. Aussi, la durée des étapes de communication (voir Figure 3.2) augmente avec le pourcentage de pertes de messages sur le réseau de communication, et qui est d'autant plus élevée que le nombre d'agents considérés. Enfin, le temps de calcul du problème d'OPF augmente avec la taille du réseau considéré.

Ainsi, la présence d'aléas de communication pourrait fortement perturber le bon déroulement des calculs centralisés. En effet, si au moment de lancer les calculs pour un instant de livraison donné, l'agent central n'a pas reçu l'intégralité des données nécessaires, il faudrait alors procéder avec une portion de données obsolètes ce qui impliquerait que la solution trouvée soit sous optimale ou non faisable car ne pouvant pas respecter les contraintes physiques du réseau.

4.2.2 Résolution centralisée asynchrone

Dans cette partie, on se place dans le cas d'une résolution du problème d'Optimal Power Flow centralisé où l'agent central n'a pas reçu toutes les données mises à jour des consomacteurs. On applique les calculs sur un cas test à 118 agents complété par des données temporelles de consommation avec une période d'échantillonnage égal à la minute, issues du jeu de données UMass [1] [99]. On effectue le calcul d'OPF sur les 1440 points d'une journée choisie arbitrairement parmi le jeu de données. Le cas test ainsi que les données temporelles utilisées sont identiques à ceux utilisés dans la section 3.2 et sont explicités dans l'Annexe A.II.1. L'algorithme considéré est similaire à l'Algorithme 3.2 du Chapitre 3. On part du principe que les données non reçues sont remplacées par les données du pas de temps précédent. Ainsi, la solution trouvée sera différente de la solution optimale (synchrone),

1. `https://traces.cs.umass.edu/index.php/smart/smart`

et en application elle risque même de ne pas respecter les contraintes physiques du réseau compte tenu que les données d'entrée du problème sont différentes des données effectivement appliquées par l'ensemble des agents.

La Figure 4.2 présente le pourcentage d'occurrences (sur les 1440 points de la journée considérée) de l'écart relatif entre la puissance produite optimale obtenue par résolution centralisée synchrone et la puissance produite déduite du résultat de l'OPF centralisé asynchrone, en fonction du pourcentage de données effectivement à jour. Les quartiles et médianes sont précisés dans le Tableau 4.1

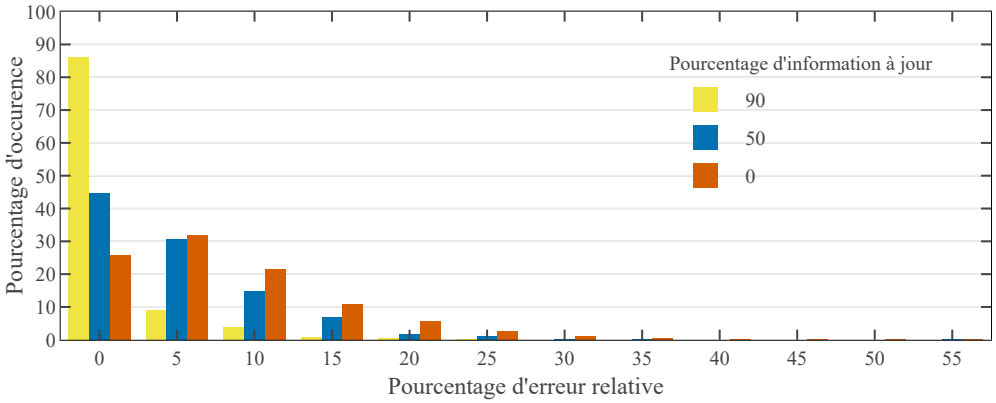


FIGURE 4.2 – Histogramme de l'erreur relative du résultat OPF centralisé à différents pourcentages d'informations à jour reçues par l'agent central.

Pourcentage de données mises à jour	Quartile 25%	Médiane	Quartile 75%
90	0.04%	0.24%	1.00%
50	1.03%	3.10%	7.43%
0	2.39%	6.17%	11.32%

TABLEAU 4.1 – Quartiles des erreurs relatives de la résolution centralisée asynchrone en fonction du pourcentage de données mises à jour.

On observe que l'écart entre la solution optimale et la solution réelle augmente statistiquement à mesure que le taux de données obsolète augmente.

— Pour 90% de données effectivement mises à jour, l'écart relatif ne dépasse pas 1% dans les trois quarts des cas.

— Pour 0% de données mises à jour, c'est-à-dire que toutes les données sont héritées du pas de temps précédent, alors l'écart relatif est beaucoup plus dispersé, pouvant dépasser les 11% dans un quart des cas.

Cet écart est limité par la corrélation temporelle des données : la puissance consommée par un agent à l'instant T a de grandes chances d'être égale ou très proche de la consommation à l'instant précédent lorsque la période d'échantillonnage est petite (de l'ordre de quelques secondes voir de la minute). L'écart augmente alors avec la période d'échantillonnage.

Si l'on veut passer outre les pertes de messages d'un pas de temps à un autre en remplaçant les données manquantes par celles précédemment reçues, alors on s'expose à des solutions sous optimales qui ne respectent pas forcément les contraintes physiques du réseau.

Enfin, et d'autant plus pour le problème d'OPF qui présente beaucoup plus de variables et de contraintes que le problème de marché, le temps de calcul augmente de manière significative lorsque l'on considère des réseaux à grande échelle, comme l'illustre la Figure 4.3 qui présente le temps de calcul du problème d'OPF en fonction du nombre de variables considérées [2].

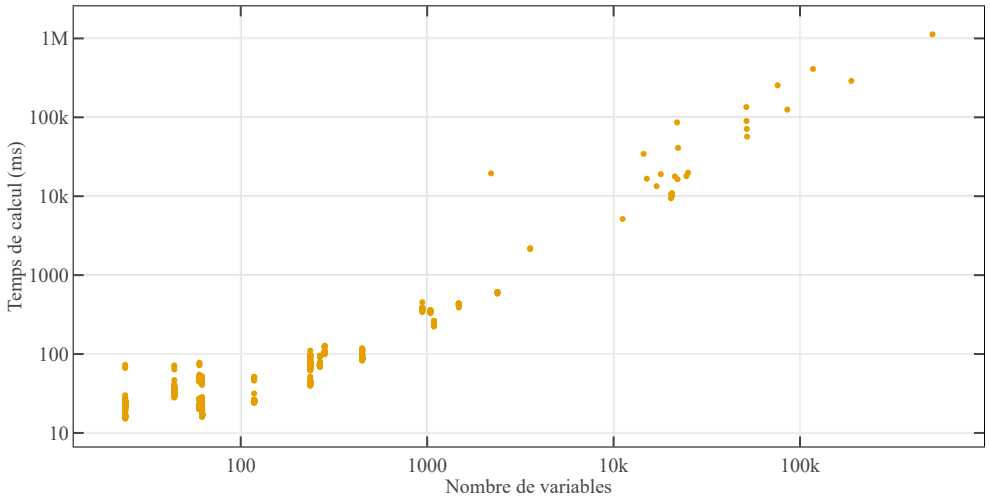


FIGURE 4.3 – Temps de calcul d'un OPF pour des cas test à différents nombres de variables. Les échelles en x et y sont logarithmiques. Le nombre de variables prend en compte les tensions complexes, les puissances complexes dans les lignes ainsi que les puissances complexes injectées par les consomacteurs.

Ces résultats sont très similaires à ceux présentés en section 3.2, ce qui montre que les problèmes de la résolution centralisée du marché de l'électricité et ceux de l'OPF centralisé sont semblables. Ce constat appelle donc à mobiliser les solutions de décentralisation asynchrone proposées au Chapitre 3, cette fois-ci dans le cadre de la résolution du problème d'Optimal Power Flow.

2. Voir la section 2.2 pour plus de précisions sur les cas tests utilisés et la machine sur laquelle les calculs ont été effectués.

4.3 Décomposition du problème pour la décentralisation de sa résolution

Afin de résoudre les problèmes soulevés dans la section précédente, on propose de se pencher sur des algorithmes d'OPF décentralisés, c'est-à-dire où le calcul de l'OPF n'est plus résolu par un seul agent central mais par une multitude d'agents qui ont chacun un rôle équivalent aux autres agents. Le réseau électrique est décomposé en plusieurs régions, chaque région inclue un certains nombre de nœuds, le plus souvent reliés par des lignes électriques. On considère qu'une machine locale à chaque région possède les informations nécessaires pour résoudre un OPF local : les valeurs des admittances des lignes de la région, ainsi que les fonctions coût des producteurs et consommateurs internes à la région. Chaque région résout un OPF local et communique avec les régions qui lui sont voisines, une région voisine étant définie comme ayant au moins un nœud connecté par une ligne physique à la région considérée.

On peut retrouver le schéma d'un partitionnement d'un réseau électrique en plusieurs régions en Figure 4.4a. La matrice de communication de chaque région est déterminée par la présence ou non de liens physiques entre deux régions. Le processus de résolution décentralisé est itératif, ce qui pousse les régions voisines à communiquer entre elles à chaque itération de l'algorithme.

À noter que la différence majeure avec le problème de marché pair à pair présenté dans le Chapitre précédent, outre le nombre de variables qui est plus élevé et les contraintes non convexes, réside dans la forme de la matrice de communication : les échanges se font de proche en proche, c'est-à-dire de régions voisines en régions voisines jusqu'à couvrir l'intégralité du réseau électrique. En revanche, dans le marché pair à pair, les échanges se font entre consomacteurs quels que soient leur position sur le réseau électrique physique. Il faut alors beaucoup plus d'itérations pour résoudre l'OPF décentralisé que pour résoudre le marché pair à pair du fait de la propagation de la solution sur le réseau entier.

La fonction à minimiser (4.1a) est une somme de fonctions coût locales à chaque consomacteur, donc locales à chaque nœud, ce qui la rend facilement décomposable. En revanche, les contraintes (4.1d) à (4.1g) ne sont pas facilement décomposables entre les différents problèmes locaux. Il est nécessaire de créer des copies locales des variables de tensions complexes de tous les nœuds limitrophes d'une région pour pouvoir totalement décomposer le problème. Plusieurs méthodes de décomposition sont possibles : consensus et innovation (C+I), ADMM, méthode des multiplicateurs de Lagrange, etc. Nous allons étudier au cours de ce chapitre deux algorithmes d'OPF décentralisés se basant sur la méthode ADMM, le premier étant issu de l'article de Erseghe [54], le second étant une décomposition redondante par frontières qui permet une asynchronisation plus facile de l'algorithme.

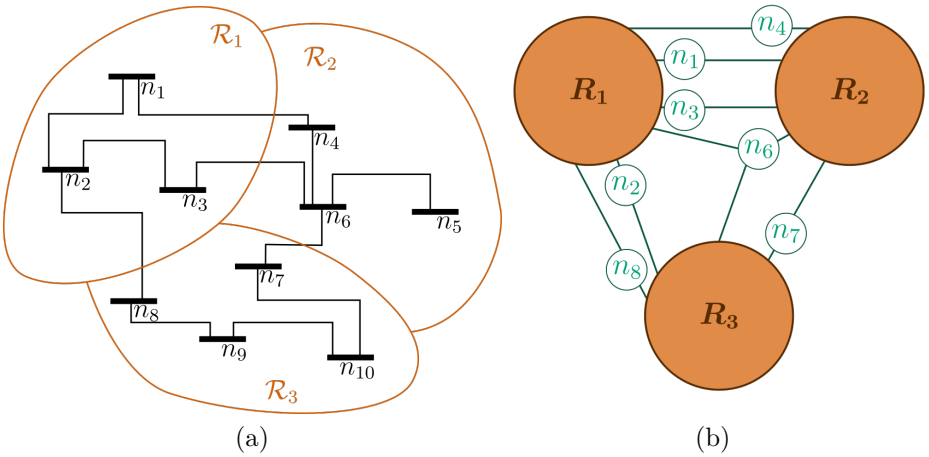
4.4 Résolution décentralisée - décomposition par nœuds

L'algorithme OPF issu de [54] qui présente une décomposition par nœuds, c'est-à-dire que pour chaque nœud limitrophe d'une région, une copie de sa tension complexe est créée et partagée entre les différentes régions voisines de ce nœud. Dans son article, Erseghe illustre la convergence de son algorithme sur trois cas test, dont le cas test IEEE à 118 nœuds que nous allons utiliser dans les sections suivantes.

4.4.1 Formulation synchrone

Chaque région k est représentée par un ensemble $\mathcal{R}_k$ de nœuds du réseau électrique. Un exemple de réseau électrique ainsi que son partitionnement en trois régions est présenté en Figure 4.4a. Pour un nœud n, on note $\mathcal{N}_n$ l'ensemble des nœuds connectés au nœud n par une ligne électrique. Parmi les nœuds d'une région, certains sont internes à la région, c'est-à-dire qu'ils ne sont connectés qu'à d'autres nœuds de la même région, n_9 et n_{10} sont internes à la région 3 par exemple. D'autres nœuds sont frontaliers, c'est-à-dire qu'ils possèdent au moins une connexion physique avec un nœud d'une autre région, par exemple n_7 et n_8 sont des nœuds frontaliers de la région 3. Les nœuds n_2 et n_6 sont aussi des nœuds frontaliers de la région 3 même s'ils n'en font pas partie. On désigne $\mathcal{O}_k$ comme l'ensemble des nœuds frontaliers d'une région k.

La décomposition par régions de l'algorithme est possible en créant des variables dupliquées des nœuds frontaliers. En effet, à cause des contraintes (4.1d) à (4.1g), les calculs locaux à la région 3 nécessitent la connaissance de la tension des nœuds $n_2 \in \mathcal{R}_1$ et $n_6 \in \mathcal{R}_2$, et réciproquement les régions 1 et 2 ont besoin de connaître les tensions des nœuds n_8, $n_7 \in \mathcal{R}_3$ respectivement. La Figure 4.4b présente les liens entre les nœuds frontaliers et les régions : si un nœud n possède un lien avec une région k, alors il existe une copie de la tension du nœud n locale à k : v_n^k.



(a) (b)

FIGURE 4.4 – Schémas d'un exemple de réseau électrique partitionné en trois régions. La Figure 4.4a représente les nœuds du réseau électrique ainsi que leurs connexions physiques via les lignes électriques. La Figure 4.4b représente les nœuds dont les tensions complexes sont dupliquées entre les différentes régions. Tout lien entre une région et un nœud signifie qu'il existe une variable interne à la région représentant l'état de ce nœud.

Formulation du problème d'Optimal Power Flow décomposé par nœuds

$$\arg\min_{\mathbf{x}} \sum_{k=1}^{R} \sum_{n\in\mathcal{R}_k} \sum_{i\in\mathcal{C}_n} f_i(P_i) \tag{4.2a}$$

$$\text{selon } \mathbf{x} : \begin{cases} V_n \in \mathbb{C} & n \in \mathcal{R}_k \cup \mathcal{O}_k,\ k \in [\![1,\ R]\!] \\ v_n^k \in \mathbb{C} & n \in \mathcal{O}_k,\ k \in [\![1,\ R]\!] \\ S_i = P_i + jQ_i \in \mathbb{C} & i \in \mathcal{C}_n,\ n \in \mathcal{O}_k,\ k \in [\![1,\ R]\!] \\ S_{nm} \in \mathbb{C} & m \in \mathcal{N}_n,\ n \in \mathcal{R}_k,\ k \in [\![1,\ R]\!] \end{cases}$$

$$\text{tel que } V_n = v_n^h \qquad n \in \mathcal{O}_k \cap \mathcal{O}_h,\ k, h \in [\![1,\ R]\!] \tag{4.2b}$$

$$\underline{V_n} \leq |V_n| \leq \overline{V_n} \qquad n \in \mathcal{R}_k \cup \mathcal{O}_k,\ k \in [\![1,\ R]\!], \tag{4.2c}$$

$$\underline{S_i} \leq S_i \leq \overline{S_i} \qquad i \in \mathcal{C}_n,\ n \in \mathcal{R}_k,\ k \in [\![1,\ R]\!], \tag{4.2d}$$

$$\sum_{i\in\mathcal{C}_n} S_i = \sum_{m\in\mathcal{N}_n} S_{nm} \qquad n \in \mathcal{R}_k,\ k \in [\![1,\ R]\!], \tag{4.2e}$$

$$S_{nm} = Y_{nm}^* V_n (V_n - V_m)^* \qquad m \in \mathcal{N}_n,\ n \in \mathcal{R}_k,\ k \in [\![1,\ R]\!], \tag{4.2f}$$

$$\underline{S_{nm}} \leq |S_{nm}| \leq \overline{S_{nm}} \qquad m \in \mathcal{N}_n,\ n \in \mathcal{R}_k,\ k \in [\![1,\ R]\!], \tag{4.2g}$$

$$\underline{\theta_{nm}} \leq \angle(V_n V_m^*) \leq \overline{\theta_{nm}} \qquad m \in \mathcal{N}_n,\ n \in \mathcal{R}_k,\ k \in [\![1,\ R]\!], \tag{4.2h}$$

$$\angle V_n = 0 \qquad n \in \mathcal{R}_k \cap \mathcal{N}_{ref},\ k \in [\![1,\ R]\!], \tag{4.2i}$$

La formulation du problème incluant les copies des tensions frontalières est donnée en (4.2). On retrouve les contraintes physiques dans les équations (4.2c)-(4.2i). La contrainte d'égalité des différentes copies des tensions composées est donnée en (4.2b). On réalise alors un consensus entre les tensions copiées grâce à la méthode ADMM, en posant m_n^h la variable duale de chaque contrainte de (4.2b), pour tout n nœud frontalier et pour toute région h voisine du nœud n. Le détail des calculs est donné en [54]. On obtient alors un algorithme itératif où chaque région procède selon l'Algorithme 4.1.

Les copies des tensions frontalières v_n^k locales à la région k sont mises à jour suivant (4.3). Cela revient à effectuer un calcul local d'Optimal Power Flow sur une partie du réseau électrique, notamment sur les nœuds de la région k et les nœuds voisins, avec une fonction coût modifiée (4.3a) présentant un terme provenant des fonctions coût locales de chaque agent et un autre terme provenant du consensus ADMM sur les tensions frontalières. Une fois ces copies mises à jour pour tous les nœuds frontaliers $n \in \mathcal{O}_k$ de la région locale k, elles sont communiquées à toutes les régions voisines qui possèdent également une copie des mêmes nœuds. Réciproquement, la région locale se met alors en attente de réception de toutes les mises à jour des copies des régions voisines. Une fois que toutes ces copies sont reçues, le calcul peut continuer : on met à jour des valeurs intermédiaires avec (4.4) (valeur de "mixage"), (4.5) (valeur de "mémoire") et (4.6).

Nota

Il peut arriver que deux régions k et h ne soient pas directement connectées via une ligne physique et malgré tout doivent communiquer le résultat d'un nœud n dont elles sont toutes deux voisines. Dans la Figure 4.4, si l'on omettait la

connexion entre n_2 et n_8, ce serait par exemple le cas des régions 1 et 3 qui devraient se communiquer les mises à jours relatant de la tension du nœud n_6.

Algorithme 4.1 OPF décomposé par nœuds synchrone, local à la région k, basé sur [54]

tant que la convergence n'est pas atteinte
si $t = 0$ **alors**
Initialiser les tensions locales $\mathbf{v}^k = \left[v_n^k\right]_{n\in\mathcal{O}_k}$
sinon
Mettre à jour les tensions locales via

$$\mathbf{x}^{temp} \leftarrow \arg\min_{\mathbf{x}} \sum_{n\in\mathcal{R}_k}\sum_{i\in\mathcal{C}_n} f_i(P_i) + \sum_{n\in\mathcal{O}_k} d_{n,k}|V_n - \beta_n^k|^2 \tag{4.3a}$$

$$\text{selon } \mathbf{x} : \begin{cases} V_n \in \mathbb{C} & n \in \mathcal{R}_k \cup \mathcal{O}_k \\ S_i = P_i + jQ_i \in \mathbb{C} & n \in \mathcal{O}_k,\ i \in \mathcal{C}_n \\ S_{nm} \in \mathbb{C} & n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \end{cases}$$

$$\text{tel que } \underline{V_n} \le |V_n| \le \overline{V_n} \qquad n \in \mathcal{R}_k \cup \mathcal{O}_k \tag{4.3b}$$

$$\underline{S_i} \le S_i \le \overline{S_i} \qquad n \in \mathcal{R}_k,\ i \in \mathcal{C}_n \tag{4.3c}$$

$$\sum_{i\in\mathcal{C}_n} S_i = \sum_{m\in\mathcal{N}_n} S_{nm} \qquad n \in \mathcal{R}_k \tag{4.3d}$$

$$S_{nm} = Y_{nm}^* V_n (V_n - V_m)^* \qquad n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \tag{4.3e}$$

$$\underline{S_{nm}} \le |S_{nm}| \le \overline{S_{nm}} \qquad n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \tag{4.3f}$$

$$\underline{\theta_{nm}} \le \angle(V_n V_m^*) \le \overline{\theta_{nm}} \qquad n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \tag{4.3g}$$

$$\angle V_n = 0 \qquad n \in \mathcal{R}_k \cap \mathcal{N}_{ref} \tag{4.3h}$$

$$v_n^k \leftarrow V_n^{temp} \qquad n \in \mathcal{O}_k \tag{4.3i}$$

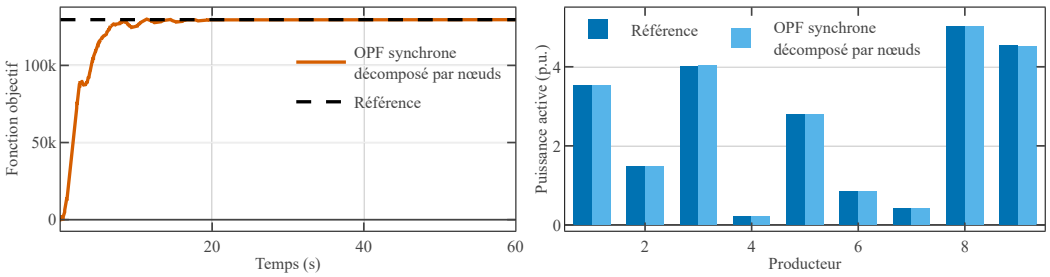
fin si
Envoyer les valeurs limitrophes v_n^k, $n \in \mathcal{O}_k$ aux régions voisines
tant que les v_n^h, $n \in \mathcal{O}_k \cap \mathcal{O}_h$ ne sont pas toutes reçues de chaque région voisine h
Attendre
fin tant que
Évaluer les valeurs intermédiaires $u_n^k \leftarrow \sum_{h\in\mathcal{M}_n} \frac{1}{2}\tilde{a}_{n,k,h}(v_n^k - v_n^h)$, $n \in \mathcal{O}_k$ (4.4)
si $t = 0$ **alors**
Initialiser les valeurs intermédiaires $m_n^k \leftarrow 0$, $n \in \mathcal{O}_k$
sinon
Mettre à jour les valeurs intermédiaires $m_n^k \leftarrow m_n^k + u_n^k$, $n \in \mathcal{O}_k$ (4.5)
fin si
$\beta_n^k \leftarrow v_n^k - u_n^k - m_n^k$, $n \in \mathcal{O}_k$ (4.6)
$t \leftarrow t + 1$
fin tant que

Les paramètres $d_{n,k}$ et $\tilde{a}_{n,k,h}$ apparaissant respectivement en (4.3a) et (4.4) sont des coefficients dépendant des nœuds et des régions. Ils sont définis en [54] à partir de poids w_k associés à chaque région et d'un paramètre ϵ.

$$a_{n,k,h} = \epsilon\, w_k\, w_h \frac{1-\delta_{k,h}}{M_n - 1} \qquad d_{n,k} = \sum_{h \in \mathcal{M}_n} a_{n,k,h} \qquad \tilde{a}_{n,k,h} = \frac{a_{n,k,h}}{d_{n,k}} \tag{4.7}$$

où $\delta_{k,h}$ est le delta de Kronecker valant zéro si $k = h$ et M_n étant le nombre de régions possédant une copie du nœud n. Le bon réglage des paramètres ϵ et (w_k) est essentiel à la convergence de l'algorithme. La vitesse de convergence est extrêmement sensible à la valeur de ces paramètres, il est donc essentiel de trouver le meilleur réglage en version synchrone avant de réaliser des essais en version asynchrone, et en supposant qu'ils restent valides en asynchrone. Erseghe préconise dans son article [54] de régler les paramètres empiriquement, en suivant approximativement la règle suivante : plus une région k produit de la puissance électrique, plus son coefficient w_k associé doit être petit. D'un point de vue opérationnel, il est possible d'appliquer l'algorithme à un cas donné dont la solution ne diffère pas fortement d'un pas de temps à un autre. Cependant, il apparaît difficile d'appliquer l'algorithme à une multitude de cas si le réglage de ces paramètres doit se faire "à la main".

Application sur un cas test L'algorithme présenté est testé sur un jeu de données IEEE à 118 nœuds, 53 générateurs et 186 lignes, partitionné en 10 régions différentes et décrit plus en détails dans l'Annexe A.II. On considère des délais de communication constants égaux à 50 ms, et des temps de calculs issus du modèle présenté en section 2.2.



(a) (b)

FIGURE 4.5 – Résultat de l'algorithme OPF décentralisé appliqué au cas test à 118 nœuds et 10 régions et comparaison par rapport à la solution centralisée synchrone de référence. En Figure 4.5a, la valeur de la fonction objectif globale est affichée au cours du temps de la simulation. En Figure 4.5b, les puissances actives de 9 producteurs au bout de 60 secondes de calculs sont comparées à la solution centralisée optimale.

Les résultats de la version synchrone de l'algorithme est donné en Figure 4.5. On y retrouve la valeur de la fonction objectif globale au cours du temps ainsi que les valeurs finales des puissances actives comparées à la solution optimale. On observe que l'algorithme converge vers la solution optimale au bout d'environ 20 secondes [3] (suivant l'horloge de la simulation), en comptant les délais de communication et les temps de calculs. Ce résultat sera à comparer avec la version asynchrone de l'algorithme.

3. Ce temps de convergence est à contextualiser compte tenu du petit nombre de variables considérées et du faible nombre de régions.

4.4.2 Formulation asynchrone

Comme expliqué lors du chapitre précédent, le fait d'attendre les messages de toutes les régions voisines à chaque itération ralentit fortement l'avancement de l'algorithme, notamment si les délais de communication sont variés. On cherche alors à effectuer le déclenchement des calculs à chaque itération après avoir reçu seulement une fraction δ des messages attendus. Les messages qui ne sont pas arrivés à temps pour être traités à l'itération en cours seront traités dans les itérations suivantes.

On a vu que chaque nœud frontalier n d'une région k pouvait être copié autant de fois que le nombre de régions proches de ce nœuds. Or, la variable duale m_n^k du consensus sur la tension v_n^k est unique peu importe le nombre de copies. Sa mise à jour nécessite une moyenne pondérée (4.4) de l'écart entre la copie locale v_n^k et les copies voisines v_n^h. L'asynchronisation de l'algorithme peut alors se faire de deux manières différentes.

— Soit l'on attend un nombre donné de messages, peu importe la provenance et les nœuds concernés. Dans ce cas, les mises à jour de la tension v_n^k et de la variable duale m_n^k associée sont effectuées si au moins un message concernant le nœud n est reçu parmi les messages pris en compte à une itération donnée.

— Soit l'on introduit une manière de compter les informations reçues qui prend en compte l'intégralité des copies d'un même nœud. Prenons un exemple où trois régions k, h, l possèdent des copies du même nœud n. Pour que la région k puisse effectuer les mises à jour de la variable locale v_n^k et de la variable duale associée m_n^k, alors elle doit avoir reçu les dernières informations v_n^h et v_n^l en provenance des régions h et l. On ne compte alors plus le nombre de messages reçus pour le déclenchement d'une itération, mais le nombre de nœuds "activés", c'est-à-dire le nombre de nœuds dont on a reçu l'intégralité des copies des régions voisines.

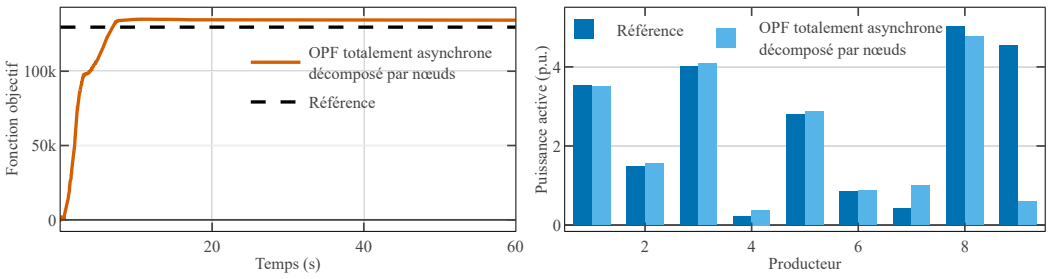


FIGURE 4.6 – Résultat de l'algorithme OPF décentralisé totalement asynchrone appliqué au cas test à 118 nœuds et et comparaison par rapport à la solution optimale. Les agents sont déclenchés après avoir reçu 10% des messages attendus. En Figure 4.6a, la valeur de la fonction objectif globale est affichée au cours du temps. En Figure 4.6b, les puissances actives de 9 producteurs au bout de 60 secondes de calculs sont comparées à la solution optimale.

En appliquant la première méthode asynchrone à notre problème d'OPF, on se rend compte en Figure 4.6 que, bien que l'algorithme converge, il ne converge pas vers la solution optimale. En effet, le fait que la valeur intermédiaire en (4.4) associée au nœud n dépende des valeurs v_n^h de toutes les régions voisines impose que toutes ces valeurs soient effectivement mises à jour durant la même itération.

La seconde méthode permet de prendre en compte toutes ces valeurs en même temps dans (4.4), ce qui revient à dire que considérant un nœud $n \in \mathcal{O}_k$, le fonctionnement reste synchrone. En revanche, considérant l'ensemble des nœuds dans $\mathcal{O}_k$, il n'est pas nécessaire d'avoir reçu toutes les informations de tous ces nœuds pour pouvoir continuer les calculs. C'est un fonctionnement **asynchrone mais synchrone par nœud**. Il est alors nécessaire de modifier le déclencheur de l'implémentation asynchrone.

Algorithme 4.2 OPF décomposé par nœuds asynchrone, local à la région k

tant que la convergence n'est pas atteinte
si $t = 0$ **alors**
Initialiser les tensions locales $\mathbf{v}^k = \left[v_n^k\right]_{n \in \mathcal{O}_k}$
$\mathcal{A}_k^0 = \mathcal{O}_k$
sinon
Mettre à jour les tensions locales via

$$\mathbf{x}^{temp} \leftarrow \arg\min_{\mathbf{x}} \sum_{n \in \mathcal{R}_k} \sum_{i \in \mathcal{C}_n} f_i(P_i) + \sum_{n \in \mathcal{O}_k} d_{n,k} |V_n - \beta_n^k|^2 \tag{4.8a}$$

$$\text{selon } \mathbf{x} : \begin{cases} V_n \in \mathbb{C} & n \in \mathcal{R}_k \cup \mathcal{O}_k \\ S_i = P_i + jQ_i \in \mathbb{C} & n \in \mathcal{O}_k,\ i \in \mathcal{C}_n \\ S_{nm} \in \mathbb{C} & n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \end{cases}$$

tel que (4.3b) - (4.3h)

$$v_n^k \leftarrow V_n^{temp} \qquad n \in \mathcal{A}_k^t \tag{4.8b}$$

fin si
Envoyer les valeurs limitrophes v_n^k, $n \in \mathcal{A}_k^t$ aux régions voisines
tant que les v_n^h, $n \in \mathcal{A}_k^{t+1}$ ne sont pas toutes reçues des régions voisines h
Attendre
fin tant que
Évaluer les valeurs intermédiaires $u_n^k \leftarrow \sum_{h \in \mathcal{M}_n} \frac{1}{2} \tilde{a}_{n,k,h} (v_n^k - v_n^h),\ n \in \mathcal{A}_k^{t+1}$ (4.9)
si $t = 0$ **alors**
Initialiser les valeurs intermédiaires $m_n^k \leftarrow 0,\ n \in \mathcal{O}_k$
sinon
Mettre à jour les valeurs intermédiaires $m_n^k \leftarrow m_n^k + u_n^k,\ n \in \mathcal{A}_k^{t+1}$ (4.10)
fin si
$\beta_n^k \leftarrow v_n^k - u_n^k - m_n^k,\ n \in \mathcal{A}_k^{t+1}$ (4.11)
$t \leftarrow t + 1$
fin tant que

Notons $\mathcal{A}_k^t \subset \mathcal{O}_k$ l'ensemble des nœuds n dont il existe une copie locale à la région k et dont l'ensemble des messages voisins v_n^h tel que $n \in \mathcal{O}_h$ ont été reçus à l'itération t par la région k. Tout au plus, on peut avoir $\mathcal{A}_k^t = \mathcal{O}_k$ dans le cas où tous les messages ont été réceptionnés par la région k. On redéfinit alors le paramètre d'asynchronisme δ dans le cas de cet algorithme asynchrone :

$$|\mathcal{A}_k^t| \geq \lceil \delta \cdot |\mathcal{O}_k| \rceil \tag{4.12}$$

avec $\mathcal{O}_k$ l'ensemble des nœuds frontaliers de la région k et $|\mathcal{A}_k^t|$ le nombre minimal de nœuds qui sont prêts à être mis à jour à l'itération t.

L'algorithme asynchrone local à la région k est présenté dans l'Algorithme 4.2. Il est similaire à l'algorithme synchrone, avec comme différence l'ensemble des nœuds dont les valeurs sont mises à jour à chaque itération. Dans la version synchrone, ce sont tous les nœuds $n \in \mathcal{O}_k$ dont les copies sont mises systématiquement à jour tandis que dans la version asynchrone, ce sont seulement les nœuds $n \in \mathcal{A}_k^t$ qui sont mis à jour à l'itération t. On peut en effet le remarquer dans (4.8b) et (4.9)-(4.11).

Nota

L'ensemble $\mathcal{A}_k^t$ est déterminé par l'ordre d'arrivée des messages et par le paramètre d'asynchronisme δ, selon (4.12). Il est constitué de tous les nœuds dont tous les messages ont été reçus avant le déclenchement de l'itération. On dit que ces nœuds ont été *activés*. Il se peut que la région k reçoive des messages pendant ses calculs qui seront pris en compte durant une itération future, ou bien que plusieurs messages soient reçus de manière simultanée, c'est pourquoi $\lceil \delta \cdot |\mathcal{O}_k| \rceil$ n'est qu'une borne inférieure du nombre de nœuds activés. Le nombre effectif de nœuds activés dépend du nombre de régions avec qui la région locale partage des nœuds et du nombre de connexions avec ces régions.

La Figure 4.7 présente la résolution de l'algorithme asynchrone telle que décrit dans l'Algorithme 4.2. On observe que la solution finale est bien la solution optimale recherchée. À partir de maintenant, on se réfère seulement à l'algorithme asynchrone "synchrone par nœud" lorsque l'on parle d'algorithme asynchrone.

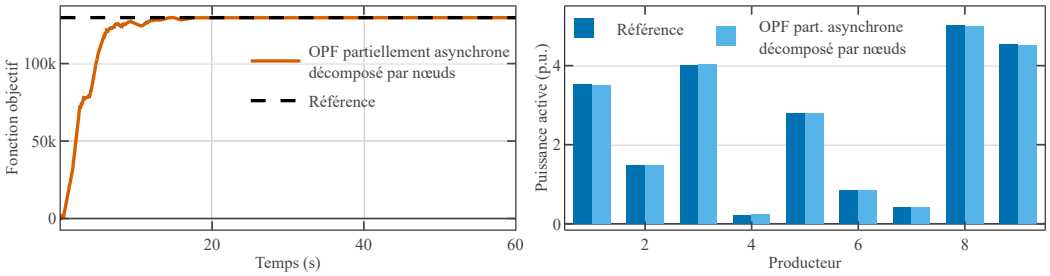


FIGURE 4.7 – Résultat de l'algorithme OPF décentralisé partiellement asynchrone ou "synchrone par nœud" appliqué au cas test à 118 nœuds et comparaison par rapport à la solution optimale centralisée synchrone. En Figure 4.7a, la valeur de la fonction objectif globale est affichée au cours du temps. En Figure 4.7b, les puissances actives de 9 producteurs au bout de 60 secondes de calculs sont comparées à la solution optimale.

Critère de convergence La définition de résidus primal ou dual en asynchrone dans le cas de cette décomposition en particulier est complexe dans la mesure où les différentes régions avancent à des rythmes variés et ne sont donc pas à la même itération en fonction du temps. On propose alors en (4.13) un critère de convergence différent du chapitre précédent, se reposant sur la minimisation de l'erreur entre la solution optimale connue $s^\star$ et la solution de l'algorithme décomposé à un instant donné s. On considère que l'algorithme a convergé dès lors que l'erreur passe en dessous d'un seuil ε.

$$\sum_{i \in \mathcal{C}} |S_i^\star - S_i|^2 \leq \varepsilon \tag{4.13}$$

Le seuil choisi pour les résultats qui suivent est de $\varepsilon = 10^{-3}$. En pratique, lorsque le nombre d'agents est de l'ordre de la dizaine, cela représente un écart de puissance de l'ordre de 0.01 p.u., soit 1% d'écart sur l'unité de puissance considérée pour chaque agent. Plus l'on considère d'agents, plus cet écart est petit à ε donné.

Paramètres des études réalisées Les sections suivantes décrivent les différentes études réalisées sur l'algorithme asynchrone décrit précédemment. Ces études sont appliquées sur un cas test à 118 nœuds, 53 générateurs et 186 lignes décrit en Annexe A.II. Les problèmes d'optimisation locaux des différentes régions sont résolus via le package Julia `PowerModels` [106] qui se base sur le solveur `Ipopt` [108] utilisant une méthode de points intérieurs. Les différents paramètres du modèle de délais de communication ainsi que du modèle de temps de calculs sont amenés à changer au cours des études. Leurs valeurs sont compilées dans le Tableau 4.2.

Section	**Cas**	Δ_{comm}	Δ_{calc}
4.4.3	118 nœuds 10 régions	$\alpha = 0$ $\beta = 50$ ms Déterministe $p_{pertes} = 0\%$	Modèle affine 2.2
4.4.4	118 nœuds 10 régions	$\alpha = 0$ $\beta = 50$ ms Déterministe avec pertes stochastiques	Modèle affine 2.2
4.4.5	118 nœuds 10 régions	$\alpha = 0$ Déterministe avec pertes stochastiques $p_{pertes} = 5\%$	Modèle affine 2.2
4.4.6	118 nœuds	$\alpha = 0$ $\beta = 50$ ms Déterministe avec pertes stochastiques $p_{pertes} = 5\%$	Modèle affine 2.2

Tableau 4.2 – Paramètres utilisés pour les simulations de l'algorithme OPF décomposé par nœuds de la section 4.4.

À noter que les délais de communication choisis sont tous égaux quelque soit les régions qui communiquent entre elles, étant donné que le paramètre α est toujours nul. C'est un choix délibéré qui nous permettra de facilement comparer les délais de communication Δ_{comm} au temps de calcul moyen par la suite. Étant donné que l'on considère un petit nombre de grandes régions, on peut supposer que les machines de calcul associées à chaque région sont situées dans des sous réseaux éloignés les uns des autres, mais néanmoins équidistants en termes de temps de communication.

On commence par étudier l'influence du paramètre d'asynchronisme en section 4.4.3. L'influence des aléas de communication sont étudiés en sections 4.4.4 et 4.4.5, notamment l'influence des pertes de messages et l'influence des délais de communication par rapport aux temps de calculs. On finit par une étude de l'influence de la taille des régions en section 4.4.6.

4.4.3 Influence du paramètre d'asynchronisme

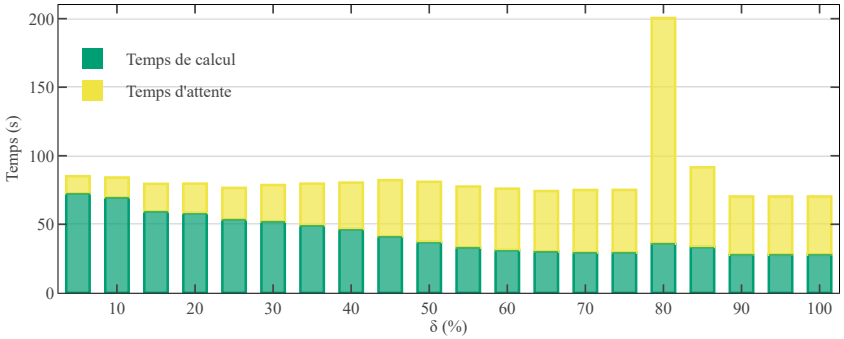


FIGURE 4.8 – Temps de convergence, constitué du temps de calcul moyen et du temps d'attente moyen, en fonction du paramètre d'asynchronisme δ, de l'algorithme OPF asynchrone, appliqué au cas test à 118 nœuds.

L'influence du paramètre d'asynchronisme δ, définit en (4.12), sur le temps de convergence est présenté en Figure 4.8 pour une première étude où l'on ne considère pas de pertes de messages. On y distingue le temps de calcul moyenné sur les différentes régions, ainsi que le temps d'attente entre chaque itération. On rappelle que l'algorithme à $\delta = 100\%$ est équivalent à l'algorithme synchrone, puisque tous les nœuds d'une région doivent être activés avant de pouvoir passer à l'itération qui suit. Ainsi, on observe que l'algorithme est plus rapide à converger dans le cas synchrone. Le temps de calcul a tendance à augmenter et le temps d'attente à diminuer à mesure que δ diminue. Seulement, le gain réalisé sur le temps d'attente n'est pas suffisant pour compenser l'augmentation du temps de calcul à mesure que l'algorithme devient de plus en plus asynchrone.

On observe autour de $\delta = 80\%$ un comportement qui contredit cette tendance. Cela est dû à un phénomène d'auto-blocage qui survient lorsque deux régions attendent chacune un message de l'autre pour pouvoir reprendre leurs calculs respectifs. Comme elles doivent effectuer leurs calculs locaux avant d'envoyer d'autres messages, elles restent bloquées et donc finissent par geler l'avancement global de l'algorithme. Nous avons ajouté un mécanisme permettant aux agents de calcul de se débloquer de cette situation après un certain délai d'attente (ici de 5 secondes). On observe alors une très forte augmentation du temps d'attente étant donné que ce phénomène d'auto-blocage se produit assez fréquemment pour cette valeur de δ.

L'augmentation du temps de calcul, c'est-à-dire du nombre moyen d'itérations, à mesure que l'asynchronisme augmente s'explique par la nécessité de combler le manque d'informations à chaque itération par l'augmentation du nombre d'itérations pour atteindre le même niveau de convergence. On voit que dans les conditions de la simulation, avec ce cas test à 10 régions et dans le cas où tous les délais de communication sont égaux, les versions asynchrones sont plus lentes à converger que la version synchrone. Ce résultat est différent de celui du marché pair à pair asynchrone car nous prenons ici en compte les temps de calcul là où ils étaient négligés dans l'étude précédente [4], et que l'augmentation du nombre d'itérations a pour effet d'augmenter le temps de calcul global moyen en asynchrone.

4. Les temps de calcul sont pris en compte pour l'OPF car ils sont du même ordre de grandeur que les délais de communication considérés.

4.4.4 Influence du taux de perte de messages

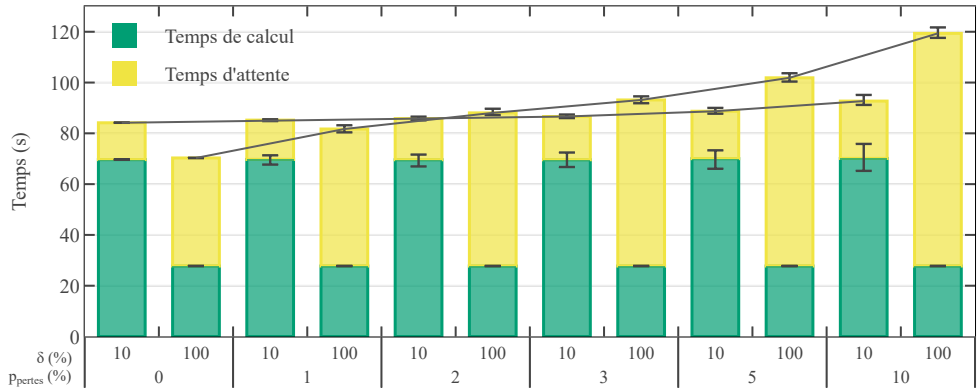


FIGURE 4.9 – Temps de convergence, constitué du temps de calcul moyen et du temps d'attente moyen, en fonction du taux de pertes de messages p_{pertes} et pour deux valeurs du paramètre d'asynchronisme $\delta = 10\%$ ou $\delta = 100\%$, appliqué au cas test à 118 nœuds. Les barres d'erreur représentent l'écart-type sur les différents délais.

On observe à présent l'influence du taux de perte de messages sur le temps de convergence de l'algorithme. On rappelle que le taux de pertes de messages p_{pertes} est défini en section 2.1 du Chapitre 2. Lorsqu'un message est perdu dans le réseau de communication, l'agent ayant envoyé le message l'envoie de nouveau après une temporisation égale à un temps d'aller-retour entre l'envoyeur et le destinataire. Ainsi, le délai de communication total d'un message ayant été perdu une fois est de 3 fois le délai de communication sans pertes.

On compare en Figure 4.9 le temps de convergence du cas synchrone à $\delta = 100\%$ au cas asynchrone à $\delta = 10\%$ pour différentes valeurs du taux de pertes p_{pertes}. Lorsque p_{pertes} est proche de zéro, peu de messages sont perdus donc sur l'ensemble des échanges, la valeur médiane des délais de communication reste identique à celle où il n'y a pas de pertes. Ainsi, les délais de communication deviennent plus variés mais conservent la même valeur médiane.

Le temps de convergence de l'algorithme augmente à mesure que le taux de perte augmente, ce qui est attendu. En revanche, cette augmentation est plus prononcée dans le cas synchrone que dans le cas asynchrone, ce qui rend l'algorithme asynchrone plus rapide à partir de $p_{pertes} = 2\%$. En effet, l'augmentation du temps de convergence est en moyenne de 11% en asynchrone entre $p_{pertes} = 0\%$ et $p_{pertes} = 10\%$ contre 70% pour le cas synchrone. Les messages ayant été perdus au moins une fois doivent tout de même être attendus par les agents en version synchrone, ce qui augmente systématiquement le temps d'attente global de l'algorithme. En revanche, ces messages ne sont pas attendus en asynchrone puisque les agents déclenchent leurs itérations avant leur réception, ce qui limite l'augmentation du temps d'attente global.

Les barres d'erreur de la Figure 4.9 représentent l'écart-type du temps de calcul ainsi que du temps d'attente. Dans le cas synchrone, quelque soit le taux de perte de messages, le temps de calcul global moyen est constant. Ceci s'explique par le besoin d'un nombre fixe d'itérations pour atteindre le même niveau de convergence.

> La version asynchrone de l'algorithme est plus robuste aux variations de délais induits par les pertes de message que la version synchrone.

4.4.5 Influence des délais de communication

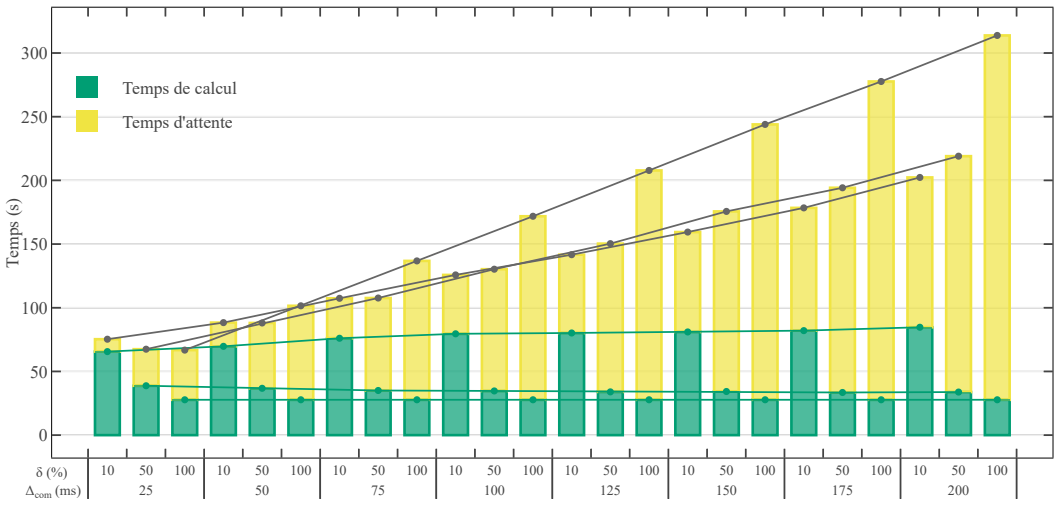


FIGURE 4.10 – Temps de convergence, constitué du temps de calcul moyen et du temps d'attente moyen, en fonction du délai de communication pour trois valeurs du paramètre d'asynchronisme $\delta = 10\%$, 50% ou $\delta = 100\%$, appliqué au cas test à 118 nœuds.

On observe à présent l'influence des délais de communication Δ_{comm} [5] sur le temps de convergence de l'algorithme OPF, à taux de perte de messages fixe $p_{pertes} = 5\%$ choisi arbitrairement. Dans cette étude, le délai de communication, identique pour toutes les communications, Δ_{comm} varie de 25 à 200 ms.

La Figure 4.10 présente les temps de calcul moyens, les délais d'attente moyens ainsi que les temps de convergence moyens en fonction du délai de communication Δ_{comm} et pour trois valeurs du paramètre d'asynchronisme $\delta \in \{10, 50, 100\}\%$. On remarque dans un premier temps que le temps de convergence augmente à mesure que les délais de communication augmentent, ce qui est naturel. De la même manière que dans la section précédente, l'augmentation du temps de convergence est moindre en asynchrone par rapport à la version synchrone de l'algorithme. En effet, en passant de $\Delta_{comm} = 25$ms à $\Delta_{comm} = 200$ ms, le temps de convergence augmente de 370% en synchrone contre 168% en asynchrone à $\delta = 10\%$. On peut en conclure que la version asynchrone est moins sensible à l'augmentation des délais de communication que la version synchrone.

Les temps de calculs moyens dépendent du nombre d'itérations locales, il est donc normal de constater qu'ils sont constants quelque soient les délais de communication en version synchrone $\delta = 100\%$ à niveau de convergence égal. En revanche, en asynchrone, on constate que les temps de calculs varient légèrement pour des délais de communication faibles. En effet, pour des délais de communication faibles à $\delta = 10\%$, les temps de calculs sont légèrement plus petits que pour des délais de communication plus élevés, ce qui traduit un plus petit nombre d'itérations.

5. On rappelle que les délais de communication sont égaux dans cette partie.

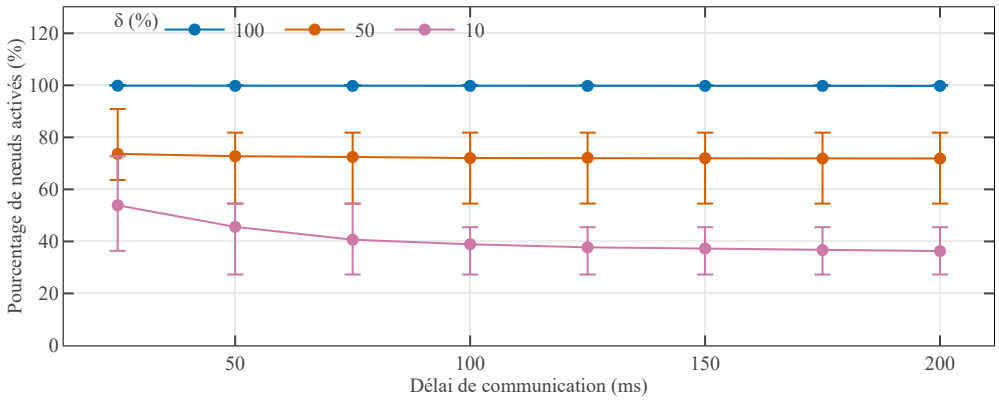


FIGURE 4.11 – Pourcentage de nœuds activés de la région $\boldsymbol{R}_4$ en fonction du délai de communication Δ_{comm} pour trois valeurs du paramètre d'asynchronisme $\delta = 10\%$, 50% ou $\delta = 100\%$, appliqué au cas test à 118 nœuds.

La Figure 4.11 réunit les informations nécessaires à l'explication de ce phénomène. Elle présente le pourcentage effectif de nœuds activés par rapport au délai de communication Δ_{comm}. Comme expliqué dans la note de la section 4.4.2, le véritable nombre de nœuds activés est différent du nombre donné à droite de l'inégalité (4.12), qui représente seulement le critère de déclenchement des calculs. La réduction des délais de communication à temps de calculs fixes permet à plus de messages d'atteindre les régions de destination avant que ces dernières n'aient terminé leurs calculs précédents. Le nœud est tout de suite activé à la fin du calcul local car tous les messages attendus sont arrivés pendant que l'agent était en train de calculer. Ainsi, le pourcentage de nœuds activés effectifs est d'autant plus grand que les délais de communication sont courts, comme illustré en Figure 4.11 pour les cas asynchrones. À $\delta = 10\%$, on observe clairement que le pourcentage de nœuds activés décroît avec Δ_{comm} jusqu'à se stabiliser après $\Delta_{comm} \geq 150$ ms. Cette variation reflète la variation de temps de calculs pour $\delta = 10\%$ de la Figure 4.10 : plus il y a de nœuds activés à chaque itération, moins il y a besoin d'itérations pour arriver à un même niveau de convergence.

4.4.6 Influence de la taille des régions

La résolution de l'OPF utilisant l'algorithme décentralisé décomposé par nœuds nécessite de régler très finement les paramètres liés à chaque région w_k, sous peine d'être confronté à une résolution très lente du problème ou même à ne pas être capable de converger vers la solution optimale. Cela rend difficile l'application de l'algorithme sur un grand nombre de partitionnements. Nous avons tout de même pu le tester sur 4 partitionnements différents (en 10, 8, 5 et 3 régions), basés sur le cas à 118 nœuds, décrits dans le Tableau II.2 en annexe. Ces partitionnements ont été construits en fusionnant les régions du cas à 10 régions jusqu'ici utilisé dans les simulations précédentes, ce qui facilite la recherche des paramètres w_k de l'algorithme, qui sont partagés dans le Tableau II.2 en Annexe A.II. Il n'y a pas assez de points dans cette étude pour en tirer des résultats statistiquement exacts, mais cela nous permet tout de même d'avoir une idée préliminaire de l'influence de l'asynchronisme sur différents partitionnements d'un même cas test.

Nombre de régions	3	5	8	10
Temps de convergence synchrone (s)	35	47	138	103
Accélération asynchrone maximale (%)	8.27	12.3	12.4	18.8
Paramètre d'asynchronisme δ correspondant à l'accélération (%)	10	30	20	60

TABLEAU 4.3 – Temps de convergence moyen pour l'algorithme décentralisé synchrone, avec un taux de perte de messages de $p_{pertes} = 5\%$, et l'accélération maximale observée en appliquant une implémentation asynchrone, l'accélération étant définie comme la différence entre le temps de convergence en asynchrone et en synchrone, divisé par le temps de convergence synchrone.

Les résultats en terme de temps de convergence synchrone et asynchrone sont compilés dans le Tableau 4.3. On observe que le temps de convergence pour l'algorithme synchrone augmente à mesure que le nombre de régions augmente. Cela est dû au fait qu'il y a plus de variables copiées entre les régions pour lesquels il faudrait atteindre un certain consensus.

En ce qui concerne le possible gain de temps qu'apporte l'implémentation asynchrone de l'algorithme (voir la ligne "Accélération asynchrone maximale" du Tableau 4.3), il est a priori plus élevé pour un nombre de région plus élevé. En effet, étant donné qu'il y a plus de messages à échanger entre les régions, l'asynchronisme a une plus grande capacité de diminution des temps d'attente. L'accélération maximale observée ne se fait pas au même degré d'asynchronisme selon le partitionnement. Il faudrait appliquer l'algorithme à un nombre plus élevé de partitionnements pour pouvoir en tirer des remarques concluantes.

Nota

Dans le cas test étudié, le nombre de nœuds total est assez petit pour que le calcul d'OPF s'effectue rapidement et sans nécessiter une grande puissance de calcul. Il faudrait tester l'algorithme sur un cas possédant un très grand nombre de nœuds pour éventuellement observer l'effet du temps de calcul sur une décomposition à faible nombre de régions.

4.4.7 Conclusions sur l'algorithme OPF asynchrone décomposé par nœuds

L'algorithme d'OPF décentralisé permet à des agents de calculs distribués par régions de résoudre le problème de l'Optimal Power Flow de manière itérative en communiquant les uns avec les autres. Chaque agent résout un sous-problème local comprenant des contraintes physiques de sa propre partie du réseau électrique. La particularité de l'algorithme décomposé par nœuds [54] sur lequel est basé notre étude est la nécessité de régler des paramètres propres à chaque région. La résolution du problème est très sensible au choix de ces paramètres, ce qui rend difficile toute comparaison de différents cas test.

Nous avons vu que la décomposition par nœuds de cet algorithme rend son asynchronisation plus compliquée que dans le chapitre précédent. En effet, il est nécessaire de rajouter une condition sur les messages reçus pour que la solution asynchrone soit bien équivalente à la solution optimale centralisée synchrone. Il est néanmoins possible d'effectuer des études similaires à celles du chapitre précédent, en étudiant le temps de convergence de l'algorithme en fonction du degré d'asynchronisme de ce dernier. Lorsque les délais de communication sont homogènes, on observe une augmentation du temps de convergence en asynchrone, dû au besoin d'effectuer plus de calculs pour pallier le manque d'information à chaque itération. Le temps d'attente diminue à mesure que l'asynchronisme augmente, mais pas assez pour pallier l'augmentation du temps de calcul moyen.

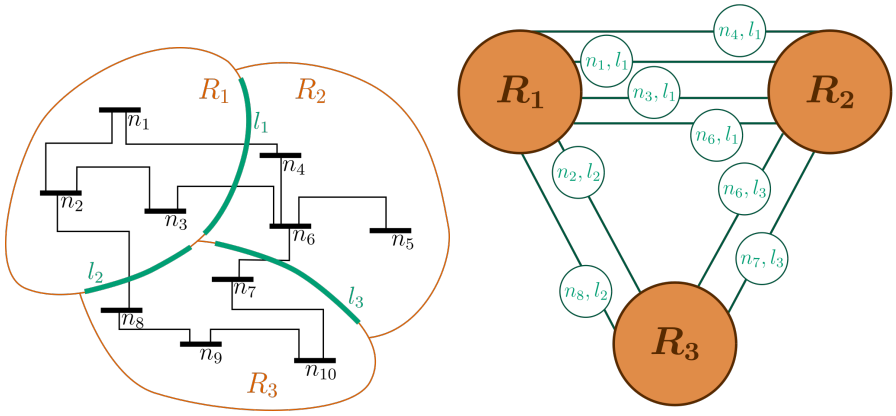
En revanche, on a pu montrer que le temps de convergence de l'algorithme asynchrone était moins sensible à la variation des délais de communication ainsi qu'à l'augmentation du taux de perte de messages que son homologue synchrone. Le gain de temps de l'algorithme asynchrone sur le synchrone est d'autant plus grand que le nombre de régions est grand, à nombre de nœuds égal. L'algorithme asynchrone a potentiellement le pouvoir de réduire fortement le temps de convergence dans le cas où l'on assigne une région à chaque nœud, c'est-à-dire lorsque l'on distribue totalement le problème d'OPF.

4.5 Résolution décentralisée - décomposition par frontières

Comme on l'a vu dans la partie 4.4, l'algorithme OPF décomposé par nœuds ne permet pas d'appliquer directement le principe de résolution asynchrone car ce dernier converge vers une solution sous-optimale. Il a fallu en effet rajouter une condition sur les messages reçus pour obtenir des résultats optimaux, ce qui limitait la possibilité d'accélération de la convergence. Pour pallier ce problème, nous proposons une nouvelle décomposition par liens, toujours basée sur la méthode ADMM, qui s'inspire de la décomposition pair-à-pair du Chapitre 3 appliquée aux tensions limitrophes de chaque région. L'intérêt de ce nouvel algorithme réside premièrement dans sa simplicité d'application, étant donné qu'il n'y a plus de paramètre associé à chaque région et qu'il n'y a plus besoin de rajouter de condition sur les messages reçus en version asynchrone. Comme pour l'algorithme de marché pair à pair asynchrone, on peut donner la même piste de preuve de convergence asynchrone que celle donnée en section 3.4, qui elle même s'inspire de la preuve donnée en [75].

4.5.1 Version synchrone

L'algorithme décrit précédemment ne pouvait pas être complètement asynchrone sans que la solution finale ne dévie de la solution optimale. En effet, la variable duale associée au consensus de la tension d'un nœud devait être identique pour toutes les régions possédant une copie de ce nœud. Il apparaît alors que cette variable duale devait être mise à jour de la même manière dans toutes les régions citées, c'est pour cela qu'il était nécessaire d'attendre tous les messages concernant un même nœud, même dans la version asynchrone. L'algorithme décrit dans cette partie duplique les tensions des nœuds limitrophes autant de fois que de régions avec lesquelles ce nœud est connecté. De cette manière, il y a autant de variables duales pour un nœud que de connexion avec des régions voisines, et on peut se permettre de ne pas attendre toutes les mises à jour liées au même nœud dans la version asynchrone.



(a) (b)

FIGURE 4.12 – Schémas d'un exemple de réseau électrique partitionné en trois régions. La Figure 4.12a représente les nœuds du réseau électrique ainsi que leurs connexions physiques via les lignes électriques. Les frontières l entre des différentes régions sont représentées en vert. La Figure 4.12b représente les indices des variables dupliquées entre les différentes régions.

La Figure 4.12 représente un exemple de réseau électrique partitionné en trois régions. Les différents nœuds ainsi que leur répartition dans les régions sont affichés en Figure 4.12a. On y a aussi affiché les liens (ou frontières) entres régions l : chaque lien définit la présence d'au moins une connexion physique entre deux régions distinctes. On retrouve aussi la notation des liens l comme étant la paire de régions qu'ils connectent $l = (\boldsymbol{R}_k, \boldsymbol{R}_k^{'})$. On note $\mathcal{L}_n$ l'ensemble des liens entre régions du réseau considéré. On rappelle aussi la notation $\mathcal{R}_k$ désignant l'ensemble des nœuds internes à la région $\boldsymbol{R}_k$. Dans l'exemple de la Figure 4.12a, n_1, n_2, n_3 sont les nœuds internes à la région $\boldsymbol{R}_1$. De même, la notation $\mathcal{O}_k$ désigne l'ensemble des nœuds frontaliers de la région $\boldsymbol{R}_k$. Dans l'exemple, n_1, n_2, n_3, n_4, n_6, n_8 sont les nœuds frontaliers de la région $\boldsymbol{R}_1$.

La Figure 4.12b représente les indices des différentes variables dupliquées et leurs relations avec les régions. Si une région $\boldsymbol{R}_k$ est connectée à un indice (n, l), alors il existe une variable représentant la tension du nœud n locale à la région $\boldsymbol{R}_k$ et exclusive au lien l : $v_{n,l}^{\boldsymbol{R}_k}$. Cela permet de décorréler les consensus sur les nœuds possédant des variables dupliquées dans trois régions ou plus, ce qui est illustré par la comparaison entre la Figure 4.4b et la Figure 4.12b concernant la duplication du nœud n_6.

Formulation du problème d'Optimal Power Flow décomposé par frontières

$$\min \sum_{k=1}^{R} \sum_{n \in \mathcal{R}_k} \sum_{i \in \mathcal{C}_n} f_i(P_i) \tag{4.14a}$$

selon

$$\begin{aligned}
V_n^{\boldsymbol{R}_k} &\in \mathbb{C}, && n \in \mathcal{R}_k \cup \mathcal{O}_k,\ k \in [\![1, R]\!] \\
S_i = P_i + jQ_i &\in \mathbb{C}, && i \in \mathcal{C}_n,\ n \in \mathcal{R}_k,\ k \in [\![1, R]\!] \\
S_{nm} &\in \mathbb{C}, && m \in \mathcal{N}_n,\ n \in \mathcal{R}_k,\ k \in [\![1, R]\!] \\
v_{n,l}^{\boldsymbol{R}_k} &\in \mathbb{C} && l \in \mathcal{L}_n,\ n \in \mathcal{O}_k,\ k \in [\![1, R]\!]
\end{aligned}$$

tel que

$$\frac{v_{n,l}^{\boldsymbol{R}_k} + v_{n,l}^{\boldsymbol{R}_k^{'}}}{2} = V_n^{\boldsymbol{R}_k} \qquad l = (\boldsymbol{R}_k, \boldsymbol{R}_k^{'}) \in \mathcal{L}_n,\ n \in \mathcal{O}_k,\ k \in [\![1, R]\!] \tag{4.14b}$$

$$\underline{V_n} \leq |V_n^{\boldsymbol{R}_k}| \leq \overline{V_n} \qquad n \in \mathcal{R}_k \cup \mathcal{O}_k,\ k \in [\![1, R]\!] \tag{4.14c}$$

$$\underline{S_i} \leq S_i \leq \overline{S_i} \qquad i \in \mathcal{C}_n,\ n \in \mathcal{R}_k,\ k \in [\![1, R]\!] \tag{4.14d}$$

$$\sum_{i \in \mathcal{C}_n} S_i = \sum_{m \in \mathcal{N}_n} S_{n,m} \qquad n \in \mathcal{R}_k,\ k \in [\![1, R]\!] \tag{4.14e}$$

$$S_{n,m} = Y_{nm}^* V_n^{\boldsymbol{R}_k} (V_n^{\boldsymbol{R}_k} - V_m^{\boldsymbol{R}_k})^* \qquad m \in \mathcal{N}_n,\ n \in \mathcal{R}_k,\ k \in [\![1, R]\!] \tag{4.14f}$$

$$\underline{S_{n,m}} \leq |S_{n,m}| \leq \overline{S_{n,m}} \qquad m \in \mathcal{N}_n,\ n \in \mathcal{R}_k,\ k \in [\![1, R]\!] \tag{4.14g}$$

$$\underline{\theta_{nm}} \leq \angle(V_n^{\boldsymbol{R}_k} V_m^{\boldsymbol{R}_k,*}) \leq \overline{\theta_{nm}} \qquad m \in \mathcal{N}_n,\ n \in \mathcal{R}_k,\ k \in [\![1, R]\!] \tag{4.14h}$$

$$\angle V_n = 0 \qquad n \in \mathcal{N}_{ref} \tag{4.14i}$$

Le problème de l'Optimal Power Flow (4.1) est reformulé en (4.14) en posant des copies des tensions limitrophes de chaque région $v_{n,l}^{\boldsymbol{R}_k}$. On s'inspire, pour la décomposition du problème avec la méthode ADMM, de l'algorithme pair à pair présenté dans le Chapitre 3

et issu des travaux de Baroche [45] : la contrainte (4.14b) permet d'atteindre le consensus sur les tensions limitrophes, en créant pour chaque paire de régions reliées par un lien l une variable duale distincte. Les contraintes (4.14c) à (4.14i) sont équivalentes aux contraintes (4.1b) à (4.1h) du problème OPF d'origine.

Algorithme 4.3 OPF décomposé par frontières synchrone, local à la région $\boldsymbol{R}_k$

tant que la convergence n'est pas atteinte

si $t = 0$ **alors**

Initialiser les tensions locales $\mathbf{v}^{\boldsymbol{R}_k} = \left[v_{n,l}^{\boldsymbol{R}_k}\right]_{n\in\mathcal{O}_k, l\in\mathcal{L}_n}$ et variables duales $\boldsymbol{\lambda}^{\boldsymbol{R}_k}$

sinon

Mettre à jour les tensions locales via

$$\mathbf{x}^{temp} \leftarrow \arg\min_{\mathbf{x}} \sum_{n\in\mathcal{R}_k}\sum_{i\in\mathcal{C}_n} f_i(P_i) + \frac{\rho}{2}\sum_{n\in\mathcal{O}_k}\sum_{\substack{l\in\mathcal{L}_n \\ l=(\boldsymbol{R}_k,\boldsymbol{R}_k')}} \left| V_n - \frac{v_{n,l}^{\boldsymbol{R}_k} + v_{n,l}^{\boldsymbol{R}_k'}}{2} + \frac{\lambda_{n,l}^{\boldsymbol{R}_k}}{\rho}\right|^2 \tag{4.15a}$$

$$\text{selon } \mathbf{x} : \begin{cases} V_n \in \mathbb{C} & n \in \mathcal{R}_k \cup \mathcal{O}_k \\ S_i = P_i + jQ_i \in \mathbb{C} & n \in \mathcal{O}_k,\ i \in \mathcal{C}_n \\ S_{nm} \in \mathbb{C} & n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \end{cases}$$

$$\text{tel que } \underline{V_n} \le |V_n| \le \overline{V_n} \qquad n \in \mathcal{R}_k \cup \mathcal{O}_k \tag{4.15b}$$

$$\underline{S_i} \le S_i \le \overline{S_i} \qquad n \in \mathcal{R}_k,\ i \in \mathcal{C}_n \tag{4.15c}$$

$$\sum_{i\in\mathcal{C}_n} S_i = \sum_{m\in\mathcal{N}_n} S_{nm} \qquad n \in \mathcal{R}_k \tag{4.15d}$$

$$S_{nm} = Y_{nm}^* V_n (V_n - V_m)^* \qquad n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \tag{4.15e}$$

$$\underline{S_{nm}} \le |S_{nm}| \le \overline{S_{nm}} \qquad n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \tag{4.15f}$$

$$\underline{\theta_{nm}} \le \angle(V_n V_m^*) \le \overline{\theta_{nm}} \qquad n \in \mathcal{R}_k,\ m \in \mathcal{N}_n \tag{4.15g}$$

$$\angle V_n = 0 \qquad n \in \mathcal{R}_k \cap \mathcal{N}_{ref} \tag{4.15h}$$

$$v_{n,l}^{\boldsymbol{R}_k} \leftarrow V_n^{temp} \qquad n \in \mathcal{O}_k,\ l \in \mathcal{L}_n \tag{4.15i}$$

fin si

Envoyer les valeurs limitrophes $v_{n,l}^{\boldsymbol{R}_k}$, $n \in \mathcal{O}_k$, $l \in \mathcal{L}_n$ aux régions voisines

tant que $v_{n,l}^{\boldsymbol{R}_k'}$, $n \in \mathcal{O}_k$, $l = (\boldsymbol{R}_k, \boldsymbol{R}_k') \in \mathcal{L}_n$ ne sont pas reçus des régions voisines $\boldsymbol{R}_k'$

Attendre

fin tant que

Évaluer les variables duales via

$$\lambda_{n,l}^{\boldsymbol{R}_k} \leftarrow \lambda_{n,l}^{\boldsymbol{R}_k} + \rho\frac{v_{n,l}^{\boldsymbol{R}_k} - v_{n,l}^{\boldsymbol{R}_k'}}{2} \qquad n \in \mathcal{O}_k,\ l = (\boldsymbol{R}_k, \boldsymbol{R}_k') \in \mathcal{L}_n \tag{4.16}$$

$t \leftarrow t + 1$

fin tant que

Après avoir appliqué la méthode ADMM sur chaque contrainte de (4.14b), on obtient des équations locales itératives qui sont présentées dans l'Algorithme 4.3. De manière similaire au précédent algorithme, on réalise dans un premier temps une mise à jour des variables copiées $v_{n,l}^{\boldsymbol{R}_k}$ locales à la région $\boldsymbol{R}_k$ en effectuant un calcul d'Optimal Power Flow en (4.15a) appliqué seulement sur les nœuds de la région ainsi que les nœuds voisins, avec une fonction coût différente du problème classique d'OPF. Cette fonction à minimiser comprend deux termes : le premier concerne les fonctions coût des consomacteurs locaux à la région considérée, le second concerne le consensus ADMM entre les variables frontalières copiées avec les autres régions. Ensuite vient une étape de communication de ces variables entre les régions voisines. Une fois que les valeurs mises à jour de toutes les copies des régions voisines sont reçues localement, il est alors possible d'effectuer la mise à jour des variables duales $\lambda_{n,l}^{\boldsymbol{R}_k}$ via (4.16).

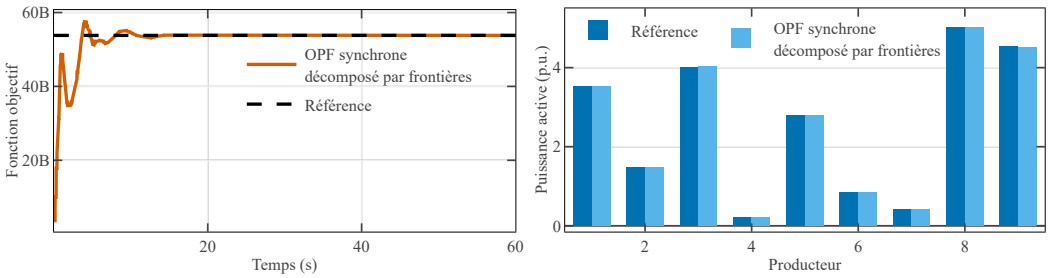
Nota

L'algorithme présenté décompose le réseau électrique en régions. Cependant, il est aussi possible de considérer le cas où l'on assigne une région par nœud électrique, ce qui correspond à une décomposition par nœuds du problème de l'OPF. Cela était difficilement réalisable pour l'algorithme précédent étant donné qu'il était nécessaire de régler la valeur d'un coefficient w_k pour chaque région, et que l'avancement de l'algorithme était très sensible par rapport aux valeurs de ces coefficients.

Nota

Pour chaque lien l entre les régions $\boldsymbol{R}_k$ et $\boldsymbol{R}_k^{'}$, lorsqu'il y a plusieurs valeurs de nœuds à envoyer à la région $\boldsymbol{R}_k^{'}$, on suppose que ces valeurs sont envoyées au sein d'un même message, c'est-à-dire qu'elles sont envoyées en même temps et mettent la même durée pour arriver à destination.

On applique l'Algorithme 4.3 au cas test à 118 nœuds, 53 générateurs et 186 lignes, décomposé en 5 régions et présenté plus en détails dans l'Annexe A.II. On considère des délais de communications égaux à $50\,ms$. Le résultat est affiché en Figure 4.13. On observe que le résultat final obtenu par l'algorithme décentralisé est équivalent au résultat final obtenue de manière centralisée synchrone.

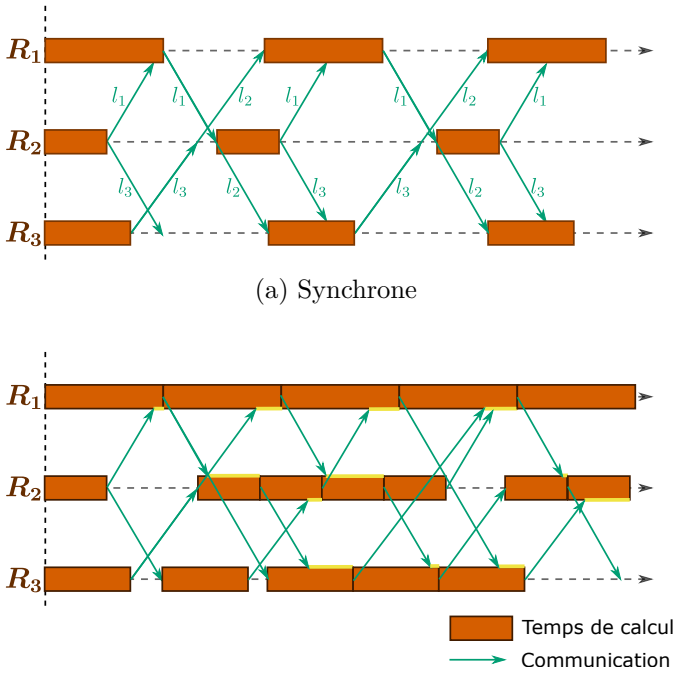


(a) (b)

FIGURE 4.13 – Résultat de l'algorithme OPF décentralisé par frontières synchrone appliqué au cas test à 118 nœuds et comparaison par rapport à la solution optimale centralisée synchrone. En Figure 4.13a, la valeur de la fonction objectif globale est affichée au cours du temps. En Figure 4.13b, les puissances actives de 9 producteurs au bout de 60 secondes de calculs sont comparées à la solution optimale.

4.5.2 Version asynchrone

De la même manière que pour l'algorithme précédent, chaque itération présente un temps de latence dû à l'attente des messages des régions voisines. Il suffit qu'une des régions voisines mette plus de temps à renvoyer un message pour que l'avancement de l'algorithme en intégralité soit ralenti. Il est donc intéressant de pouvoir lancer les calculs locaux avant d'avoir reçu tous les messages, de manière à écourter l'attente entre chaque itération.



(a) Synchrone

(b) Asynchrone

FIGURE 4.14 – Illustration du temps de calcul et des échanges de messages dans le cas à 3 régions de la Figure 4.12 en synchrone et en asynchrone. En synchrone, les calculs sont déclenchés après la réception de deux messages (venant des deux régions voisines). En asynchrone, les calculs ne sont déclenchés qu'après la réception d'un seul message. Le déclenchement de nouveaux calculs ne peut se faire que lorsque le calcul en cours est terminé.

Le critère de déclenchement d'une itération en asynchrone peut être soit une limite de temps d'attente depuis l'itération précédente, un nombre de messages minimal à recevoir, ou bien une combinaison des deux. Par souci de simplicité, nous définissons le critère de déclenchement comme étant le nombre de messages reçus depuis la dernière itération. On illustre aussi les échanges de messages en Figure 4.14 dans les cas synchrone et asynchrone. On devine intuitivement que le cas asynchrone permet de fortement réduire les délais d'attente entre chaque itération.

On note $\mathcal{A}_k^t$ l'ensemble des paires d'indice (n, l) dont la région $\boldsymbol{R}_k$ a reçu une mise à jour durant l'itération t. Le paramètre d'asynchronisme δ est alors défini comme étant la proportion minimale de messages reçus par rapport au nombre total de messages attendus

dans le cas synchrone :

$$|\mathcal{A}_k^t| \geq \lceil\, \delta \cdot \sum_{n\in\mathcal{O}_k} |\mathcal{L}_n| \,\rceil \tag{4.17}$$

Ici, $\sum_{n\in\mathcal{O}_k} |\mathcal{L}_n|$ désigne le nombre total de paires (n,l) avec $n \in \mathcal{O}_k$ étant un nœud frontalier de la région k et l un lien reliant ce nœud avec une des régions voisines. On considère le paramètre d'asynchronisme δ indépendant de la région considérée et constant tout au long de la résolution de l'algorithme.

Algorithme 4.4 OPF décomposé par frontières asynchrone, local à la région $\boldsymbol{R}_k$

tant que la convergence n'est pas atteinte
  si $t = 0$ **alors**
    Initialiser les tensions locales $\mathbf{v}^{\boldsymbol{R}_k} = \left[v_{n,l}^{\boldsymbol{R}_k}\right]_{n\in\mathcal{O}_k, l\in\mathcal{L}_n}$
    Initialiser les variables duales $\boldsymbol{\lambda}^{\boldsymbol{R}_k} = \left[\lambda_{n,l}^{\boldsymbol{R}_k}\right]_{n\in\mathcal{O}_k, l\in\mathcal{L}_n}$
    $\mathcal{A}_k^0 = \{(n,l)\,|\, n \in \mathcal{O}_k,\, l \in \mathcal{L}_n\}$
  sinon
    Mettre à jour les tensions locales via

$$\mathbf{x}^{temp} \leftarrow \arg\min_{\mathbf{x}} \sum_{n\in\mathcal{R}_k}\sum_{i\in\mathcal{C}_n} f_i(P_i) + \frac{\rho}{2}\sum_{n\in\mathcal{O}_k}\sum_{\substack{l\in\mathcal{L}_n \\ l=(\boldsymbol{R}_k,\boldsymbol{R}_k')}} \left| V_n - \frac{v_{n,l}^{\boldsymbol{R}_k} + v_{n,l}^{\boldsymbol{R}_k'}}{2} + \frac{\lambda_{n,l}^{\boldsymbol{R}_k}}{\rho}\right|^2 \tag{4.18a}$$

$$\text{selon } \mathbf{x} : \begin{cases} V_n \in \mathbb{C} & n \in \mathcal{R}_k \cup \mathcal{O}_k \\ S_i = P_i + jQ_i \in \mathbb{C} & n \in \mathcal{O}_k,\, i \in \mathcal{C}_n \\ S_{nm} \in \mathbb{C} & n \in \mathcal{R}_k,\, m \in \mathcal{N}_n \end{cases}$$

    tel que (4.15b) – (4.15h)

$$v_{n,l}^{\boldsymbol{R}_k} \leftarrow V_n^{temp} \qquad (n,l) \in \mathcal{A}_k^t \tag{4.18b}$$

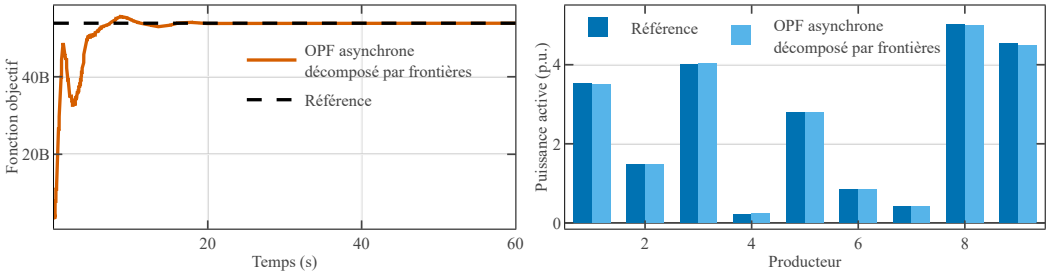
  fin si
  Envoyer les valeurs limitrophes $v_{n,l}^{\boldsymbol{R}_k}$, $(n,l) \in \mathcal{A}_k^t$ aux régions voisines concernées
  tant que les $v_{n,l}^{\boldsymbol{R}_k}$, $(n,l) \in \mathcal{A}_k^{t+1}$ ne sont pas toutes reçues des régions voisines concernées
    Attendre
  fin tant que
  Évaluer les variables duales via

$$\lambda_{n,l}^{\boldsymbol{R}_k} \leftarrow \lambda_{n,l}^{\boldsymbol{R}_k} - \rho\frac{v_{n,l}^{\boldsymbol{R}_k} - v_{n,l}^{\boldsymbol{R}_k'}}{2} \qquad (n,l) \in \mathcal{A}_k^{t+1},\, l = (\boldsymbol{R}_k, \boldsymbol{R}_k') \in \mathcal{L}_n \tag{4.19}$$

  $t \leftarrow t + 1$
fin tant que

L'algorithme asynchrone de chaque région est présenté dans l'Algorithme 4.4. À chaque itération, seules les valeurs d'indice $(n,l) \in \mathcal{A}_k^t$ sont mises à jour, comme en témoignent

les équations (4.18b) et (4.19). Aussi, une région $\boldsymbol{R}_k$ ne communique la valeur $v_{n,l}^{\boldsymbol{R}_k}$ que si $(n,l) \in \mathcal{A}_k^t$, c'est-à-dire si un message de la région voisine associée a été reçu par $\boldsymbol{R}_k$ concernant le nœud n pour l'itération courante t.



(a) (b)

FIGURE 4.15 – Résultat de l'algorithme OPF décentralisé par frontières asynchrone à $\delta = 10\%$ appliqué au cas test à 118 nœuds et et comparaison par rapport à la solution optimale. En Figure 4.15a, la valeur de la fonction objectif globale est affichée au cours du temps. En Figure 4.15b, les puissances actives de 9 producteurs au bout de 60 secondes de calculs sont comparées à la solution optimale.

On affiche en Figure 4.15 le résultat de l'algorithme asynchrone appliqué à un cas test à 118 nœuds et décomposé en 5 régions. On observe que l'algorithme asynchrone converge bien vers la solution optimale centralisée synchrone.

Le critère de convergence choisi pour l'étude de cet algorithme est de $\varepsilon = 10^{-4}$. Toutes les études qui suivent sont réalisées sur le cas test IEEE à 14 nœuds, 5 générateurs et 20 lignes, explicité en Annexe A.III. Comme pour l'algorithme OPF décomposé par nœuds, les problèmes d'optimisation locaux sont résolus via le package Julia `PowerModels` [106] qui se base sur le solveur `Ipopt` [108] utilisant une méthode de points intérieurs. Les différentes valeurs des paramètres de ces études sont compilées dans le Tableau 4.4. On note que le modèle de délais de communication utilisé donne un délai de communication moyen d'environ 100 ms.

Section	**Cas**	Δ_{comm}	Δ_{calc}
4.5.3	14 nœuds 14 régions	$\alpha = 100$ $\beta = 50$ ms Déterministe $p_{pertes} = 0\%$	Modèle affine 2.2
4.5.4	14 nœuds 14 régions	$\alpha = 100$ $\beta = 50$ ms Déterministe avec pertes stochastiques	Modèle affine 2.2
4.5.5	14 nœuds	$\alpha = 100$ $\beta = 50$ ms Déterministe avec pertes stochastiques $p_{pertes} = 1\%$	Modèle affine 2.2

TABLEAU 4.4 – Paramètres utilisés pour les simulations de l'algorithme OPF décomposé par frontières.

On réalise une étude de l'influence du paramètre d'asynchronisme en section 4.5.3. On étudie aussi l'influence du taux de pertes de messages en section 4.5.4. Enfin, on applique l'algorithme sur plusieurs partitionnements du même cas d'étude en section 4.5.5.

4.5.3 Influence du paramètre d'asynchronisme

Dans un premier temps, on étudie une version de l'algorithme où une région est attribuée à un nœud à la fois. Ainsi, le cas étudié comporte 14 régions pour 14 nœuds. De plus, pour cette étude le taux de pertes de messages est nul.

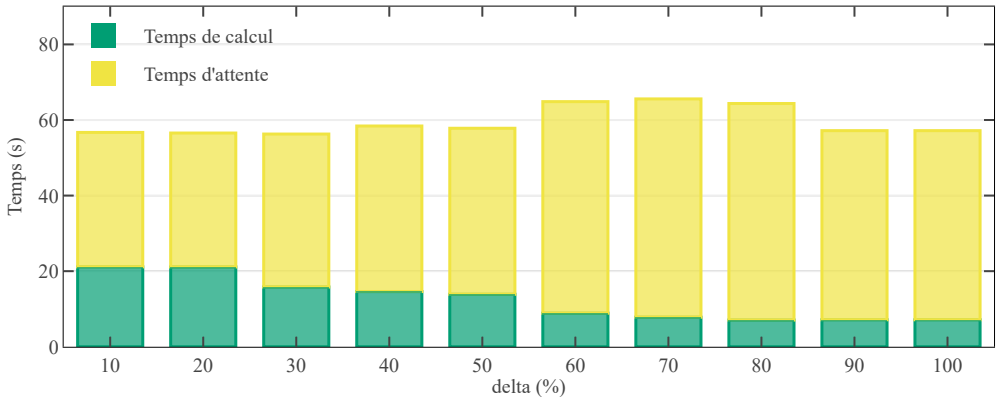


FIGURE 4.16 – Temps de convergence, constitué du temps de calcul moyen et du temps d'attente moyen, en fonction du paramètre d'asynchronisme δ, de l'algorithme OPF asynchrone, appliqué au cas test à 14 nœuds.

Le temps de convergence en fonction du paramètre d'asynchronisme δ est affiché en Figure 4.16, avec la part de temps d'attente en jaune, et celle de temps de calculs en vert. On n'observe pas de différences significatives entre le temps de convergence synchrone à $\delta = 100\%$ et le temps de convergence asynchrone. On a cependant bien diminué les temps d'attente, comme l'illustrait la Figure 4.14.

Le temps d'attente légèrement plus élevé pour les cas à $\delta \in [60, 80]\%$ vient de l'agencement du réseau électrique. En effet, par exemple en s'appuyant sur la Figure III.2 en Annexe qui représente les différents nœuds ainsi que leurs connexions physiques, pour $\delta = 80\%$, toutes les régions sauf celle associée au nœud n_4 fonctionnent en synchrone puisque tous les nœuds sauf le nœud n_4 possèdent 4 voisins ou moins. Ainsi, la région associée à n_4 doit attendre au minimum 4 messages sur ses 5 voisins. Or, ces voisins ont un fonctionnement synchrone donc ils seront forcément ralentis par n_4 : la région qui a envoyé le cinquième message à n_4 devra attendre que cette dernière finisse le calcul précédent avant de prendre en compte son message dans le calcul qui suit et qu'elle lui renvoie la valeur mise à jour.

On constate aussi dans la Figure 4.16 une augmentation du temps de calcul à mesure que l'asynchronisme augmente, c'est-à-dire à mesure que le paramètre δ diminue. En effet, du fait de l'asynchronisme, les régions possèdent des informations manquantes à chaque itération, ce qui les mènent à réaliser plus de calculs à niveau de convergence égal. C'est la même remarque que l'on a fait dans l'étude du paramètre d'asynchronisme de l'algorithme OPF décomposé par nœuds en section 4.4.3.

4.5.4 Influence du taux de pertes de messages

On fait maintenant varier le taux de pertes de messages p_{pertes} en Figure 4.17, et on compare les temps de convergence en asynchrone à $\delta = 10\%$ et en synchrone à $\delta = 100\%$. On rappelle que lorsqu'un message est envoyé par un agent A et qu'il se perd dans le réseau de communication, il est renvoyé par l'agent A au bout de deux fois le délai de communication sans pertes. Ainsi, le délai de communication total avec perte est de trois fois le délai de communication sans pertes. On remarque comme dans la section 4.4.4 que la version synchrone est beaucoup plus sensible au taux de perte de message que la version asynchrone. L'augmentation du temps de convergence entre $p_{pertes} = 0\%$ et $p_{pertes} = 5\%$ est de 73% en synchrone contre 15% en asynchrone.

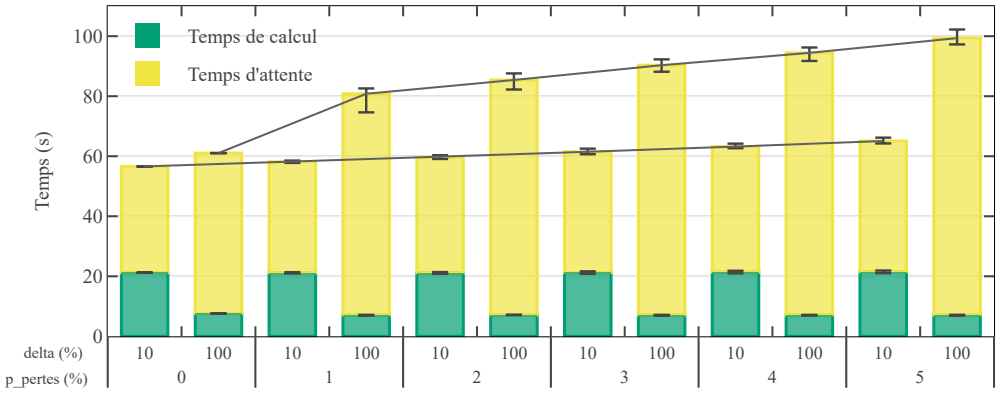


FIGURE 4.17 – Temps de convergence, constitué du temps de calcul moyen et du temps d'attente moyen, en fonction du taux de pertes de messages p_{pertes} et pour deux valeurs du paramètre d'asynchronisme $\delta = 10\%$ ou $\delta = 100\%$, appliqué au cas test à 14 nœuds. Les barres d'erreur représentent l'écart-type sur les différents délais.

Lorsque le taux de pertes de message augmente, les variations de délais de communication augmentent à médiane constante. En effet, on rappelle comme dans la section 4.4.4 que lorsque p_{pertes} est proche de zéro, peu de messages sont perdus donc sur l'ensemble des échanges, la valeur médiane des délais de communication reste alors identique à celle où il n'y a pas de pertes. Ainsi, les délais de communication deviennent plus variés mais conservent la même valeur médiane, et l'on voit en Figure 4.17 que le temps d'attente de l'algorithme synchrone est fortement affecté par ces variations de délais, là où l'algorithme asynchrone l'est beaucoup moins. Cela peut s'expliquer par le fait qu'en synchrone, l'intégralité des messages doit être attendu à chaque itération, donc le temps d'attente est fixé par le dernier message qui arrive, et dans ce cas, il s'agit du message qui a été perdu une ou plusieurs fois. En revanche, en asynchrone, le message perdu arrivera pour une itération ultérieure car l'agent destinataire ne l'attend pas pour réaliser ses calculs, ce qui écourte ses délais d'attente.

L'algorithme asynchrone est plus robuste aux variations de délais de communication en termes de temps de convergence global.

4.5.5 Influence du partitionnement

Dans les sections précédentes, nous avons étudié un cas où chaque région se voyait attribuer un unique nœud, réalisant la démonstration d'un partitionnement par nœuds. Or, l'algorithme présenté en (4.14) permet de considérer des partitionnements par régions. Nous allons alors procéder à divers partitionnements du cas IEEE 14 nœuds, où le nombre de régions peut aller de 3 à 13, et réaliser une étude selon un procédé de Monte-Carlo pour en déduire des statistiques sur le temps de convergence de l'algorithme synchrone et de l'algorithme asynchrone en fonction des partitionnements. La génération des partitionnements est précisée en Annexe A.III.

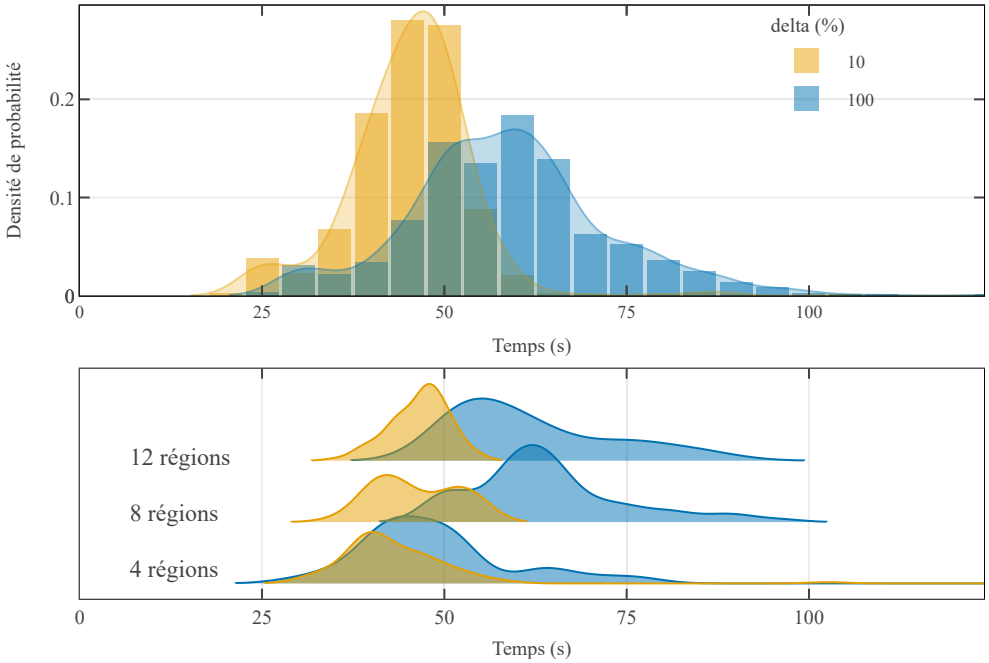


FIGURE 4.18 – Densité de probabilité du temps de convergence pour l'algorithme asynchrone décomposé par frontières à $\delta = 10\%$ et à $\delta = 100\%$, à taux de pertes de messages fixe de $p_{pertes} = 1\%$. La première figure représente l'histogramme et la densité de probabilité du temps de convergence pour tous les partitionnements sans distinction, la seconde figure présente la densité de probabilité pour des partitionnements de 4, 8 ou 12 régions séparément.

La Figure 4.18 représente la densité de probabilité du temps de convergence de l'algorithme synchrone (en bleu) et de l'algorithme asynchrone (en orange). On observe sur la première figure la densité de probabilité pour tous les partitionnements confondus. Le temps de convergence est en moyenne plus petit pour la version asynchrone. La queue de densité est aussi plus longue en synchrone qu'en asynchrone, ce qui montre que la variance du temps de convergence est plus élevée en synchrone. La seconde figure précise les densités de probabilité pour des partitionnements de 4, 8 et 12 régions. On y observe que plus le nombre de régions est grand, plus on met de temps à converger en général. Aussi, la moyenne de temps de convergence est systématiquement plus faible en asynchrone.

La version asynchrone de l'algorithme est statistiquement plus rapide que la version synchrone lorsque l'on considère différents partitionnements.

4.5.6 Conclusions sur l'algorithme OPF asynchrone décomposé par frontières

L'algorithme OPF décentralisé décomposé par frontières présenté dans cette partie permet de réaliser une version asynchrone de manière moins contraignante que pour l'algorithme décomposé par nœuds. Il est aussi plus facile à mettre en place étant donné qu'il n'y a pas de paramètre à retrouver pour chaque région. Cela permet une grande liberté dans le partitionnement du problème : il est possible de réaliser un partitionnement par nœuds, où un agent de calcul est associé à chaque nœud physique du réseau, ou bien un partitionnement par régions.

Les simulations de cet algorithme ont été réalisées sur un partitionnement par nœuds, c'est-à-dire où chaque région se voit affecter un seul nœud. Les conclusions de ces simulations sont similaires à celles de l'algorithme OPF précédent décomposé par nœuds : l'implémentation asynchrone est moins sensible que l'implémentation synchrone aux aléas de communication en termes de temps de convergence. La facilité de mise en œuvre de l'algorithme a aussi permis de réaliser une étude sur de nombreux partitionnements. Le temps de convergence asynchrone est statistiquement plus petit que son homologue synchrone, quel que soit le nombre de régions du partitionnement.

4.6 Conclusions et perspectives du chapitre

En conclusion, le problème de l'Optimal Power Flow est un problème complexe mêlant des contraintes physiques du réseau électrique aux coûts économiques du marché de l'électricité. La résolution de ce problème est difficilement tenable en centralisé à grande échelle, et nécessite sa décentralisation en partitionnant le problème soit en régions, soit en nœuds. Deux algorithmes décentralisés d'OPF sont analysés dans ce chapitre. Ils nécessitent tous deux un grand nombre de communications pour pouvoir converger vers la solution optimale attendue, ce qui les rend potentiellement vulnérables aux aléas de communication. L'implémentation asynchrone permet de limiter cette vulnérabilité en termes de temps de calculs, en ce qu'elle diminue fortement les délais d'attente entre chaque itération.

Les résultats en temps de calcul sont largement dépendants des cas test sur lesquels on applique les algorithmes, ainsi que du modèle de délais de communication et de calculs utilisés. Cependant, on a pu montrer que l'augmentation des délais de communication avait une influence moins forte sur les versions asynchrones des algorithmes. De plus, cet avantage de l'implémentation asynchrone semble aussi être valable quelque soit le partitionnement du problème.

Ces algorithmes mériteraient d'être testés sur des cas test plus conséquents pour en vérifier la faisabilité à grande échelle. Il serait aussi intéressant de se pencher sur d'autres types d'OPF comme ceux basés sur la programmation MILP (Mixed Integer Linear Programming) ou MINLP (Mixed Integer Non Linear Programming) qui prennent en compte des variables booléennes (pour modifier le ratio des transformateurs par exemple). L'implémentation en ligne avec initialisation en *warm start*, c'est-à-dire initialisé au point de fonctionnement du pas de temps précédent, serait une suite logique des études réalisées dans ce chapitre, cela pourrait potentiellement accélérer la résolution et permettre de lancer le calcul au plus proche de l'instant de livraison et ainsi minimiser les erreurs de prédiction.

Chapitre 5

Marché de l'électricité endogène asynchrone

Sommaire

Nous nous sommes concentrés jusqu'ici sur la simulation d'algorithmes décentralisés soumis à des aléas de communication, notamment des algorithmes résolvant le problème de marché pair à pair au Chapitre 3 et le problème d'*Optimal Power Flow* au Chapitre 4.

La résolution décentralisée du marché de l'électricité permet à chaque producteur, consommateur ou consomacteur d'avoir un rôle dans le marché tout en évitant de communiquer des données privées à un parti tierce, mais peut cependant donner des solutions non applicables sur le réseau électrique. L'OPF, en revanche, permet de trouver une solution qui minimise les coûts des producteurs et consommateurs, tout en respectant les contraintes physiques du réseau électrique. Cela se fait en échange d'une augmentation du nombre de variables dans le problème considéré, ainsi que d'un ajout de contraintes non linéaires qui rendent le problème beaucoup plus complexe à résoudre. La résolution décentralisée de l'OPF permet de pallier le problème de mise à l'échelle, cependant le problème de confidentialité des données est toujours d'actualité. En effet, si la décomposition se fait par région, alors les agents doivent communiquer leurs données privées avec le gestionnaire de la région, et si la décomposition se fait par nœud, alors il se peut que plusieurs producteurs et consommateurs soient connectés au même nœud, ce qui conduit de même au partage des données privées.

On propose dans ce chapitre de se pencher sur le problème de marché pair à pair endogène, qui consiste en la résolution d'un marché où un des agents, l'opérateur système SO, s'occupe des contraintes physiques du réseau (4.1b)-(4.1h) présentées dans le chapitre précédent. Les agents classiques du marché fonctionnent de la même manière que présenté dans le Chapitre 3, c'est-à-dire qu'ils se mettent d'accord entre eux sur les volumes d'électricité échangés, mais communiquent aussi à chaque itération avec l'agent SO, qui les incite à modifier leurs solution en puissance de manière à ce que la solution finale respecte les contraintes physiques. On peut alors montrer que la solution finale est équivalente à la solution du problème de l'Optimal Power Flow, étant donné que l'on respecte les mêmes contraintes physiques.

Deux types d'algorithmes pair à pair endogènes sont présentés.

— **Marché pair à pair endogène avec SO centralisé** : l'opérateur système est représenté par un unique agent qui calcule les contraintes du réseau complet. Cela revient à résoudre un problème d'OPF avec une fonction à minimiser modifiée. Les fonctions coût des agents du marché restent privées mais on retrouve le problème de mise à l'échelle et de communication centralisée, puisque tous les agents du marché doivent communiquer avec ce même agent opérateur système.
— **Marché pair à pair endogène avec SO décentralisé** : l'opérateur système est décomposé en plusieurs opérateurs qui s'occupent de résoudre la partie OPF entre eux d'un côté, et qui communiquent avec les agents locaux pour les inciter à choisir une solution faisable du point de vue du réseau. La séparation du problème d'OPF en plusieurs régions permet de pouvoir considérer des réseaux électriques à grande échelle.

L'effet des aléas de communication sur la résolution de cet algorithme endogène est d'autant plus important que le nombre d'échanges augmente grandement. En effet, on considère les nombreuses communications entre pairs présentées au Chapitre 3 et on y ajoute des communications entre chaque pair et l'opérateur système, centralisé ou bien l'opérateur en charge de la région dans laquelle se trouve l'agent. Dans le cas où l'opérateur système est décentralisé, on y rajoute aussi les communications ayant lieu entre les différents opérateurs locaux. Dans une volonté d'implémentation réaliste de ces algorithmes décentralisés, il faut que l'on traite les problèmes liés aux aléas de communication. Pour cela et comme dans les chapitres précédents, on propose d'étudier les versions asynchrones de ces algorithmes pour minimiser les effets des aléas sur le temps de convergence.

Dans les sections qui suivent, on commencera d'abord par présenter une formulation du marché pair à pair endogène en section 5.1 avant de présenter les études réalisées sur le cas avec opérateur système centralisé en section 5.2 et de présenter les résultats préliminaires du cas avec opérateur système décentralisé en section 5.3.

5.1 Formulation du marché pair à pair endogène

Le problème de marché endogène est introduit pour la première fois dans le manuscrit de thèse de Thomas Baroche [46]. La solution du problème de marché endogène est équivalente à celle de l'Optimal Power Flow donné en (4.1). Il s'agit de formuler un marché de l'électricité pair à pair, comme celui présenté dans le Chapitre 3, tout en ajoutant les contraintes physiques du réseau électrique à l'intérieur du problème d'optimisation, d'où le terme *endogène* utilisé. Le reste du problème est décomposé en échanges pair à pair entre les différents agents du marché, ou consomacteurs. La formulation du problème de marché endogène est donné en (5.1). On rappelle que $\mathcal{C}$ désigne l'ensemble des agents du réseau électrique (producteurs, consommateurs) et $\mathcal{C}_i$ désigne l'ensemble des partenaires commerciaux de l'agent i.

Formulation du marché endogène

$$\min \sum_{i\in\mathcal{C}} f_i(P_i, Q_i) + \zeta_{AC}(P, Q) \tag{5.1a}$$

$$\text{selon } S_i = P_i + jQ_i \qquad i \in \mathcal{C}$$

$$\mathbf{P} = [p_{ij}]_{i,j\in\mathcal{C}}$$

$$\mathbf{W} = [w_{ij}]_{i,j\in\mathcal{C}}$$

$$\text{tel que } (\mathbf{W} - \mathbf{W}^\top)/2 = \mathbf{P} \tag{5.1b}$$

$$P_i = \sum_{j\in\mathcal{C}_i} p_{ij} \qquad i \in \mathcal{C} \tag{5.1c}$$

$$\underline{S_i} \leq S_i \leq \overline{S_i} \qquad i \in \mathcal{C} \tag{5.1d}$$

$$p_{Loss} = -\sum_{i\in\mathcal{C}} p_i \tag{5.1e}$$

avec $\zeta_{AC}(P, Q)$ la fonction de régularisation, égale à zéro si les contraintes réseaux définies dans le Chapitre 4 (4.1b)-(4.1h) sont bien vérifiées et égale à $+\infty$ sinon. Ce terme est calculé par un agent représentant l'opérateur du réseau électrique, noté *SO* pour *system operator*, qui est donc un agent différent des participants du marché. La variable $\mathbf{W}$ est la variable qui sert de copie aux termes d'échanges $\mathbf{P}$. Elle est rapidement remplacée [1] après avoir appliqué les formules de l'ADMM car on peut prouver que $\frac{w_{ij}^t - w_{ji}^t}{2} = \frac{p_{ij}^t - p_{ji}^t}{2}$ pour toute itération $t > 1$.

Nota

Les variables physiques telles que la tension aux nœuds du réseau et les puissances dans les lignes ne sont pas représentées en (5.1) car elles sont considérées comme internes à la fonction $\zeta_{AC}(P, Q)$.

Étant donné que l'on simule un véritable réseau électrique, il n'est plus possible de considérer que l'ensemble des puissances produites par les producteurs soit consommée par les consommateurs. En effet, une partie de la puissance produite est perdue en tant que pertes dans les lignes. Un nouvel agent du marché "Loss" est alors introduit, il s'occupe de

1. Se référer à la section 3.3.

racheter les puissances perdues sur les lignes du réseau électrique en échangeant avec les producteurs. La puissance perdue dans les lignes est définie en (5.1e) et est déterminée par le SO. On définit alors l'ensemble des pairs participants au marché comme $\mathcal{C}^+ = \mathcal{C} \cup \{Loss\}$.

Nota

L'agent *Loss* se comporte comme n'importe quel consomacteur du marché en ce qu'il réalise des échanges commerciaux avec les autres consomacteurs et qu'il communique la valeur de sa puissance apparente au SO. On considère que sa fonction coût locale f_{Loss} est nulle et qu'il ne cherche qu'à trouver la solution qui vérifie la contrainte (5.1c), avec p_{Loss} imposé en (5.1e).

L'agent SO communique avec les consomacteurs pour que leurs échanges d'énergie ne résultent pas en une violation des contraintes réseau. Tout au long de la résolution, si la solution est incompatible avec les contraintes réseau, le SO encourage les consomacteurs à modifier leurs point de fonctionnement jusqu'à retrouver une solution qui optimise les coûts des consomacteurs tout en satisfaisant les contraintes physiques.

Il existe deux formulations de marché endogène, que l'on développe dans les deux sections qui suivent : la première considère que l'opérateur système SO est pris en charge par un unique agent de calcul, on dit que le SO est centralisé ; la seconde considère l'opérateur système comme l'union de plusieurs agents de calcul qui s'occupent chacun d'une zone du réseau électrique, on dit que le SO est décentralisé. Pour l'algorithme avec SO centralisé, une étude comparative des versions synchrone et asynchrone est menée en section 5.2. Pour l'algorithme avec SO décentralisé, on présentera en section 5.3 la formulation mathématique ainsi que les algorithmes synchrone et asynchrone.

5.2 Résolution décentralisée du marché avec SO centralisé

On se concentre dans un premier temps sur la version avec SO centralisé, dont le schéma des communications est présenté en Figure 5.1.

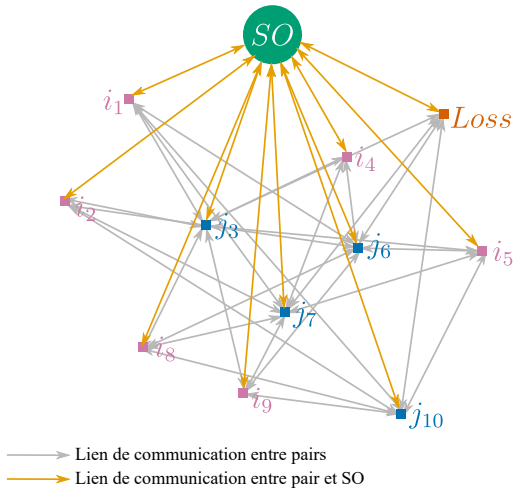


Figure 5.1 – Schéma exemple d'un réseau dont la résolution de marché est effectuée de pair à pair, et où à chaque itération de la résolution, l'opérateur réseau vérifie les contraintes physiques de la solution et influe sur le résultat. Les flèches grises représentent les liens de communication entre les agents du marché. Les flèches oranges représentent les liens de communication entre le SO et les agents. Les agents en rose notés i sont des consommateurs, les agents en bleu notés j sont des producteurs et l'agent *Loss* en orange achète les pertes du réseau aux producteurs.

5.2.1 Version synchrone

Dans un premier temps, on considère que l'opérateur réseau SO est représenté par un unique agent de calcul. Les communications entre les agents du marché et le SO sont alors centralisées, tandis que les communications entre agents du marché se font de pairs à pairs. En décomposant le problème (5.1) entre les consomacteurs et l'agent SO, on rajoute les variables P_i^{SO}, Q_i^{SO} locales au SO, ainsi que les contraintes d'égalité (5.2).

$$P_i^{SO} = P_i \qquad Q_i^{SO} = Q_i \tag{5.2}$$

L'application de la méthode ADMM au problème (5.1) est explicitée dans [46] et mène aux équations itératives (5.3) à (5.8), où $\sigma^\rho(x,y,z) = z(y-x) + \frac{\rho}{2}(y-x)^2$ est une fonction permettant de simplifier les différentes formulations.

$$\begin{aligned}(P_i, Q_i, (p_{ij})_j)^{t+1} = \arg\min_{P_i, Q_i, (p_{ij})_j} f_i(P_i, Q_i) &+ \sum_{j\in\mathcal{C}_i} \sigma^\rho\left(p_{ij}, \frac{p_{ij}^t - p_{ji}^t}{2}, \lambda_{ij}^t\right) \\ &+ \sigma^\rho\left(P_i, \frac{P_i^{SO,t} + P_i^t}{2}, \eta_i^{p,t}\right) \\ &+ \sigma^\rho\left(Q_i, \frac{Q_i^{SO,t} + Q_i^t}{2}, \eta_i^{q,t}\right)\end{aligned} \tag{5.3a}$$

$$\text{tel que } P_i = \sum_{j\in\mathcal{C}_i} p_{ij} \tag{5.3b}$$

$$\underline{P_i} \le P_i \le \overline{P_i} \tag{5.3c}$$

$$\underline{Q_i} \le Q_i \le \overline{Q_i} \tag{5.3d}$$

$$\begin{aligned}(\mathbf{P}^{SO}, \mathbf{Q}^{SO})^{t+1} = \arg\min_{\substack{\mathbf{P}^{SO}=(P_i^{SO})_i \\ \mathbf{Q}^{SO}=(Q_i^{SO})_i}} \zeta_{AC}(\mathbf{P}^{SO}, \mathbf{Q}^{SO}) &+ \sum_{i\in\mathcal{C}} \sigma^\rho\left(\frac{P_i^{SO,t} + P_i^t}{2}, P_i^{SO}, \eta_i^{p,t}\right) \\ &+ \sum_{i\in\mathcal{C}} \sigma^\rho\left(\frac{Q_i^{SO,t} + Q_i^t}{2}, Q_i^{SO}, \eta_i^{q,t}\right) \\ &+ \sum_{i\in\mathcal{C}} \sigma^\rho\left(P_i^{SO,t} - \frac{1}{|\mathcal{C}|}\frac{P_{Loss}^{SO,t} + P_{Loss}^t}{2}, P_i^{SO} - \frac{1}{|\mathcal{C}|}P_{Loss}^{SO,t}, \frac{1}{|\mathcal{C}|}\eta_{Loss}^{p,t}\right)\end{aligned} \tag{5.4}$$

$$P_{Loss}^{SO,t+1} = -\sum_{i\in\mathcal{C}} P_i^{SO,t+1}, \quad Q_{Loss}^{SO,t+1} = Q_{Loss}^t \tag{5.5}$$

$$\lambda_{ij}^{t+1} = \lambda_{ij}^t - \rho\frac{p_{ij}^{t+1} + p_{ji}^{t+1}}{2} \qquad i\in\mathcal{C}^+,\ j\in\mathcal{C}_i \tag{5.6}$$

$$\eta_i^{p,t+1} = \eta_i^{p,t} + \rho\frac{P_i^{SO,t+1} - P_i^{t+1}}{2} \qquad i\in\mathcal{C}^+ \tag{5.7}$$

$$\eta_i^{q,t+1} = \eta_i^{q,t} + \rho\frac{Q_i^{SO,t+1} - Q_i^{t+1}}{2} \qquad i\in\mathcal{C}^+ \tag{5.8}$$

— Les mises à jour des variables locales à chaque consomacteur $i \in \mathcal{C}^+$ sont données en (5.3). Analysons la fonction à minimiser localement (5.3a) : le premier terme représente la fonction coût du consomacteur i que l'on cherche à minimiser localement. Le second terme est obtenu en appliquant la méthode ADMM sur la contrainte de réciprocité des échanges (5.1b), où λ_{ij} est la variable duale de la contrainte $\frac{w_{ij}-w_{ji}}{2}$. Les troisième et quatrième termes sont obtenus en appliquant la méthode ADMM aux contraintes (5.2), avec η_i^p et η_i^q les variables duales associées.
— Les mises à jour des variables locales au SO sont données en (5.4). Il s'agit ici de trouver une solution de manière à ce que les contraintes physiques du réseau soient vérifiées, d'où la présence du terme ζ_{AC}. On retrouve des termes similaires à ceux présents dans la fonction objectif des pairs (les deux premières sommes en (5.4)) qui permettent de réaliser un consensus sur les variables de puissance actives et réactives partagées avec les consomacteurs. La troisième somme permet de déterminer la puissance perdue dans les lignes à partir des puissances actives de tous les autres consomacteurs, et sa variable duale associée est η_{Loss}^p.
— La valeur de puissance active perdue dans les lignes est calculée du côté du SO en (5.5), qui possède une copie de toutes les puissances actives des agents du marché. Elle est ensuite communiquée à l'agent "Loss".

Nota

Les mises à jour des consomacteurs et du SO se font en parallèle, il n'y a pas besoin d'attendre que les agents du marché aient terminé leurs calculs pour que le SO effectue les siens.

Une fois que les calculs des agents du marché ainsi que du SO sont terminés, ces derniers peuvent échanger sur les dernières informations mises à jour, comme illustré en Figure 5.2. Lorsque tous les messages sont arrivés à destination, les variables duales peuvent être mises à jour en utilisant les équations (5.6), (5.7) et (5.8) :
— $\lambda_{ij}, \forall i \in \mathcal{C}^+$, $j \in \mathcal{C}_i$ est la variable duale de la contrainte (5.1b),
— η_i^p et $\eta_i^q \, \forall i \in \mathcal{C}^+$ sont les variables duales des contraintes en (5.2).

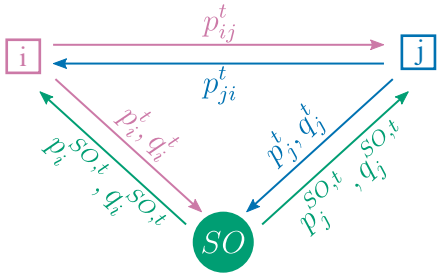


FIGURE 5.2 – Schéma des échanges entre les consomacteurs i et j et l'agent opérateur système SO durant l'itération t.

On obtient alors les Algorithmes 5.1 et 5.2 qui décrivent respectivement le comportement d'un agent du marché (agent *Loss* y compris) et celui du gestionnaire de réseau SO en version synchrone. À chaque itération, un agent i du marché attend toutes les valeurs d'échanges réciproques p_{ji} ainsi que les valeurs de puissances active et réactive en provenance du SO. De même, le SO attend à chaque itération toutes les valeurs de puissances actives et réactives provenant de tous les agents du marché (agent *Loss* y compris).

Algorithme 5.1 Marché endogène avec SO centralisé synchrone, algorithme local à l'agent i

tant que la convergence n'est pas atteinte
  si $t = 0$ **alors**
    Initialiser les échanges locaux $(p_{ij})_{j\in\mathcal{C}_i}$ et variables duales associées $(\lambda_{ij})_{j\in\mathcal{C}_i}$
    Initialiser les puissances locales P_i et Q_i et variables duales associées η_i^p, η_i^q
  sinon
    Mettre à jour les variables locales via

$$P_i, Q_i, (p_{ij})_j \leftarrow (5.3) \tag{5.9}$$

  fin si
  Envoyer les valeurs p_{ij} mises à jour aux agents $j \in \mathcal{C}_i$
  Envoyer les valeurs P_i, Q_i mises à jour à l'agent SO
  tant que les p_{ji} des agents $j \in \mathcal{C}_i$ et les P_i^{SO}, Q_i^{SO} du SO ne sont pas arrivées
    Attendre
  fin tant que
  Évaluer les variables duales via

$$\lambda_{ij} \leftarrow (5.6) \qquad j \in \mathcal{C}_i$$
$$\eta_i^p, \eta_i^q \leftarrow (5.7), (5.8)$$

  $t \leftarrow t + 1$
fin tant que

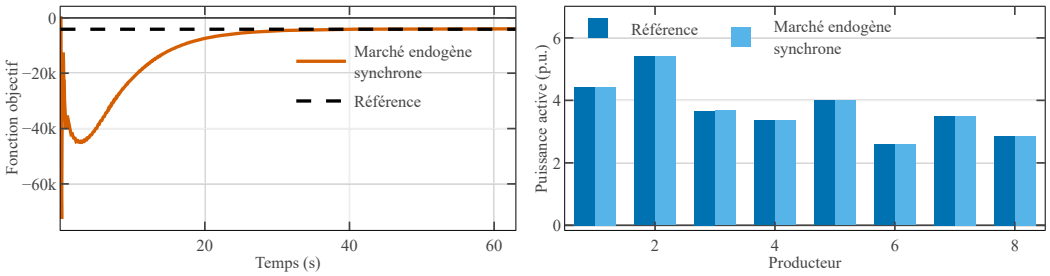
Algorithme 5.2 Marché endogène avec SO centralisé synchrone, algorithme local à l'agent SO

tant que la convergence n'est pas atteinte
  si $t = 0$ **alors**
    Initialiser les puissances actives $(P_i^{SO})_{i\in\mathcal{C}}$ et les variables duales associées $(\eta_i^p)_{i\in\mathcal{C}}$
    Initialiser les puissances réactives $(Q_i^{SO})_{i\in\mathcal{C}}$ et les variables duales associées $(\eta_i^q)_{i\in\mathcal{C}}$
    Initialiser les puissances P_{Loss}^{SO}, Q_{Loss}^{SO} et les variables duales associées η_{Loss}^p, η_{Loss}^q
  sinon
    Mettre à jour les variables locales via
    $\mathbf{P}_{SO}, \mathbf{Q}_{SO} \leftarrow (5.4)$
    $P_{Loss}^{SO}, Q_{Loss}^{SO} \leftarrow (5.5)$
  fin si
  Envoyer les valeurs P_i^{SO}, Q_i^{SO} mises à jour aux agents $i \in \mathcal{C}^+$
  tant que les P_i, Q_i des agents $i \in \mathcal{C}^+$ ne sont pas arrivées
    Attendre
  fin tant que
  Évaluer les variables duales via

$$\eta_i^p, \eta_i^q \leftarrow (5.7), (5.8) \qquad i \in \mathcal{C}^+$$

  $t \leftarrow t + 1$
fin tant que

On applique l'algorithme à un cas test à 39 nœuds, 31 générateurs et 46 lignes, dont les données sont précisées en Annexe A.IV. On prend en compte des délais de communication de l'ordre de 50 ms en utilisant le modèle de la section 2.1. On considère aussi que les temps de calculs des agents du marché sont négligeables compte tenu des délais de communication, sauf pour le SO qui doit effectuer un calcul d'OPF. La Figure 5.3 représente la résolution de l'algorithme de marché pair à pair endogène synchrone, et la comparaison de la solution finale avec la valeur optimale de référence centralisée synchrone, obtenue au travers d'un calcul d'OPF utilisant le package Julia `PowerModels.jl`. D'après l'évolution de la fonction objectif représentée en Figure 5.3a, l'algorithme pair à pair endogène atteint la même valeur objectif que la solution de référence au bout d'environ 30s de calculs (horloge interne à la simulation), ce qui correspond à environ 270 itérations. On vérifie à l'aide de la Figure 5.3b que la solution finale est bien identique à la solution de référence en comparant les puissances des générateurs.



(a) (b)

FIGURE 5.3 – Résultat de l'algorithme synchrone de marché pair à pair endogène appliqué au cas test à 31 agents et comparaison par rapport à la solution optimale d'un OPF centralisé synchrone. En Figure 5.3a, la valeur de la fonction objectif globale est affichée au cours du temps. En Figure 5.3b, les puissances actives de 8 producteurs sont comparées à la solution optimale.

5.2.2 Version asynchrone

Le schéma asynchrone des communications entre pairs est identique à celui du marché pair à pair. On définit le paramètre d'asynchronisme δ_{pair} comme la proportion minimale des messages pris en compte à l'itération t par l'agent i. Ces messages peuvent provenir des pairs voisins (éléments de $\mathcal{C}_i$) ou bien du SO.

$$|\mathcal{A}_i^t| \geq \lceil \delta_{pair} \cdot (|\mathcal{C}_i| + 1) \rceil \tag{5.10}$$

où $\mathcal{A}_i^t \subset \{\mathcal{C}_i \cup \{SO\}\}$ est l'ensemble des agents dont l'agent i a reçu un message pour l'itération t. Une fois que le nombre de messages reçus par un agent est supérieur ou égal à $\lceil \delta_{pair} \cdot (|\mathcal{C}_i| + 1) \rceil$, les calculs de l'itération suivante sont déclenchés.

Quant aux communications entre l'agent SO et les consomacteurs, on définit un nouveau paramètre d'asynchronisme δ_{SO} qui correspond à la proportion de messages reçus par le SO par rapport au nombre total de consomacteurs :

$$|\mathcal{A}_{SO}^t| \geq \lceil \delta_{SO} \cdot |\mathcal{C}^+| \rceil \tag{5.11}$$

où $\mathcal{A}_{SO}^t \subset \mathcal{C}^+$ est l'ensemble des agents dont l'agent SO a reçu un message pour l'itération

t. Une fois que le nombre de messages reçus par le SO est supérieur ou égal à $\lceil \delta_{SO} \cdot |\mathcal{C}^+| \rceil$, les calculs de l'itération suivante du SO sont déclenchés.

Les algorithmes 5.3 et 5.4 décrivent le fonctionnement asynchrone des agents consomacteurs et du SO respectivement. On y retrouve les étapes d'attente de certains messages des agents appartenant à $\mathcal{A}_i^t$ pour les agents du marché, et des agents appartenant à $\mathcal{A}_{SO}^t$ pour le SO centralisé. Aussi, on retrouve des étapes intermédiaires (5.12) et en (5.13) où l'on effectue les calculs d'itération et on les sauvegarde dans des variables temporaires $.^{temp}$. Ces variables temporaires permettre de stocker les résultats des problèmes locaux et de décider a posteriori quels éléments des variables locales seront effectivement mises à jour. Par exemple, le vecteur $(p_{ij})_{j\in\mathcal{C}_i}^{temp}$ possède $|\mathcal{C}_i|$ éléments, et seuls les éléments liés à des agents voisins j dont on a reçu un message avant l'itération sont mis à jour. Les variables locales primales et duales sont par la suite mises à jour seulement si elles sont associées à des agents dont on a reçu un message durant l'itération concernée. On remarque que l'on doit faire une distinction parmi les messages reçus par un agent du marché : si l'un de ces messages provient du SO, alors on peut mettre à jour les variables locales de puissances active et réactive ainsi que la variable duale η associée. On peut faire la même remarque pour le SO et l'agent *Loss*.

Algorithme 5.3 Marché endogène avec SO centralisé asynchrone, local à l'agent i

tant que la convergence n'est pas atteinte
  si $t = 0$ **alors**
    Initialiser les échanges locaux $(p_{ij})_{j\in\mathcal{C}_i}$ et variables duales associées $(\lambda_{ij})_{j\in\mathcal{C}_i}$
    Initialiser les puissances locales P_i et Q_i et variables duales associées η_i^p, η_i^q
    $\mathcal{A}_i^0 = \mathcal{C}_i \cup \{SO\}$
  sinon
    Mettre à jour les variables locales via

$$(P_i, Q_i, (p_{ij})_j)^{temp} \leftarrow (5.3) \qquad (5.12)$$
$$p_{ij} \leftarrow p_{ij}^{temp} \qquad j \in \mathcal{A}_i^t \setminus \{SO\}$$

    si $SO \in \mathcal{A}_i^t$ **alors**
      $P_i,\ Q_i \leftarrow P_i^{temp},\ Q_i^{temp}$
    fin si
  fin si
  Envoyer les valeurs mises à jour aux agents de $\mathcal{A}_i^t$
  tant que les valeurs mises à jour ne sont pas reçues de $\mathcal{A}_i^{t+1}$
    Attendre
  fin tant que
  Évaluer les variables duales via

$$\lambda_{ij} \leftarrow (5.6) \qquad j \in \mathcal{A}_i^{t+1} \setminus \{SO\}$$

  si $SO \in \mathcal{A}_i^{t+1}$ **alors**
    $\eta_i^p,\ \eta_i^q \leftarrow (5.7),\ (5.8)$
  fin si
  $t \leftarrow t + 1$
fin tant que

Algorithme 5.4 Marché endogène avec SO centralisé asynchrone, local à l'agent SO

tant que la convergence n'est pas atteinte
  si $t = 0$ **alors**
    Initialiser les puissances actives $(P_i^{SO})_{i\in\mathcal{C}}$ et les variables duales associées $(\eta_i^p)_{i\in\mathcal{C}}$
    Initialiser les puissances réactives $(Q_i^{SO})_{i\in\mathcal{C}}$ et les variables duales associées $(\eta_i^q)_{i\in\mathcal{C}}$
    Initialiser les puissances P_{Loss}^{SO}, Q_{Loss}^{SO} et les variables duales associées η_{Loss}^p, η_{Loss}^q
    $\mathcal{A}_{SO}^0 = \mathcal{C}^+$
  sinon
    Mettre à jour les variables locales via

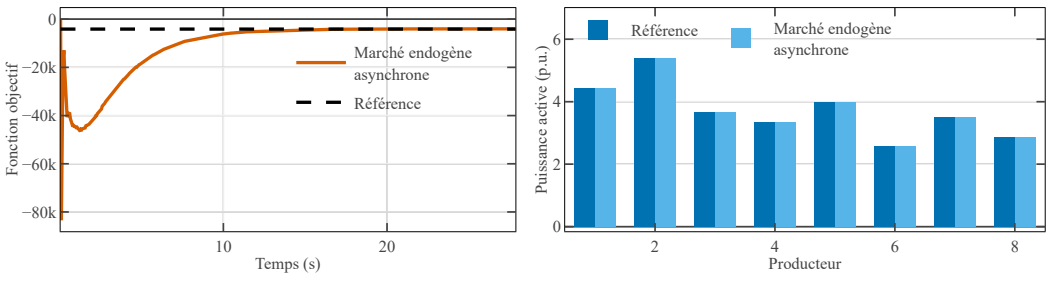
$$(\mathbf{P}_{SO}, \mathbf{Q}_{SO})^{temp} \leftarrow (5.4) \qquad (5.13)$$
$$P_i^{SO}, Q_i^{SO} \leftarrow P_i^{temp}, Q_i^{temp} \quad i \in \mathcal{A}_{SO}^t \setminus \{Loss\}$$

    si $Loss \in \mathcal{A}_{SO}^t$ **alors**
      $P_{Loss}^{SO}, Q_{Loss}^{SO} \leftarrow (5.5)$
    fin si
  fin si
  Envoyer les valeurs mises à jour aux agents de $\mathcal{A}_{SO}^t$
  tant que les valeurs mises à jour ne sont pas reçues de $\mathcal{A}_{SO}^{t+1}$
    Attendre
  fin tant que
  Évaluer les variables duales via

$$\eta_i^p, \eta_i^q \leftarrow (5.7), (5.8) \qquad i \in \mathcal{A}_{SO}^{t+1}$$

  $t \leftarrow t + 1$
fin tant que

5.2.3 Résultats de l'implémentation asynchrone



(a) (b)

FIGURE 5.4 – Résultat de l'algorithme asynchrone de marché pair à pair endogène appliqué au cas test à 31 agents et comparaison par rapport à la solution optimale d'un OPF centralisé synchrone. Les paramètres d'asynchronisme sont réglés ici à $\delta_{pair} = 10\%$ et $\delta_{SO} = 10\%$. En Figure 5.4a, la valeur de la fonction objectif globale est affichée au cours du temps. En Figure 5.4b, les puissances actives de 9 producteurs au bout de 27 secondes de calculs sont comparées à la solution optimale.

Tout d'abord, comparons les solutions du problème endogène avec la solution de référence donnée par l'AC OPF de Matpower dans la Figure 5.3 et la Figure 5.4. Nous pouvons observer que les solutions endogènes, qu'elles soient synchrones ou asynchrones, sont similaires en ce qui concerne les injections de puissance active. La même remarque peut être faite pour les injections de puissance réactive et pour toute valeur de δ_{pair} et δ_{SO}. La différence qui peut être remarquée entre les solutions est due aux diverses erreurs de précision qui sont reportées sur les quelques centaines d'itérations pour chaque agent local.

Notre étude porte principalement sur les variations du temps de convergence de l'algorithme et du nombre d'itérations par rapport à l'implémentation asynchrone. Une seule valeur ${\epsilon^{p,tol}}^2 = 10^{-3}$ est ainsi choisie pour le critère d'arrêt sur le résidu primal. Des études similaires peuvent être effectuées avec des valeurs différentes ou un critère de résidu dual.

Le paramètre d'asynchronisme des pairs δ_{pair} est défini en (5.10) comme la proportion minimale de messages qu'un pair doit recevoir avant de commencer les calculs d'itération locaux. Pour des raisons d'interprétabilité des résultats, tous les pairs sont fixés à la même valeur de δ_{pair} bien qu'ils puissent différer dans une implémentation réelle. De même, le paramètre asynchrone du SO δ_{SO} est défini en (5.11) comme la proportion minimale de messages que le SO doit recevoir avant d'effectuer les calculs d'itération locaux. Le cas $\delta_{pair} = \delta_{SO} = 1$ correspond au cas synchrone, car chaque agent et le SO doivent attendre le nombre de messages maximum possible.

Le Tableau 5.1 compile les différents paramètres des modèles de délais de communication et de calculs utilisés dans les études qui suivent. Le cas d'étude utilisé est le cas à 31 agents, 39 nœuds et 46 lignes utilisé plus tôt dans le chapitre. Les problèmes d'optimisation locaux des agents du marché sont résolus via le solveur `OSQP` [98] basé sur la méthode ADMM tandis que le problème d'OPF modifié du SO est résolu via le package Julia `PowerModels` [106], qui se base sur le solveur `Ipopt` [108], utilisant une méthode de points intérieurs.

Section	**Cas**	Δ_{comm}	Δ_{calc}
5.2.4	31 agents 39 nœuds $\rho = 1$	$\alpha = 90.0$ $\beta = 10$ ms Déterministe $p_{pertes} = 0\%$	$\Delta_{calc,SO} = 0$ $\Delta_{calc,pair} = 0$
5.2.5	31 agents 39 nœuds $\rho = 1$	$\alpha = 90.0$ $\beta = 10$ ms Déterministe $p_{pertes} = 0\%$	$\Delta_{calc,SO} > 0$ constant $\Delta_{calc,pair} = 0$

Tableau 5.1 – Paramètres utilisés pour les simulations de l'algorithme de marché pair à pair.

Comme pour les chapitres précédents, on commence par étudier l'influence des paramètres d'asynchronisme sur le temps de convergence de l'algorithme en section 5.2.4. On effectue ensuite en section 5.2.5 une étude de l'influence du temps de calcul du SO, en le comparant notamment au délai de communication moyen.

5.2.4 Influence des paramètres d'asynchronisme

La Figure 5.5 montre le temps de convergence normalisé de l'algorithme asynchrone pour différentes valeurs de δ_{pair} et δ_{SO}. Ici, nous ne prenons en compte que les délais de communication, ce qui signifie que tous les temps de calcul sont fixés à zéro. Le temps de convergence a été normalisé par le délai de communication moyen, qui est égal à 60

ms pour cette étude de cas, et on ne considère pas de perte de message. Nous pouvons observer que le temps de convergence augmente avec le paramètre d'asynchronisme des pairs δ_{pair}, sauf lorsque $\delta_{pair} = 1$. Le temps de convergence augmente également avec le paramètre d'asynchronisme du SO δ_{SO} tant que δ_{pair} n'est pas proche de 1. Cependant, δ_{SO} a un impact plus faible sur le temps de convergence que δ_{pair}. La configuration qui semble minimiser le temps de convergence est celle qui présente les plus petites valeurs de δ_{pair} et δ_{SO}. Cette configuration fait que les agents effectuent un calcul à chaque fois qu'un message est reçu, ce qui augmente considérablement le nombre de calculs locaux des pairs, et de calculs du SO, que nous définirons comme k^*_{SO}.

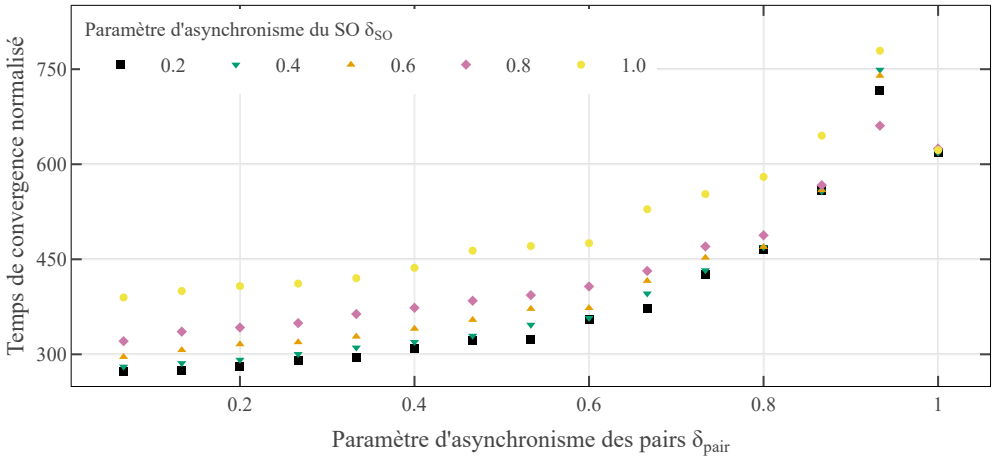


FIGURE 5.5 – Temps de convergence normalisé par le délai de communication moyen en fonction du paramètre d'asynchronisme des pairs δ_{pair} pour différentes valeurs du paramètre d'asynchronisme de l'opérateur système SO δ_{SO}.

Comme nous pouvons l'observer dans le Tableau 5.2, le nombre moyen de calculs k^*_{SO} augmente lorsque le paramètre asynchrone du SO δ_{SO} diminue. Le nombre de calculs du SO k^*_{SO} est multiplié par cinq pour $\delta_{SO} = 20\%$ par rapport au cas synchrone. Cette augmentation importante entraîne une grande charge de travail de calcul pour le SO, ce qui signifie que son temps de traitement doit être pris en compte dans toute la suite de l'étude.

δ_{SO}	1.0	0.6	0.2
$\mathbb{E}[k^*_{SO}]$	325	544	1603
$\sigma[k^*_{SO}]$	1.6	14.6	82.9

TABLEAU 5.2 – Moyenne et écart-type du nombre de calculs du SO k^*_{SO} par rapport au paramètre d'asynchronisme du SO δ_{SO}.

5.2.5 Influence du temps de calcul du SO

On représente dans le Tableau 5.3 les temps de calcul réels des différents problèmes locaux : ceux des pairs, producteurs et consommateurs, et ceux du SO. On a séparé les producteurs et les consommateurs étant donné leurs nombres de voisins respectifs. En effet,

les producteurs sont voisins avec tous les consommateurs et l'agent *Loss*, et vice-versa, ce qui modifie leurs temps de calcul.

	PowerModels (AC-OPF)	**OSQP (Pair)**	
		producteur	***consommateur***
$\mathbb{E}$[temps de calcul]	75 ms	77 μs	58 μs
$\boldsymbol{\sigma}$(temps de calcul)	15 ms	22 μs	18 μs

TABLEAU 5.3 – Moyenne et écart type des temps de calcul effectués en Julia, sur un processeur i7-8850H

Dans cette section, nous ajoutons différentes valeurs de temps de calcul pour l'opérateur système SO. Sur la base des données des tableaux 5.2 et 5.3, le SO n'aura pas assez de temps pour effectuer le calcul d'un OPF complet entre chaque message reçu si son temps de calcul a le même ordre de grandeur que les délais de communication. Ainsi, le nombre réel de calculs du SO sera probablement inférieur à celui du tableau 5.2 et la simulation aura failli à estimer un temps de convergence réaliste de l'algorithme étudié.

En ce qui concerne les temps de calcul des pairs, la résolution du problème local par les pairs du marché est 1000 fois plus rapide que l'OPF AC réalisée par le SO, selon le tableau 5.3. Seuls les temps de calcul du SO seront donc pris en compte par la suite. Les résultats suivants sont valables tant que les temps de calcul de chaque pair du marché sont négligeables par rapport à ceux du SO. Cette hypothèse devrait être facilement validée même si les pairs ont des capacités de calcul modestes, compte tenu de la différence de complexité entre les deux problèmes. Notons c_{SO} le rapport calcul sur communication défini en (5.14).

$$c_{SO} = \frac{\text{temps de calcul du SO}}{\text{délai de communication moyen}} \tag{5.14}$$

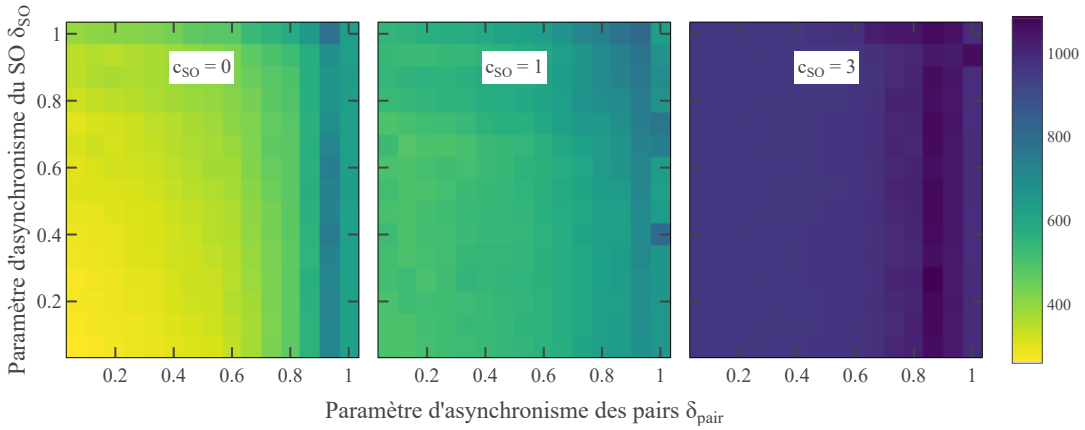


FIGURE 5.6 – Temps de convergence normalisé en fonction des paramètres d'asynchronisme des pairs δ_{pair} et de l'opérateur système SO δ_{SO}, pour des valeurs différentes du ratio calcul/communication c_{SO}.

La Figure 5.6 représente le temps de convergence normalisé en fonction des deux para-

mètres asynchrones δ_{pair} et δ_{SO}, et pour trois valeurs du rapport c_{SO}. La première figure, pour laquelle $c_{SO} = 0$, représente le même cas d'étude que dans la section précédente où il n'y avait pas de temps de calcul, à titre de comparaison. Les deuxième et troisième figures montrent les cas $c_{SO} = 1$ et $c_{SO} = 3$ respectivement. Nous pouvons observer que le temps de convergence est toujours le plus faible pour les faibles valeurs de δ_{pair} et δ_{SO}. Le paramètre d'asynchronisme des pairs δ_{pair} a un impact plus important sur le temps de convergence que le paramètre d'asynchronisme des SO δ_{SO}, quelle que soit la valeur de c_{SO}.

Naturellement, le temps de convergence augmente avec le temps de calcul du SO, étant donné qu'une partie de ce temps de convergence est proportionnel au temps de calcul. Cependant, lorsque le temps de calcul est supérieur au délai moyen de communication (par exemple, $c_{SO} = 3$), les deux paramètres asynchrones n'ont pas un grand impact sur le temps de convergence. Cela est dû au fait que presque tous les messages arrivent au SO pendant son calcul, de sorte que l'itération suivante soit équivalente à une mise à jour presque synchrone, quelle que soit la valeur de δ_{SO}.

Ce phénomène est également visible dans la Figure 5.7, qui montre l'influence du rapport calcul/communication c_{SO} sur le temps de convergence et le nombre total d'itérations SO k^*_{SO}, pour une valeur fixée du paramètre d'asynchronisme des pairs de $\delta_{pair} = 20\%$. Nous pouvons voir que lorsque $c_{SO} < 1$, le paramètre d'asynchronisme du SO δ_{SO} a bien un effet sur le temps de convergence de l'algorithme, en contrepartie d'une augmentation significative du nombre de calculs SO k^*_{SO}. Cependant, au fur et à mesure que c_{SO} augmente, le temps de calcul du SO devient de plus en plus grand, ce qui limite la quantité de calculs effectués et la réduction du temps entre $\delta_{SO} = 20\%$ et $\delta_{SO} = 100\%$. Le nombre d'itérations du SO diminue vers la valeur à $\delta_{SO} = 100\%$ à mesure que le temps de calcul augmente.

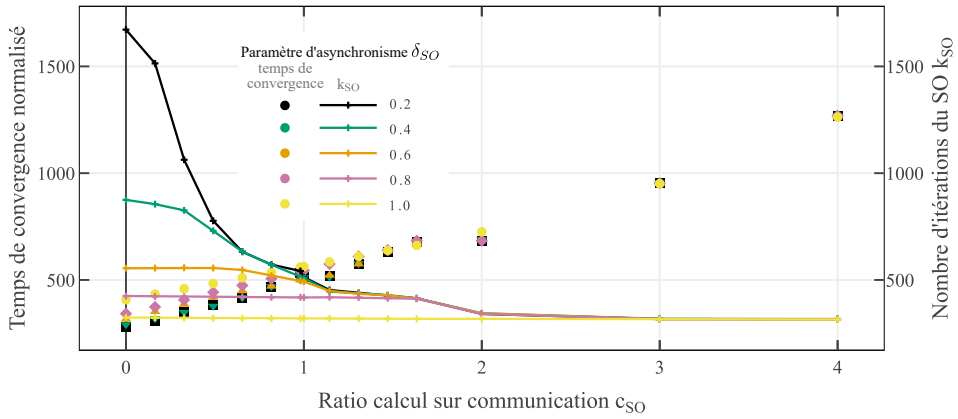


FIGURE 5.7 – Temps de convergence et nombre d'itérations du SO en fonction du ratio calcul sur communication c_{SO}, pour différentes valeurs du paramètre d'asynchronisme du SO δ_{SO} et pour une valeur fixe $\delta_{pair} = 20\%$. Plus le temps de calcul du SO augmente, plus les différentes versions asynchrones sont équivalentes en termes de temps de convergence et de messages échangés. En effet, tous les messages ont le temps d'arriver à destination lors de la phase de calcul du SO, ce qui revient finalement à un fonctionnement quasi-synchrone de l'algorithme.

5.2.6 Conclusions sur l'algorithme de marché pair à pair endogène avec opérateur système centralisé

L'implémentation asynchrone d'un marché de pair à pair endogène a été étudiée sur un cas test de 31 agents. Nous avons vérifié que la solution du problème asynchrone était la même que celle d'une OPF. Nous avons montré que le paramètre asynchrone des pairs δ_{pair} avait un impact sur le temps de convergence de l'algorithme, alors que le paramètre asynchrone du SO δ_{SO} avait un impact sur le nombre de calculs effectués par le SO pour atteindre la convergence. Cependant, lorsque le temps de calcul du SO augmente, le nombre de calculs qu'il peut effectuer diminue et toutes les valeurs de δ_{SO} conduisent à la même performance. L'algorithme proposé nécessite encore une validation supplémentaire, en particulier une mise en œuvre sur un système électrique expérimental. De plus, avec l'augmentation du nombre d'agents participants, le temps de calcul du SO est amené à augmenter également. Il est donc nécessaire d'envisager de décentraliser l'entité "opérateur système".

5.3 Résolution décentralisée du marché avec SO décentralisé

L'inconvénient de l'algorithme endogène avec SO centralisé réside en ce qu'un seul agent est chargé de vérifier les contraintes réseaux, en réalisant un OPF modifié sur l'intégralité du réseau. La complexité du problème de l'OPF ne permet pas de passer à l'échelle dans le cas d'un réseau électrique étendu. De plus, la centralisation des échanges entre le SO et tous les agents du marché implique naturellement un point de défaillance unique, c'est-à-dire que si l'agent opérateur système venait à défaillir, alors l'intégralité de la résolution en sera perturbée.

Il devient alors intéressant de décentraliser l'opérateur système en plusieurs agents, chacun s'occupant des nœuds d'une région donnée. La décentralisation du problème d'OPF a été approfondie dans le Chapitre 4 et peut se transposer dans le cadre de notre problème endogène. Les agents du marché échangent de la même manière entre eux des volumes d'électricité, mais au lieu de communiquer avec un unique opérateur système, ils communiquent chacun avec l'opérateur système local qui leur est associé, comme illustré en Figure 5.8. Les différents opérateurs système locaux communiquent alors entre eux pour atteindre un consensus sur les valeurs des tensions partagées, dans le but de résoudre l'OPF modifié de manière décentralisée et en parallèle du marché de l'électricité. Enfin, l'agent *Loss* communique avec l'ensemble des opérateurs système locaux pour déterminer les pertes totales du réseau et les racheter aux producteurs.

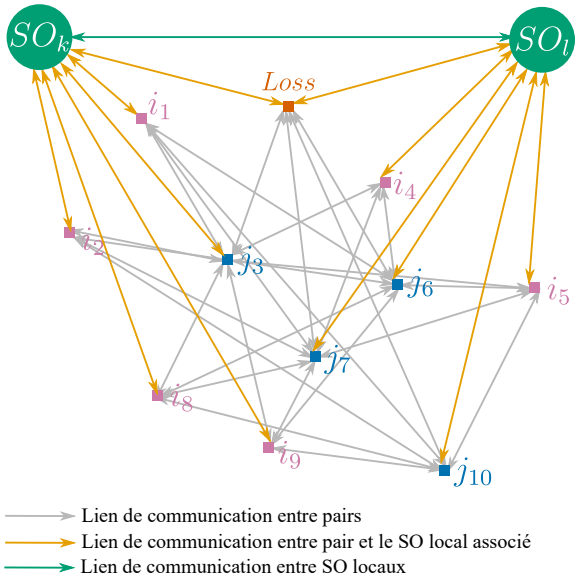


FIGURE 5.8 – Schéma exemple d'un réseau dont la résolution de marché est effectuée de pair à pair, et où le réseau électrique est décomposé en deux régions k et l. À chaque itération de la résolution, les opérateurs réseau locaux vérifient les contraintes physiques de la solution et influent sur le résultat. Les flèches grises représentent les liens de communication entre les agents du marché. Les flèches oranges représentent les liens de communication entre le SO et les agents. Les flèches vertes représentent les liens de communication entre les SO locaux. Les agents en rose notés i sont des consommateurs, les agents en bleu notés j sont des producteurs, et l'agent *Loss* en orange rachète les pertes du réseau aux producteurs.

Nota On aurait aussi pu considérer le cas où plusieurs agents sont en charge de racheter les pertes du réseau, par exemple un agent de pertes par région qui rachète aux producteurs de la région considérée.

5.3.1 Version synchrone

On s'intéresse toujours au problème global endogène présenté en (5.1), mais au lieu de considérer les contraintes physiques dans une unique fonction ζ_{AC}, on découpe cette fonction en plusieurs fonctions propres à chaque région k considérée $\zeta_{AC,k}$ et on note SOk l'opérateur système local à la région k. Cette nouvelle fonction locale prend en compte les contraintes de tension des nœuds internes à la région, ainsi que les contraintes des lignes dont au moins une extrémité appartient à la région. De la même manière que lorsqu'on décentralise un problème d'OPF, il est nécessaire de créer des copies locales des tensions limitrophes à la région pour pouvoir prendre en compte les contraintes sur les lignes sortantes. Ainsi, en parallèle du consensus sur les puissances échangées p_{ij}, de celui sur les puissances entre les agents du marché et l'opérateur système p_i, on rajoute un consensus sur les tensions limitrophes $v_{n,l}^{SOk}$ entre les opérateurs systèmes locaux voisins. Les consensus sont réalisés à l'aide de la méthode ADMM.

Nota Pour ne pas alourdir les notations, les tensions considérées sont des tensions complexes mais elles sont implémentées en phase et en amplitude dans le code.

Formulation du marché endogène avec SO décomposé en régions

$$\min \sum_{k=1}^{R} \left(\sum_{i \in \mathcal{C}_k} f_i(P_i, Q_i) + \zeta_{AC,k}(P, Q) \right) \tag{5.15a}$$

$$\begin{aligned}
&\text{selon } S_i = P_i + jQ_i,\ S_i^{SO_k} = P_i^{SO_k} + jQ_i^{SO_k} && k \in [\![1, R]\!],\ i \in \mathcal{C}_k^+ \\
&\quad p_{ij} && i, j \in \mathcal{C}^+ \\
&\quad v_{n,l}^{SO_k} && k \in [\![1, R]\!],\ n \in \mathcal{O}_k,\ l = (SO_k, SO_{k'}) \in \mathcal{L}_n \\
&\text{tel que } p_{ij} = -p_{ji} && i, j \in \mathcal{C}^+ && (5.15b) \\
&\quad v_{n,l}^{SO_k} = v_{n,l}^{SO_{k'}} && k \in [\![1, R]\!],\ n \in \mathcal{O}_k,\ l = (SO_k, SO_{k'}) \in \mathcal{L}_n && (5.15c) \\
&\quad S_i = S_i^{SO_k} && k \in [\![1, R]\!],\ i \in \mathcal{C}_k^+ && (5.15d) \\
&\quad P_i = \sum_{j \in \mathcal{C}_i} p_{ij} && i \in \mathcal{C} && (5.15e) \\
&\quad \underline{S_i} \leq S_i \leq \overline{S_i} && i \in \mathcal{C} && (5.15f) \\
&\quad p_{Loss} = -\sum_{i \in \mathcal{C}} p_i && && (5.15g)
\end{aligned}$$

Le Tableau 5.4 présente les contraintes des différents consensus ainsi que leurs variables duales associées.

Contraintes de consensus	**Variables duales**
$p_{ij} = -p_{ji}$	λ_{ij}
$P_i = P_i^{SOk}$	η_i^p
$Q_i = Q_i^{SOk}$	η_i^q
$v_{n,l}^{SO_k} = v_{n,l}^{SO_{k'}}$	$\gamma_{n,l}$

TABLEAU 5.4 – Contraintes ajoutées au problème de marché pair à pair endogène avec SO décentralisé pour pouvoir décentraliser la résolution parmi les différents agents du marché et les opérateurs système locaux.

On note que le consensus entre les opérateurs système locaux se fait sur une tension de type $v_{n,l}$ associée au nœud n et à la frontière l, comme présenté en section 4.5 du chapitre précédent. On pourrait aussi faire le choix de décomposer au niveau des tensions de chaque nœud limitrophe comme étudié dans [54] et en section 4.4, ou bien de réaliser des consensus sur la somme et la différence des tensions aux bornes de chaque ligne sortante d'une région comme présenté en [79, 109].

Les équations (5.16) et (5.17) représentent les étapes de mise à jour respectivement d'un pair i et d'un opérateur système local à la région k. À noter que (5.16) est équivalent au cas avec SO centralisé (5.3), en remplaçant seulement l'exposant SO par SO_k où k désigne la région dont fait partie l'agent i. Pour ce qui est de la mise à jour de l'opérateur système local SO_k en (5.17), elle reprend les termes du cas avec SO centralisé (5.4), où l'on ajoute les termes de consensus sur les tensions frontalières.

Le puissance active de l'agent *Loss* est calculée grâce aux équations en (5.18) et (5.19). Dans un premier temps, on ne prend pas en compte la puissance réactive de pertes pour simplifier les équations. La puissance active totale de pertes (5.19) est calculée à partir des puissances perdues dans chaque région du réseau.

Les variables duales sont mises à jour avec (5.20), (5.21), (5.22) et (5.23).

$$(P_i, Q_i, (p_{ij})_j)^{t+1} = \arg\min_{P_i, Q_i, (p_{ij})_j} f_i(P_i, Q_i) + \sum_{j \in \mathcal{C}_i} \sigma^\rho \left(p_{ij}, \frac{p_{ij}^t - p_{ji}^t}{2}, \lambda_{ij}^t \right) + \sigma^\rho \left(P_i, \frac{P_i^{SO_k,t} + P_i^t}{2}, \eta_i^{p,t} \right) + \sigma^\rho \left(Q_i, \frac{Q_i^{SO_k,t} + Q_i^t}{2}, \eta_i^{q,t} \right) \tag{5.16a}$$

$$\text{tel que } P_i = \sum_{j \in \mathcal{C}_i} p_{ij} \tag{5.16b}$$

$$\underline{P_i} \leq P_i \leq \overline{P_i} \tag{5.16c}$$

$$\underline{Q_i} \leq Q_i \leq \overline{Q_i} \tag{5.16d}$$

$$
\begin{aligned}
(\mathbf{P}^{SO_k}, \mathbf{Q}^{SO_k}, \mathbf{v}^{SO_k})^{t+1} = \arg \min_{\substack{\mathbf{P}^{SO_k}=(P_i^{SO_k})_i \\ \mathbf{Q}^{SO_k}=(Q_i^{SO_k})_i \\ \mathbf{v}^{SO_k}=(v_n^{SO_k})_n}} & \zeta_{AC,k}(\mathbf{P}^{SO_k}, \mathbf{Q}^{SO_k}, \mathbf{v}^{SO_k}) \\
& + \sum_{i \in \mathcal{C}_k} \sigma^\rho \left(\frac{P_i^{SO_k,t} + P_i^t}{2}, P_i^{SO_k}, \eta_i^{p,t} \right) + \sum_{i \in \mathcal{C}_k} \sigma^\rho \left(\frac{Q_i^{SO_k,t} + Q_i^t}{2}, Q_i^{SO_k}, \eta_i^{q,t} \right) \\
& + \sum_{i \in \mathcal{C}_k} \sigma^\rho \left(P_i^{SO_k,t} - \frac{1}{|\mathcal{C}_k|} \frac{P_{Loss}^{SO_k,t} + P_{Loss}^t}{2}, P_i^{SO_k} - \frac{1}{|\mathcal{C}_k|} P_{Loss}^{SO_k,t}, \frac{1}{|\mathcal{C}_k|} \eta_{Loss}^{p,t} \right) \\
& + \sum_{n \in \mathcal{O}_k} \sum_{\substack{l \in \mathcal{L}_n \\ l=(SO_k, SO_{k'})}} \sigma^\rho \left(\frac{v_{n,l}^{SO_k,t} + v_{n,l}^{SO_{k'},t}}{2}, v_n^{SO_k}, \gamma_{n,l}^{SO_k,t} \right)
\end{aligned} \tag{5.17}
$$

$$
P_{Loss,k}^{SO_k,t+1} = - \sum_{i \in \mathcal{C}_k} P_i^{SO_k,t+1} \qquad k \in [\![1, R]\!] \tag{5.18}
$$

$$
P_{Loss}^{t+1} = \sum_{k=1}^{R} P_{Loss,k}^{SO_k,t+1} \tag{5.19}
$$

$$
\lambda_{ij}^{t+1} = \lambda_{ij}^t - \rho \frac{p_{ij}^{t+1} + p_{ji}^{t+1}}{2} \qquad i \in \mathcal{C}^+,\ j \in \mathcal{C}_i \tag{5.20}
$$

$$
\eta_i^{p,t+1} = \eta_i^{p,t} + \rho \frac{P_i^{SO_k,t+1} - P_i^{t+1}}{2} \qquad k \in [\![1, R]\!],\ i \in \mathcal{C}_k^+ \tag{5.21}
$$

$$
\eta_i^{q,t+1} = \eta_i^{q,t} + \rho \frac{Q_i^{SO_k,t+1} - Q_i^{t+1}}{2} \qquad k \in [\![1, R]\!],\ i \in \mathcal{C}_k^+ \tag{5.22}
$$

$$
\gamma_{n,l}^{SO_k,t+1} = \gamma_{n,l}^{SO_k,t} + \rho \frac{v_{n,l}^{SO_k,t+1} - v_{n,l}^{SO_{k'},t+1}}{2} \qquad \begin{array}{r} k \in [\![1, R]\!],\ n \in \mathcal{O}_k, \\ l = (SO_k, SO_{k'}) \in \mathcal{L}_n \end{array} \tag{5.23}
$$

Le comportement d'un agent du marché quelconque ne diffère pas de celui décrit dans l'Algorithme 5.1 dans le cas où le SO était centralisé, en considérant que les messages précédemment envoyés à l'unique opérateur système sont maintenant envoyés à l'opérateur système local auquel l'agent est associé.

Pour ce qui est de l'algorithme décrivant les opérateurs locaux, il est donné en Algorithme 5.5, et celui décrivant l'agent *Loss* est donné en Algorithme 5.6. L'opérateur système local fonctionne en partie comme l'opérateur système central de l'Algorithme 5.2, mais doit aussi prendre en compte le consensus des tensions avec les régions voisines : des termes liés à ces consensus sont rajoutés à sa fonction coût locale et des communications avec les opérateurs voisins sont aussi rajoutées. Le comportement de l'agent *Loss* est légèrement différent du cas avec SO centralisé étant donné qu'il doit déduire l'ensemble des pertes sur le réseau à partir des pertes déterminées par chaque opérateur local : $p_{Loss} = \sum_{k=1}^{R} p_{Loss,k}^{SO_k}$.

Algorithme 5.5 Marché endogène avec SO décentralisé synchrone, algorithme d'un opérateur système local SO_k

tant que la convergence n'est pas atteinte
  si $t = 0$ **alors**
    Initialiser les puissances actives $(P_i^{SO_k})_{i\in\mathcal{C}_k}$ et les variables duales associées $(\eta_i^p)_{i\in\mathcal{C}_k}$
    Initialiser les puissances réactives $(Q_i^{SO_k})_{i\in\mathcal{C}_k}$ et les variables duales associées $(\eta_i^q)_{i\in\mathcal{C}_k}$

    Initialiser la puissance $P_{Loss,k}^{SO_k}$ et la variable duale associée $\eta_{Loss,k}^p$
    Initialiser les tensions complexes $v_{n,l}^{SO_k}$ et les variables duales associées $\gamma_{n,l}^{SO_k}$
  sinon
    Mettre à jour les variables locales via
    $\mathbf{P}^{SO_k}, \mathbf{Q}^{SO_k}, \mathbf{v}^{SO_k} \leftarrow$ (5.17)
    $P_{Loss,k}^{SO_k} \leftarrow$ (5.18)
  fin si
  Envoyer les valeurs $P_i^{SO_k}$, $Q_i^{SO_k}$ mises à jour aux agents $i \in \mathcal{C}_k$
  Envoyer la valeur $P_{Loss,k}^{SO_k}$ à l'agent $Loss$
  Envoyer les valeurs $v_{n,l}^{SO_k}$ aux opérateurs voisins tels que $n \in \mathcal{O}_k$, $l \in \mathcal{L}_n$
  tant que les P_i, Q_i ne sont pas reçus des agents $i \in \mathcal{C}_k$, $P_{Loss,k}$ de l'agent $Loss$, et $v_{n,l}^{SO_{k'}}$ des opérateurs voisins
    Attendre
  fin tant que
  Évaluer les variables duales via

$$\eta_i^p, \leftarrow (5.21) \qquad i \in \mathcal{C}_k^+$$
$$\eta_i^q \leftarrow (5.22) \qquad i \in \mathcal{C}_k$$
$$\gamma_{n,l}^{SO_k} \leftarrow (5.23) \qquad n \in \mathcal{O}_k,\ l = (SO_k, SO_{k'}) \in \mathcal{L}_n$$

  $t \leftarrow t + 1$
fin tant que

Algorithme 5.6 Marché endogène avec SO décentralisé synchrone, algorithme local à l'agent $i = Loss$

tant que la convergence n'est pas atteinte
  si $t = 0$ **alors**
    Initialiser les échanges locaux $(p_{ij})_{j\in\mathcal{C}_i}$ et variables duales associées $(\lambda_{ij})_{j\in\mathcal{C}_i}$
    Initialiser les puissances $(P_{Loss,k})_{k\in[\![1,\,R]\!]}$ et variables duales associées $(\eta^p_{Loss,k})_{k\in[\![1,\,R]\!]}$
  sinon
    Mettre à jour les variables locales via

$$((P_{Loss,k})_k, (p_{ij})_j)^{t+1} = \arg\min \sum_{j\in\mathcal{C}_i} \sigma^\rho\left(p_{ij}, \frac{p^t_{ij} - p^t_{ji}}{2}, \lambda^t_{ij}\right) + \sum_{k=1}^{R} \sigma^\rho\left(P_{Loss,k}, \frac{P^{SO_k,t}_{Loss,k} + P^t_{Loss,k}}{2}, \eta^{p,t}_{Loss,k}\right) \tag{5.24a}$$

$$\text{tel que } \sum_{k=1}^{R} P_{Loss,k} = \sum_{j\in\mathcal{C}_i} p_{ij} \tag{5.24b}$$

  fin si
  Envoyer les valeurs p_{ij} mises à jour aux agents $j \in \mathcal{C}_i$
  Envoyer la valeur $P_{Loss,k}$ mise à jour aux opérateurs système SO_k associés
  tant que les p_{ji} ne sont pas reçus de $j \in \mathcal{C}_i$ et $P^{SO_k}_{Loss,k}$ des SO_k
    Attendre
  fin tant que
  Évaluer les variables duales via
    $\lambda_{ij} \leftarrow$ (5.20) $\qquad j \in \mathcal{C}_i$
    $\eta^p_{Loss,k} \leftarrow$ (5.21) $\qquad k \in [\![1,\,R]\!]$
  $t \leftarrow t + 1$
fin tant que

En ce qui concerne la communication, on considère maintenant trois types d'échanges, illustrés en Figure 5.9.

— Les échanges entre agents du marché : pour déterminer les volumes d'électricité échangés de pair à pair p_{ij}.
— Les échanges entre un agent du marché et son opérateur système associé : pour se mettre d'accord sur la puissance produite ou consommée par l'agent du marché (qui peut être l'agent *Loss*).
— Les échanges entre opérateurs système locaux : pour se mettre d'accord sur la valeur des tensions limitrophes.

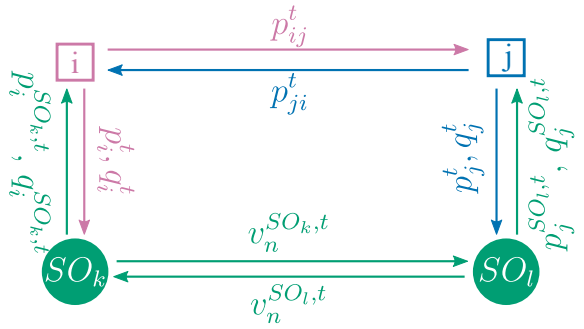
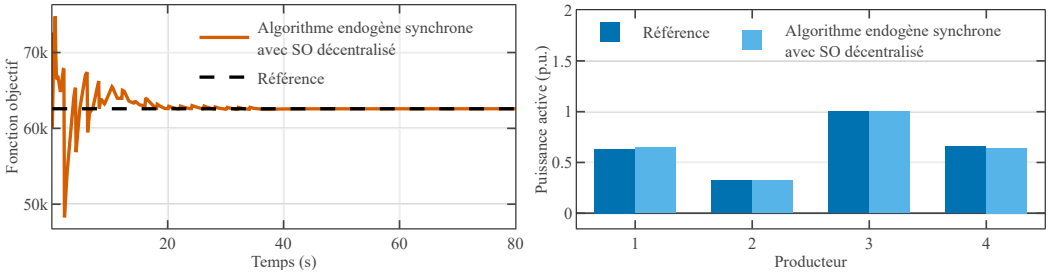


FIGURE 5.9 – Schéma des échanges entre les consomacteurs i et j et les agents opérateur système locaux SO_k et SO_l durant l'itération t.

Ces échanges s'effectuent possiblement via des moyens de communication différents. En effet, les opérateurs système locaux jouant un rôle important et s'occupant de régions fixes, on peut imaginer que leurs échanges se font sur un réseau de communication dédié, par exemple au travers de fibres optiques. En revanche, les agents participant au marché pouvant être de types très différents (des centrales renouvelables au consommateur particulier), il serait intéressant de considérer que leurs échanges passent par un réseau global existant WAN, par exemple Internet. Ainsi, les délais de communication ainsi que les taux de perte de messages seront très différents selon si l'on considère les échanges entre opérateurs ou entre agents du marché.

Les agents du marché cherchent à minimiser leurs propres fonctions coût locales tout en trouvant un consensus sur les échanges de puissance qu'ils réalisent avec les autres agents du marché, et un consensus sur leurs puissances actives et réactives avec le SO local. L'agent *Loss* a pour rôle de s'acquitter financièrement de la surproduction due aux pertes. Pour ce faire, il effectue un consensus sur l'ensemble des pertes du réseau électrique avec les différents SO locaux. Quant aux opérateurs système (SO) locaux, ces derniers cherchent d'une part à atteindre le consensus des puissances actives et réactives avec les agents du marché internes à leurs régions respectives, et d'autre part à atteindre le consensus sur les variables des tensions limitrophes qu'ils partagent avec les autres SO.

On représente en Figure 5.10 la résolution synchrone de l'algorithme pair à pair endogène avec SO décentralisé appliqué à un cas test à 14 nœuds présenté en Annexe A.III et découpé en 2 régions distinctes. On considère des délais fixes de 100 ms par itération. On observe en Figure 5.10a que l'algorithme retrouve la valeur optimale de la fonction objectif, et de petites oscillations qui témoignent d'une convergence non monotone. Les valeurs finales des puissances actives représentées en Figure 5.10b correspondent elles aussi aux valeurs optimales.



(a) (b)

FIGURE 5.10 – Résultat de l'algorithme synchrone de marché pair à pair endogène avec SO décentralisé appliqué au cas test à 14 agents divisé en 2 régions, et comparaison par rapport à la solution optimale. En Figure 5.10a, la valeur de la fonction objectif globale est affichée au cours du temps. En Figure 5.10b, les puissances actives de 4 producteurs sont comparées à la solution optimale.

5.3.2 Version asynchrone

Comme décrit dans la section précédente, de nombreux échanges ont lieu entre les agents du marché, les opérateurs système locaux, et l'agent *Loss*. L'hétérogénéité des délais de communication peut ralentir l'avancement global de l'algorithme, de même lorsque l'on considère les temps de calculs hétérogènes entre les opérateurs système et les agents du marché.

Pour palier ces ralentissements, on introduit alors une version asynchrone de l'algorithme, où les différents agents et opérateurs peuvent continuer leurs calculs malgré le fait qu'ils n'aient pas forcément reçu l'intégralité des messages attendus lors de l'itération précédente. On définit le déclencheur des calculs d'une itération comme suit.

Du point de vue d'un agent i du marché (à l'exception de l'agent *Loss*) : cet agent attend au total $N_{\mathcal{C}_i}$ messages de ses partenaires commerciaux plus 1 message de l'opérateur système local auquel il est associé. Comme dans la section précédente 5.2 avec l'opérateur système SO centralisé, on définit un paramètre d'asynchronisme des pairs δ_{pair} tel que :

$$|\mathcal{A}_i^t| \geq \lceil \delta_{pair} \cdot (|\mathcal{C}_i| + 1) \rceil \tag{5.25}$$

où $\mathcal{A}_i^t \subset \mathcal{C}_i \cup \{SO_k\}$ est l'ensemble des agents dont l'agent i a reçu un message pour l'itération t. Lorsque l'agent i a reçu au moins $\lceil \delta_{pair} \cdot (|\mathcal{C}_i| + 1) \rceil$ messages depuis sa dernière itération, alors l'itération suivante est déclenchée.

Du point de vue de l'agent *Loss*, qui attend les $N_{\mathcal{C}_{Loss}}$ messages de ses partenaires commerciaux et les R messages de chaque opérateur système, plusieurs possibilités sont offertes :

- soit on définit un unique paramètre d'asynchronisme propre à l'agent *Loss* tel que $|\mathcal{A}_{Loss}^t| \geq \lceil \delta_{Loss} \cdot (|\mathcal{C}_{Loss}| + R) \rceil$
- soit on définit deux paramètres d'asynchronisme, c'est-à-dire un quantifiant le nombre minimum de messages reçus de la part des pairs $\delta_{Loss,pairs}$ et un deuxième quantifiant le nombre minimum de messages reçus des opérateurs système $\delta_{Loss,SO}$. Le déclenchement se ferait lorsque les deux conditions se trouvent remplies.

Enfin, du point de vue des opérateurs système SO_k, qui attendent d'une part $N_{\mathcal{C}_k}$ messages de la part de tous les agents internes à leur région, et d'autre part $N_{\mathcal{O}_k}$ messages

provenant des opérateurs voisins, les mêmes possibilités sont offertes :

- soit on définit un unique paramètre d'asynchronisme propre aux opérateurs système tel que $|\mathcal{A}^t_{SO_k}| \geq \lceil \delta_{SO} \cdot (N_{\mathcal{C}_k} + N_{\mathcal{O}_k}) \rceil$
- soit on définit deux paramètres d'asynchronisme, c'est-à-dire un quantifiant le nombre minimum de messages reçus de la part des pairs $\delta_{SO,pairs}$ et un deuxième quantifiant le nombre minimum de messages reçus des opérateurs système $\delta_{SO,SO}$. Le déclenchement se ferait lorsque les deux conditions se trouvent remplies.

À l'heure actuelle, nous n'avons pas encore effectué l'implémentation asynchrone ni l'étude de l'algorithme endogène asynchrone avec SO décentralisé. Ce travail, dans la continuité aux études proposées dans ce manuscrit, aurait pu inclure des études portant sur plusieurs aspects.

- L'influence de l'hétérogénéité des temps de calculs et de communication sur le temps de convergence global de l'algorithme : est-ce que le fait que les calculs des opérateurs soient très longs comparés aux calculs des pairs impacte l'avancement de l'algorithme, et en quoi la version asynchrone pallie à ce problème. On peut se poser la même question en ce qui concerne les délais de communication qui peuvent varier si l'on considère des liaisons rapides et fiables entre opérateurs versus des liaisons lentes et peu fiables entre les agents.
- Une étude similaire à la section 5.2.5 pourrait montrer les limites de l'asynchronisme lorsque le ratio entre les temps de calcul et les délais de communication dépasse un certain seuil. On pourrait déterminer lesquels des délais entre agents, entre opérateurs ou entre agents et opérateurs sont cruciaux dans le cas endogène avec SO décentralisé.
- Une étude de l'influence des différents paramètres d'asynchronisme sur le nombre de messages échangés serait intéressante. Notamment pour vérifier si le changement d'un paramètre d'asynchronisme influerait sur le nombre de messages échangés sur un canal annexe, par exemple si la variation de $\delta_{SO,SO}$ influe sur le nombre d'échanges entre les pairs.

5.4 Conclusions et perspectives du chapitre

Le marché de l'électricité pair à pair endogène est une extension du marché pair à pair qui prend en compte les contraintes physiques du réseau électrique. La décomposition du problème endogène permet aux acteurs économiques du marché, les pairs, de fonctionner séparément du gestionnaire de réseau qui s'assure que les contraintes sont respectées, et malgré cela de tout de même obtenir une solution égale à la solution optimale du problème d'Optimal Power Flow.

On propose deux formulations du marché pair à pair endogène : la première avec un opérateur système (SO) unique et centralisé qui s'occupe du réseau électrique en entier, et la seconde avec de multiples opérateurs système qui s'occupent chacun d'une partie du réseau électrique considéré. Ces deux formulations résultent en des algorithmes itératifs qui nécessitent de nombreux échanges d'information à chaque itération. Entre les agents du marché qui communiquent directement les uns avec les autres, les échanges ayant lieu entre agents du marché et opérateur(s) système et les échanges entre opérateurs système dans la seconde formulation, le nombre de messages nécessaires à l'avancement de ces algorithmes est très important. Ainsi, lorsqu'ils sont implémentés dans un contexte réel, le bon déroulement de ces algorithmes dépend fortement de l'état du réseau de communication. De plus, il est essentiel de prendre en compte les temps de calcul des opérateurs système, étant donné qu'ils

sont du même ordre de grandeur que les délais de communication et peuvent donc fortement influencer le comportement des algorithmes.

De la même manière que pour les chapitres précédents, on propose d'étudier les versions asynchrones de ces algorithmes pour pallier les aléas de communication et rendre la résolution robuste à l'état du réseau de communication. Nous avons effectué l'étude de l'algorithme asynchrone dans le cas où l'opérateur système était centralisé, et nous avons pu montrer que l'algorithme asynchrone permet de réduire grandement les temps d'attente entre chaque itération et ainsi réduire le temps de convergence par rapport au cas synchrone. Nous avons aussi étudié l'influence des temps de calculs de l'opérateur système, notamment en comparaison avec le délai de communication moyen. On a pu déterminer que l'accélération du temps de convergence fournie par l'asynchronisme est limitée par le temps de calcul du SO, notamment lorsque ce dernier dépasse deux fois le délai de communication moyen. Ce résultat peut mener au dimensionnement de la machine de calcul ayant la tâche de calculer le problème du SO à chaque itération : il faut une puissance de calcul telle que le temps de calcul ne dépasse pas le délai de communication pour optimiser le temps de convergence global de l'algorithme [2].

En perspectives de ce chapitre, il serait intéressant de mener des études similaires sur l'algorithme asynchrone pair à pair endogène avec SO décentralisé. En effet, le nombre de messages échangés est encore plus important que dans l'algorithme avec SO centralisé, ce qui laisse présager que la version asynchrone puisse diminuer le temps de convergence en réduisant les délais d'attente.

2. Si le réseau à étudier est si grand qu'une telle machine n'existe pas encore, alors la formulation avec SO décentralisé serait peut-être plus appropriée. Chaque région aurait une taille limitée, ce qui permettrait de réduire les temps de calcul des opérateurs système.

Conclusions et perspectives

Bilan des travaux

La thématique des Smart Grids est fondée sur l'interaction d'un réseau électrique et d'un réseau de communication reliant des agents distribués, "intelligents" et communicants. Dans le cadre d'un réseau électrique classique, les événements imprévus – comme la perte d'un équipement, la saturation d'une ligne ou l'erreur de prévision de production d'une centrale – sont principalement amortis et compensés par l'inertie du réseau et ses centrales de réserve ainsi que la gestion coordonnée d'un gestionnaire de réseau unique. Un Smart Grid se propose de réduire cette dépendance à l'aide de stratégies de gestion distribuées intelligentes faisant appel à des échanges d'information et donc à un réseau de communication. Les algorithmes de gestion décentralisés reposent sur la décomposition d'un problème global en plusieurs sous-problèmes locaux résolus séparément et de manière itérative. Ils présentent des avantages comme le passage à l'échelle pour des réseaux de grandes envergures ainsi que la confidentialité des données. Cependant, la décentralisation se paie au prix de nombreuses communications, ce qui rend la résolution de ces algorithmes vulnérables aux aléas du réseau de communication. Les travaux présentés dans ce manuscrit analysent finement l'impact des aléas de communication sur la résilience d'un Smart Grid et présentent des stratégies de gestion prenant en compte ces aléas pour garantir l'opérabilité et l'efficience du Smart Grid. Ces travaux sont notamment centrés sur des versions asynchrones des algorithmes décentralisés, qui réduisent la durée des itérations et présentent également une certaine robustesse face aux aléas de communication.

Le contexte détaillé du réseau électrique d'aujourd'hui et de ses évolutions futures ainsi que le positionnement des travaux est présenté au Chapitre 1. Après avoir définit des modèles de délais de communication paramétriques, déterministes ou stochastiques, nous présentons l'architecture de la plateforme de simulation nous permettant de réaliser différentes analyses et comparaisons au Chapitre 2. Cette plateforme de simulation est implémentée de manière à être flexible et reconfigurable, et se veut open source.

Ensuite, une première étude est menée au Chapitre 3 sur un marché pair à pair utilisant une résolution synchrone, puis asynchrone. Deux types de modèles de délais de communication stochastiques sont analysés, en particulier via l'impact des délais sur le temps de convergence. Le premier modèle gaussien rudimentaire est comparé à un modèle plus élaboré et précis reproduisant la longue queue de distribution observable sur les délais de communication réels. Nous montrons que ces deux modèles sont équivalents lorsque l'on considère l'implémentation asynchrone de l'algorithme, mais qu'ils présentent des disparités lorsque l'on considère la version synchrone. En effet, nous montrons que les variations des délais de communication affectent peu l'algorithme asynchrone en comparaison de l'algorithme

synchrone. De plus, l'algorithme asynchrone présente une accélération de 40% du temps de convergence par rapport au synchrone. Cette immunité de l'implémentation asynchrone vis-à-vis des délais de communication rend envisageable un déploiement opérationnel capable de garantir des délais de convergence d'un marché pair à pair.

Les contraintes physiques du réseau électrique, telles que le plan de tension et les limites de congestion des lignes, ne sont pas prises en compte dans le marché pair à pair. Elles sont alors intégrées dans une deuxième étude portant sur des algorithmes d'Optimal Power Flow au Chapitre 4. Là aussi, l'impact des aléas de communication sur le temps de convergence est étudié, on y montre que l'algorithme asynchrone est robuste aux variations de délais de communication dues aux pertes de messages et dues à l'augmentation des délais de communication par rapport aux temps de calcul. On montre ensuite que le partitionnement du problème influe fortement sur les résolutions synchrones et asynchrones des algorithmes, et que la résolution asynchrone présente une accélération du temps de convergence plus élevée pour un partitionnement plus fin du réseau électrique.

Nous cherchons ensuite au Chapitre 5 à revenir sur la topologie d'échanges pair à pair de la première étude, tout en y incorporant un agent à part entière s'occupant des contraintes physiques du réseau : le marché pair à pair endogène. Ce marché permet de séparer les aspects économiques des aspects physiques du réseau. Des études sont réalisées sur l'impact conjoint des aléas de communication ainsi que des temps de calcul sur le temps de convergence de ce marché. On y constate que l'algorithme asynchrone permet une accélération du temps de convergence. De plus, l'augmentation du temps de calcul à délais de communication constants réduit les effets de l'asynchronisme.

Nous avons pu montrer que, pour les différents algorithmes, la résolution asynchrone permet de réduire le temps de convergence par rapport à la résolution synchrone dans le cas où les délais de communication étaient variés, notamment dans le cas où l'on ajoute des pertes de messages. L'asynchronisme réduit le temps de convergence, mais aussi ses variations. En contrepartie, l'asynchronisme augmente significativement le nombre total de messages échangés sur le réseau de communication.

Perspectives

Parmi les prolongements possibles de ces travaux de thèse, il serait intéressant d'appliquer tous les algorithmes présentés dans ce manuscrit, ainsi que d'autres algorithmes décentralisés asynchrones présents dans la littérature, sur un même cas test dans le but de comparer quantitativement les effets de l'asynchronisme de chacun des algorithmes. La difficulté vient du réglage de certains paramètres, notamment sur l'algorithme OPF, qui influe grandement la vitesse de convergence. De plus, les travaux sur l'algorithme pair à pair endogène avec SO décentralisé présenté dans le Chapitre 5 mériteraient une implémentation asynchrone ainsi qu'une étude des impacts des aléas de communication sur le temps de convergence, comme on a pu le faire pour les autres algorithmes. Cette étude est d'autant plus nécessaire que plusieurs types d'échanges coexistent au sein d'une même résolution, ce qui permet d'analyser l'impact de différents paramètres d'asynchronisme sur la convergence. Pour prolonger ce mouvement, l'implémentation asynchrone d'un marché pair à pair des capacités mériterait des études similaires.

À moyen terme, un déploiement opérationnel à grande échelle des algorithmes asynchrones étudiés incluant un couplage avec une plateforme physique serait nécessaire pour deux raisons. La première est de pouvoir tester les délais de communication et de calculs véritables, notamment si plusieurs agents se trouvent dans des réseaux différents et com-

muniquent via Internet, et de confronter nos modèles aux mesures réelles. La seconde est de pouvoir effectuer une preuve de concept du déploiement réel d'un algorithme de gestion décentralisé, qui descende jusqu'aux échanges physiques d'électricité.

Pour les perspectives plus générales de cette thèse, il s'agirait d'étendre la question de la résilience du réseau non plus seulement aux aléas de communication, mais aussi aux dysfonctionnements matériels, aux aléas algorithmiques dus aux dysfonctionnements logiciels ou cyber-attaques et aux aléas prévisionnels, c'est-à-dire lorsque les acteurs du réseau ne font pas ce qu'ils avaient annoncé. D'autres applications concernant les algorithmes de gestion décentralisés pourraient être explorées. Jusqu'ici, on se concentrait sur des algorithmes de gestion résolus en amont de l'instant de livraison. L'étude de ces mêmes algorithmes appliqués à la gestion temps réel du réseau électrique pourrait se montrer pertinente. Le principe serait d'appliquer les résultats intermédiaires de chaque itération au réseau réel même si l'algorithme n'a pas encore atteint la convergence, et de la même manière que dans l'expérimentation du Chapitre 3, pouvoir modifier les fonctions coût des agents au cours de la résolution. Dans ce contexte d'implémentation en temps réel, l'asynchronisme permettrait de diminuer les temps d'attente dus à la communication. Enfin, l'étude d'un marché hybride entre un marché totalement pair à pair et un marché centralisé, qui serait composé de différentes zones locales pair à pair coordonnées par un gestionnaire centralisé présente aussi un potentiel d'études d'une version asynchrone au vu des différents niveaux de communication.

Annexe

A.I Cas test de marché à 110 agents

Un cas à 110 agents a été créé pour tester nos algorithmes de marché pair à pair synchrones et asynchrones. Il est composé de 30 producteurs et de 80 consommateurs, tous flexibles, avec des fonctions coût et des limites de puissances différentes. La matrice de communication du marché pair à pair est telle que tous les producteurs sont connectés à tous les consommateurs, et vice versa.

La résolution de l'algorithme de marché pair à pair ne nécessite que des informations sur les coûts des agents (leurs fonctions coût) ainsi que leurs limites de production et de consommation. Le problème d'optimisation en (3.1) considère des fonctions coût quadratiques de la forme $a_i \cdot p_i^2 + b_i \cdot p_i$. Les coefficients a_i et b_i de ces fonctions coût ont été générés aléatoirement, ainsi que les limites de puissance $\underline{p_i}$ et $\overline{p_i}$ admises.

Pour qu'une solution puisse exister, on s'assure que la somme des puissances maximales des consommateurs soit inférieure à celle des producteurs. En ce qui concerne les délais de communication entre les agents, ils sont calculés à partir de leurs distances respectives dans le plan, selon le modèle décrit en section 2.1 du Chapitre 2. Les positions en x et en y des agents sont tirées aléatoirement dans l'intervalle $[0, 2]$. Les valeurs de tous les coefficients nécessaires au calcul sont compilées dans le Tableau I.1.

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

Tableau I.1 – Valeurs numeriques des coefficients du marché test à 110 agents.

A.II Cas test à 118 nœuds pour calcul d'Optimal Power Flow

On est amené tout au long des Chapitres 3 et 4 à appliquer les différents algorithmes sur le cas test IEEE 118 nœuds [3] mis à disposition par le logiciel `matpower`. Ce cas test compile les informations de 118 nœuds, 53 générateurs et 186 lignes nécessaires à la résolution d'un OPF.

A.II.1 Cas agrémenté de données temporelles de consommation

La comparaison des résolutions centralisées synchrone et asynchrone des sections 3.2 et 4.2 nécessite l'ajout de données temporelles au cas test IEEE 118 nœuds.

On remplace les données de consommation par un extrait du jeu de données UMass [4] [99], qui décrit la consommation domestique à un pas de temps d'une minute, sur l'intégralité d'une journée, soit 1440 points de données en tout. Les données des 99 premières maisons au cours de la journée arbitrairement choisie du 3 Juillet 2016 sont associées aux consommateurs déjà présents du cas test IEEE. Elles sont mises à l'échelle de manière à ce que l'ensemble de la consommation ne dépasse pas la capacité de production de l'ensemble des producteurs, et aussi de manière à ne pas saturer les lignes du réseau électrique.

A.II.2 Cas partitionné en régions

Pour tester l'algorithme d'OPF décomposé par nœuds inspiré de [54], on utilise les données originales du cas test IEEE 118 nœuds. On le teste sur un instant donné, donc il n'y a plus besoin des données temporelles de la section précédente. La décomposition du problème se faisant par régions, nous proposons différents partitionnements de ce cas test, en 10, 8, 5 et 3 régions. La décomposition en 10 régions est représentée en Figure II.1 et les autres décompositions sont déduites du cas à 10 régions, en fusionnant certaines régions ensemble.

Les nœuds associés à chaque région sont donnés en Table II.2. Les poids w_k ainsi que le paramètre ϵ utilisés dans l'algorithme d'OPF décentralisé de la section 4.4 sont aussi précisés en Table II.2. Ces paramètres sont déterminés de manière empirique pour minimiser le temps de convergence de l'algorithme décentralisé synchrone. On garde les mêmes valeurs de paramètres en asynchrone pour pouvoir comparer les différents temps de convergence à paramètres égaux.

3. `https://icseg.iti.illinois.edu/ieee-118-bus-system/`
4. `https://traces.cs.umass.edu/index.php/smart/smart`

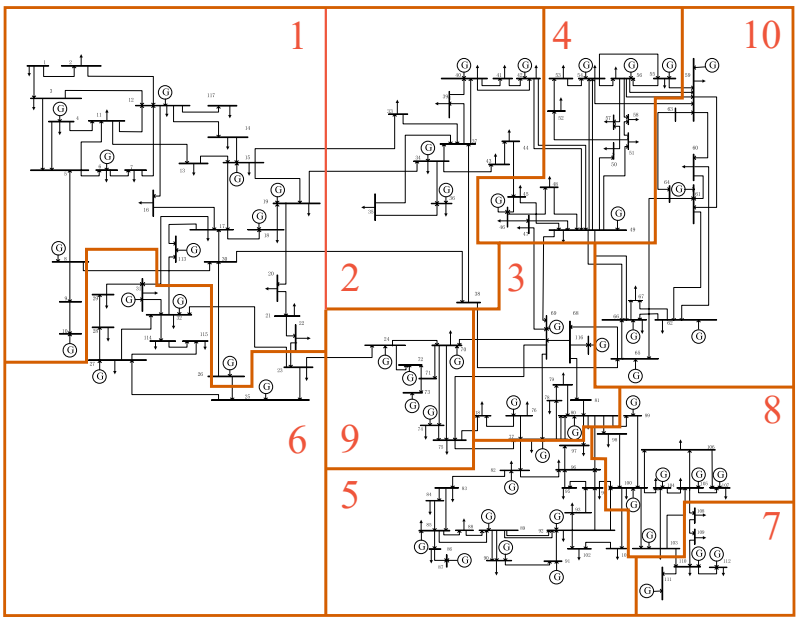


FIGURE II.1 – Représentation du cas test IEEE à 118 nœuds, partitionné en 10 régions.

Nombre de régions	Partitionnement	Poids w_k	Paramètre ϵ
10	$\mathcal{R}_1$: {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}	0.221, 0.362, 0.401, 0.157, 0.708, 0.594, 0.767, 0.048, 0.112, 0.538	2.0
8	$\mathcal{R}_1$; $\mathcal{R}_2$; $\mathcal{R}_3 \cup \mathcal{R}_9$; $\mathcal{R}_4 \cup \mathcal{R}_{10}$; $\mathcal{R}_5$; $\mathcal{R}_6$; $\mathcal{R}_7$; $\mathcal{R}_8$	0.222, 0.287, 0.415, 0.433, 0.600, 0.600, 0.388, 0.134	2.0
5	$\mathcal{R}_1 \cup \mathcal{R}_6$; $\mathcal{R}_2$; $\mathcal{R}_3 \cup \mathcal{R}_3$; $\mathcal{R}_4 \cup \mathcal{R}_{10}$; $\mathcal{R}_5 \cup \mathcal{R}_7 \cup \mathcal{R}_8$	0.139, 0.938, 0.849,0.400, 0.526	1.0
3	$\mathcal{R}_1 \cup \mathcal{R}_5 \cup \mathcal{R}_6 \cup \mathcal{R}_7 \cup \mathcal{R}_8$; $\mathcal{R}_2$; $\mathcal{R}_3 \cup \mathcal{R}_4 \cup \mathcal{R}_9 \cup \mathcal{R}_{10}$	0.184, 0.345, 0.773	1.0

TABLEAU II.2 – Régions et paramètres associés aux différents partitionnements du cas test à 118 nœuds. Les cas à 3, 5 et 8 régions sont dérivés du cas à 10 régions.

A.III Cas test à 14 nœuds pour calcul d'Optimal Power Flow

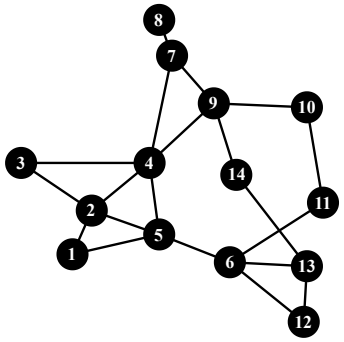


FIGURE III.2 – Représentation des nœuds et des lignes physiques du cas test IEEE 14 nœuds. Le positionnement 2D représenté ici est utilisé pour déterminer les délais de communication entre les agents.

L'algorithme asynchrone de la section 4.5 est testé sur le cas test IEEE à 14 nœuds [5], 5 générateurs et 20 lignes, illustré en Figure III.2. Ce cas a été choisi spécifiquement pour pouvoir tester une version où chaque région est assimilée à un nœud du réseau électrique. Le petit nombre de nœuds permet aux calculs de simulation de garder une durée raisonnable, d'autant plus que l'on utilise une méthode de Monte-Carlo pour obtenir une estimation statistique des temps de convergence.

Partitionnements aléatoires

Le partitionnement aléatoire utilisé pour l'étude de la section 4.5.5 est réalisé à partir du cas test IEEE à 14 nœuds et en utilisant un algorithme de partitionnement tiré du package `Metis`. Le partitionnement est déterminé avec une méthode récursive multi-niveaux [110]. Pour pouvoir obtenir des partitionnements aléatoires, un poids aléatoire entre 1 et 2 est attribué à chaque nœud avant chaque partitionnement. À partir du cas à 14 nœuds, nous pouvons ainsi créer des partitionnements avec un nombre de régions allant de 3 à 13.

5. `https://icseg.iti.illinois.edu/ieee-14-bus-system/`

A.IV Cas test à 39 nœuds pour calcul de marché endogène

L'algorithme endogène avec SO centralisé présenté au Chapitre 5 est appliqué sur le cas test IEEE à 39 nœuds [6], 31 générateurs et 46 lignes illustré en Figure IV.3. On considère que tous les générateurs communiquent avec tous les consommateurs du réseau électrique.

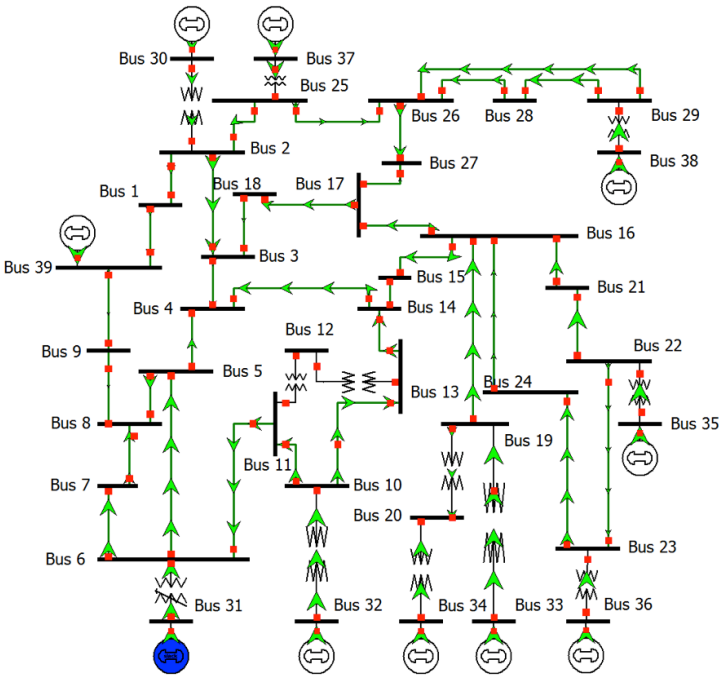


FIGURE IV.3 – Représentation du cas test IEEE à 39 nœuds.

6. https://icseg.iti.illinois.edu/ieee-39-bus-system/

Bibliographie

[1] RTE, *Futurs énergétiques 2050 Principaux résultats*, rapp. tech., 2021.

[2] Daniel S. Kirschen et Goran Strbac, *Fundamentals of Power System Economics, 2nd Ed.* Wiley, 2018, ISBN : 978-1-119-21325-3, URL : https://www.wiley.com/en-us/Fundamentals+of+Power+System+Economics%2C+2nd+Edition-p-9781119213253.

[3] Luis Baringo et Morteza Rahimiyan, « Virtual Power Plants », in : *Virtual Power Plants and Electricity Markets : Decision Making Under Uncertainty*, Cham : Springer International Publishing, 2020, p. 1-7, ISBN : 978-3-030-47602-1, DOI : 10.1007/978-3-030-47602-1_1, URL : https://doi.org/10.1007/978-3-030-47602-1_1.

[4] Amin Shokri Gazafroudi et al., « Evolving New Market Structures », in : *Pathways to a Smarter Power System*, Academic Press, 2019, chap. 6, p. 183-203, ISBN : 9780081025925, DOI : 10.1016/B978-0-08-102592-5.00006-5.

[5] Cherrelle Eid et al., « Aggregation of demand side flexibility in a smart grid : A review for European market design », in : *International Conference on the European Energy Market, EEM* 2015-Augus (2015), ISSN : 21654093, DOI : 10.1109/EEM.2015.7216712.

[6] Roman Le Goff Latimier, « Gestion et dimensionnement d'une flotte de véhicules électriques associée à une centrale photovoltaïque : co-optimisation stochastique et distribuée », thèse de doct., 2016, URL : https://tel.archives-ouvertes.fr/tel-01419931.

[7] Thomas Morstyn et al., « Using peer-to-peer energy-trading platforms to incentivize prosumers to form federated power plants », in : *Nature Energy 2018 3 :2* 3.*2* (2018), p. 94-101, ISSN : 2058-7546, DOI : 10.1038/s41560-017-0075-y, URL : https://www.nature.com/articles/s41560-017-0075-y.

[8] Nils Dorsch, Stefan Böcker et Christian Wietfeld, « ICT Requirements and Recent Developments », in : *Pathways to a Smarter Power System*, Academic Press, 2019, chap. 12, p. 343-369, ISBN : 9780081025925, DOI : 10.1016/B978-0-08-102592-5.00012-0.

[9] Amir Ghasemkhani et al., « Optimal Design of a Wide Area Measurement System for Improvement of Power Network Monitoring Using a Dynamic Multiobjective Shortest Path Algorithm », in : *IEEE Systems Journal* 11.*4* (2015), p. 2303-2314, ISSN : 1932-8184, DOI : 10.1109/JSYST.2015.2469742.

[10] Yu Wang et al., « A Distributed Control Scheme of Microgrids in Energy Internet Paradigm and Its Multisite Implementation », in : *IEEE Transactions on Industrial Informatics* 17.*2* (2021), p. 1141-1153, ISSN : 19410050, DOI : 10.1109/TII.2020.2976830.

[11] Moustafa Chenine, Kun Zhu et Lars Nordström, « Survey on priorities and communication requirements for PMU-based applications in the nordic region », in : *2009 IEEE Bucharest PowerTech : Innovative Ideas Toward the Electrical Grid of the Future* (2009), DOI : 10.1109/PTC.2009.5281956.

[12] Yu Wang et al., « Cyber-Physical Design and Implementation of Distributed Event-Triggered Secondary Control in Islanded Microgrids », in : *IEEE Transactions on Industry Applications* 55.*6* (2019), p. 5631-5642, ISSN : 19399367, DOI : 10.1109/TIA.2019.2936179.

[13] Dennis Van Der Velde, Omer Sen et Immanuel Hacker, « Towards a scalable and flexible smart grid co-simulation environment to investigate communication infrastructures for resilient distribution grid operation », in : *SEST 2021 - 4th International Conference on Smart Energy Systems and Technologies* (2021), DOI : 10.1109/SEST50973.2021.9543387.

[14] William Lardier, Quentin Varo et Jun Yan, « Quantum-sim : An open-source co-simulation platform for quantum key distribution-based smart grid communications », in : *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2019* (2019), DOI : 10.1109/SMARTGRIDCOMM.2019.8909806.

[15] Laya Das et al., « Measuring smart grid resilience : Methods, challenges and opportunities », in : *Renewable and Sustainable Energy Reviews* 130 (2020), p. 109918, ISSN : 1364-0321, DOI : 10.1016/J.RSER.2020.109918.

[16] Vincenzo Liberatore et Ahmad Al-Hammouri, « Smart grid communication and co-simulation », in : *IEEE 2011 EnergyTech, ENERGYTECH 2011* (2011), DOI : 10.1109/ENERGYTECH.2011.5948542.

[17] Nouredine Hadjsaid et al., « Modeling cyber and physical interdependencies - Application in ICT and power grids », in : *2009 IEEE/PES Power Systems Conference and Exposition, PSCE 2009* (2009), DOI : 10.1109/PSCE.2009.4840183.

[18] Mehrdad Aslani et al., « A novel clustering-based method for reliability assessment of cyber-physical microgrids considering cyber interdependencies and information transmission errors », in : (2022), DOI : 10.1016/j.apenergy.2022.119032, URL : https://doi.org/10.1016/j.apenergy.2022.119032.

[19] Catalin Gavriluta et al., « Cyber-physical framework for emulating distributed control systems in smart grids », in : *International Journal of Electrical Power and Energy Systems* 114 (2020), p. 105375, ISSN : 0142-0615, DOI : 10.1016/J.IJEPES.2019.06.033.

[20] T. Baroche, F. Moret et P. Pinson, « Prosumer markets : A unified formulation », in : *2019 IEEE Milan PowerTech, PowerTech 2019*, Institute of Electrical et Electronics Engineers Inc., 2019, ISBN : 9781538647226, DOI : 10.1109/PTC.2019.8810474.

[21] Tiago Sousa et al., *Peer-to-peer and community-based markets : A comprehensive review*, 2019, DOI : 10.1016/j.rser.2019.01.036, arXiv : 1810.09859.

[22] T. Ackermann, G. Andersson et L. Soder, « Electricity market regulations and their impact on distributed generation », in : (2002), p. 608-613, DOI : 10.1109/DRPT.2000.855735.

[23] Dimitrios J Vergados et al., « Prosumer clustering into virtual microgrids for cost reduction in renewable energy trading markets », in : *Sustainable Energy, Grids and Networks* 7 (2016), p. 90-103, DOI : 10.1016/j.segan.2016.06.002, URL : http://dx.doi.org/10.1016/j.segan.2016.06.002.

[24] Felix F. Wu et Pravin Varaiya, « Coordinated multilateral trades for electric power networks : Theory and implementation », in : *International Journal of Electrical Power and Energy Systems* 21.*2* (1999), p. 75-102, ISSN : 01420615, DOI : 10.1016/s0142-0615(98)00031-3.

[25] Hani Muhsen et al., « Business Model of Peer-to-Peer Energy Trading : A Review of Literature », in : *mdpi.com* (2022), DOI : 10.3390/su14031616, URL : https://www.mdpi.com/1477152.

[26] Chankook Park et Taeseok Yong, « Comparative review and discussion on P2P electricity trading », in : *Energy Procedia* 128 (2017), p. 10-12, DOI : 10.1016/j.egypro.2017.09.003, URL : www.sciencedirect.comwww.elsevier.com/locate/procediawww.sciencedirect.comwww.elsevier.com/locate/procedia1876-6102.

[27] Mission Innovation, *Piclo peer-to-peer energy trading platform*, URL : http://mission-innovation.net/our-work/mission-innovation-breakthroughs/piclo-peer-to-peer-energy-trading-platform/ (visité le 02/04/2022).

[28] Piclo, *The leading independent marketplace for flexibility services*, URL : https://www.piclo.energy/product (visité le 02/04/2022).

[29] Vandebron, *L'énergie durable du sol néerlandais*, URL : https://vandebron.nl/ (visité le 02/04/2022).

[30] Sonnen, *sonnenCommunity*, URL : https://sonnengroup.com/sonnencommunity/ (visité le 02/04/2022).

[31] Brooklyn Microgrid, *Community Powered Energy*, URL : https://www.brooklyn.energy/ (visité le 02/04/2022).

[32] Esther Mengelkamp et al., « Designing microgrid energy markets A case study : The Brooklyn Microgrid », in : (2017), DOI : 10.1016/j.apenergy.2017.06.054, URL : http://dx.doi.org/10.1016/j.apenergy.2017.06.054.

[33] Wayes Tushar et al., « Peer-to-Peer Trading in Electricity Networks : An Overview », in : *IEEE Transactions on Smart Grid* 11.*4* (2020), p. 3185-3200, ISSN : 19493061, DOI : 10.1109/TSG.2020.2969657, arXiv : 2001.06882.

[34] Wayes Tushar et al., « A motivational game-theoretic approach for peer-to-peer energy trading in the smart grid », in : (2019), DOI : 10.1016/j.apenergy.2019.03.111, URL : https://doi.org/10.1016/j.apenergy.2019.03.111.

[35] Chenghua Zhang et al., « Peer-to-Peer energy trading in a Microgrid », in : *Applied Energy* 220 (2018), p. 1-12, ISSN : 03062619, DOI : 10.1016/J.APENERGY.2018.03.010, URL : https://doi.org/10.1016/j.apenergy.2018.03.010.

[36] Amrit Paudel et al., « Peer-to-peer energy trading in a prosumer-based community microgrid : A game-theoretic model », in : *IEEE Transactions on Industrial Electronics* 66.*8* (2019), p. 6087-6097, ISSN : 02780046, DOI : 10.1109/TIE.2018.2874578.

[37] Wayes Tushar et al., « Challenges and prospects for negawatt trading in light of recent technological developments », in : *nature.com* (2020), arXiv : 2007.08331v1, URL : https://www.nature.com/articles/s41560-020-0671-0.

[38] Z Jing, M Pipattanasomporn 2019 IEEE PES GTD . . . et undefined 2019, « Blockchain-based Negawatt trading platform : Conceptual architecture and case studies », in : *ieeexplore.ieee.org* (), URL : https://ieeexplore.ieee.org/abstract/document/8715890/.

[39] Juan M. Morales et al., *Integrating renewables in electricity markets - Operational problems*, t. 205, 2014, p. 429, ISBN : 9781461494102, DOI : 10.1007/978-1-4614-9411-9.

[40] Wayes Tushar et al., « Grid Influenced Peer-to-Peer Energy Trading », in : *IEEE Transactions on Smart Grid* 11.*2* (2020), p. 1407-1418, ISSN : 19493061, DOI : 10.1109/TSG.2019.2937981, arXiv : 1908.09449.

[41] Cherrelle Eid et al., « Managing electric flexibility from Distributed Energy Resources : A review of incentives for market design », in : *Renewable and Sustainable Energy Reviews* 64 (2016), p. 237-247, ISSN : 18790690, DOI : 10.1016/J.RSER.2016.06.008.

[42] Vitor N Coelho et al., « Multi-agent systems applied for energy systems integration : State-of-the-art applications and trends in microgrids », in : (2016), DOI : 10.1016/j.apenergy.2016.10.056, URL : http://dx.doi.org/10.1016/j.apenergy.2016.10.056.

[43] Paul Codani, Marc Petit et Yannick Perez, « Participation of an electric vehicle fleet to primary frequency control in France », in : *International Journal of Electric and Hybrid Vehicles* 7.*3* (2015), p. 233-249, ISSN : 17514096, DOI : 10.1504/IJEHV.2015.071639.

[44] Etienne Sorin, Lucien Bobo et Pierre Pinson, « Consensus-Based Approach to Peer-to-Peer Electricity Markets with Product Differentiation », in : *IEEE Transactions on Power Systems* 34.*2* (2019), p. 994-1004, ISSN : 08858950, DOI : 10.1109/TPWRS.2018.2872880, arXiv : 1804.03521.

[45] Thomas Baroche et al., « Exogenous Cost Allocation in Peer-to-Peer Electricity Markets », in : *IEEE Transactions on Power Systems* 34.*4* (2019), p. 2553-2564, ISSN : 15580679, DOI : 10.1109/TPWRS.2019.2896654, arXiv : 1803.02159.

[46] Thomas Baroche, « Peer-to-peer electricity markets in power systems », thèse de doct., École Normale Supérieure de Rennes, 2020.

[47] Jaysson Guerrero, Archie C. Chapman et Gregor Verbic, « Decentralized P2P Energy Trading under Network Constraints in a Low-Voltage Network », in : *IEEE Transactions on Smart Grid* (2018), ISSN : 19493053, DOI : 10.1109/TSG.2018.2878445, arXiv : 1809.06976.

[48] J Carpentier, « Contribution to the economic dispatch problem », in : *Bulletin de la Societe Francaise des Electriciens* 3.*8* (1962), p. 431-447.

[49] J. Carpentier, « Optimal power flows », in : *International Journal of Electrical Power and Energy Systems* 1.*1* (1979), p. 3-15, ISSN : 01420615, DOI : 10.1016/0142-0615(79)90026-7.

[50] Daniel K. Molzahn, « Computing the Feasible Spaces of Optimal Power Flow Problems », in : *IEEE Transactions on Power Systems* 32.*6* (2017), p. 4752-4763, ISSN : 08858950, DOI : 10.1109/TPWRS.2017.2682058, arXiv : 1608.00598.

[51] Javad Lavaei et Steven H. Low, « Zero duality gap in optimal power flow problem », in : *IEEE Transactions on Power Systems* 27.*1* (2012), p. 92-107, ISSN : 08858950, DOI : 10.1109/TPWRS.2011.2160974.

[52] Anya Castillo et Richard P O'neill, *Survey of Approaches to Solving the ACOPF*, rapp. tech., Federal Energy Regulatory Commission, 2013.

[53] Balho H. Kim et Ross Baldick, « A comparison of distributed optimal power flow algorithms », in : *IEEE Transactions on Power Systems* 15.*2* (2000), p. 599-604, ISSN : 08858950, DOI : 10.1109/59.867147.

[54] Tomaso Erseghe, « Distributed optimal power flow using ADMM », in : *IEEE Transactions on Power Systems* 29.*5* (2014), p. 2370-2380, ISSN : 08858950, DOI : 10.1109/TPWRS.2014.2306495.

[55] Junyao Guo, Gabriela Hug et Ozan K. Tonguz, « A Case for Nonconvex Distributed Optimization in Large-Scale Power Systems », in : *IEEE Transactions on Power Systems* 32.*5* (2017), p. 3842-3851, ISSN : 08858950, DOI : 10.1109/TPWRS.2016.2636811.

[56] Gabriela Hug-Glanzmann et Göran Andersson, « Decentralized optimal power flow control for overlapping areas in power systems », in : *IEEE Transactions on Power Systems* 24.*1* (2009), p. 327-336, ISSN : 08858950, DOI : 10.1109/TPWRS.2008.2006998.

[57] Alexander Engelmann et al., « Toward distributed OPF using ALADIN », in : *IEEE Transactions on Power Systems* 34.*1* (2019), p. 584-594, ISSN : 08858950, DOI : 10.1109/TPWRS.2018.2867682, arXiv : 1802.08603.

[58] Xinliang Dai et al., « Rapid Scalable Distributed Power Flow with Open-Source Implementation », in : (), arXiv : 2203.16335v1, URL : https://github.com/KIT-IAI/rapidPF.

[59] Amin Kargarian et al., « Toward Distributed/Decentralized DC Optimal Power Flow Implementation in Future Electric Power Systems », in : *IEEE Transactions on Smart Grid* 9.*4* (2018), p. 2574-2594, ISSN : 19493053, DOI : 10.1109/TSG.2016.2614904.

[60] Tian Liu et al., « Energy management of cooperative microgrids with P2P energy sharing in distribution networks », in : *2015 IEEE International Conference on Smart Grid Communications, SmartGridComm 2015* (2016), p. 410-415, DOI : 10.1109/SMARTGRIDCOMM.2015.7436335.

[61] Qiuyu Peng et Steven H. Low, « Distributed optimal power flow algorithm for radial networks, I : Balanced single phase case », in : *IEEE Transactions on Smart Grid* 9.*1* (2018), p. 111-121, ISSN : 19493053, DOI : 10.1109/TSG.2016.2546305.

[62] Daniel K. Molzahn et al., « Implementation of a large-scale optimal power flow solver based on semidefinite programming », in : *IEEE Transactions on Power Systems* 28.*4* (2013), p. 3987-3998, ISSN : 08858950, DOI : 10.1109/TPWRS.2013.2258044.

[63] Emiliano Dall'Anese, Hao Zhu et Georgios B. Giannakis, « Distributed optimal power flow for smart microgrids », in : *IEEE Transactions on Smart Grid* 4.*3* (2013), p. 1464-1475, ISSN : 19493053, DOI : 10.1109/TSG.2013.2248175, arXiv : 1211.5856.

[64] Joshua Adam Taylor, *Convex Optimization of Power Systems*, Cambridge University Press, 2015, isbn : 9781139924672, doi : 10.1017/CBO9781139924672.

[65] Junyao Guo, Gabriela Hug et Ozan K. Tonguz, « Intelligent Partitioning in Distributed Optimization of Electric Power Systems », in : *IEEE Transactions on Smart Grid* 7.*3* (2016), p. 1249-1258, issn : 19493053, doi : 10.1109/TSG.2015.2490553.

[66] Dimitri Bertsekas et John Tsitsiklis, *Parallel and distributed computation : numerical methods*, Athena Scientific, 1989.

[67] Anni Huang et al., « Asynchronous decentralized method for interconnected electricity markets », in : *International Journal of Electrical Power and Energy Systems* 30.*4* (2008), p. 283-290, issn : 01420615, doi : 10.1016/j.ijepes.2007.10.001.

[68] Franck Iutzeler et al., « Asynchronous distributed optimization using a randomized alternating direction method of multipliers », in : *Proceedings of the IEEE Conference on Decision and Control*, 2013, isbn : 9781467357173, doi : 10.1109/CDC.2013.6760448, arXiv : 1303.2837.

[69] Azary Abboud, « Distributed optimization in large interconnected systems using ADMM », in : (2016), url : https://tel.archives-ouvertes.fr/tel-01306958.

[70] Ruiliang Zhang et James T. Kwok, « Asynchronous distributed ADMM for consensus optimization », in : *31st International Conference on Machine Learning, ICML 2014* 5.*2* (2014), p. 3689-3697.

[71] Tsung Hui Chang et al., « Asynchronous Distributed ADMM for Large-Scale Optimization - Part I : Algorithm and Convergence Analysis », in : *IEEE Transactions on Signal Processing* 64.*12* (2016), p. 3118-3130, issn : 1053587X, doi : 10.1109/TSP.2016.2537271, arXiv : 1509.02597.

[72] Rui Zhu, Di Niu et Zongpeng Li, « A Block-wise, Asynchronous and Distributed ADMM Algorithm for General Form Consensus Optimization », in : (2018), arXiv : 1802.08882, url : http://arxiv.org/abs/1802.08882.

[73] Jiafeng 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 et Electronics Engineers Inc., 2019, p. 22-28, isbn : 9781728104669, doi : 10.1109/HPBDIS.2019.8735442.

[74] Nicola Bastianello et al., « Asynchronous Distributed Optimization over Lossy Networks via Relaxed ADMM : Stability and Linear Convergence », in : *IEEE Transactions on Automatic Control* 66.*6* (2021), p. 2620-2635, doi : 10.1109/TAC.2020.3011358.

[75] Layla Majzoobi, Vahid Shah-Mansouri et Farshad Lahouti, « Analysis of distributed ADMM algorithm for consensus optimisation over lossy networks », in : *IET Signal Process* 12.*6* (2018), p. 786-794, issn : 1751-9675, doi : 10.1049/iet-spr.2018.0033, url : www.ietdl.org.

[76] Jianhua Zhang et al., « Convergence analysis of ADMM-based power system mode estimation under asynchronous wide-area communication delays », in : *IEEE Power and Energy Society General Meeting* 2015-Septe (2015), issn : 19449933, doi : 10.1109/PESGM.2015.7286038.

[77] Jianhua Zhang et al., « ADMM Optimization Strategies for Wide-Area Oscillation Monitoring in Power Systems under Asynchronous Communication Delays », in : *IEEE Transactions on Smart Grid* 7.*4* (2016), p. 2123-2133, ISSN : 19493053, DOI : 10.1109/TSG.2016.2547939.

[78] Jiangjiao Xu, Hongjian Sun et Chris J. Dent, « ADMM-based Distributed OPF Problem Meets Stochastic Communication Delay », in : *IEEE Transactions on Smart Grid* 10.*5* (2018), p. 5046-5056, ISSN : 19493053, DOI : 10.1109/TSG.2018.2873650.

[79] Junyao Guo, Gabriela Hug et Ozan Tonguz, « Impact of communication delay on asynchronous distributed optimal power flow using ADMM », in : *2017 IEEE International Conference on Smart Grid Communications, SmartGridComm 2017*, 2018, ISBN : 9781538640555, DOI : 10.1109/SmartGridComm.2017.8340718, arXiv : 1711.01702.

[80] Md Habib Ullah et Jae Do 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 et Electronics Engineers Inc., 2019, ISBN : 9781538682326, DOI : 10.1109/ISGT.2019.8791568.

[81] Chin Yao Chang, Jorge Cortes et Sonia Martinez, « Scheduled-asynchronous distributed algorithm for optimal power flow », in : *IEEE Transactions on Control of Network Systems* 6.*1* (2019), p. 261-275, ISSN : 23255870, DOI : 10.1109/TCNS.2018.2809963, arXiv : 1710.02852.

[82] Benjamin S Millar et Danchi Jiang, « Asynchronous consensus for optimal power flow control in smart grid with zero power mismatch », in : *Journal of Modern Power Systems and Clean Energy* 6.*3* (2018), p. 412-422.

[83] Javad Mohammadi, Gabriela Hug et Soummya Kar, « Asynchronous distributed approach for DC Optimal Power Flow », in : *2015 IEEE Eindhoven PowerTech, PowerTech 2015*, Institute of Electrical et Electronics Engineers Inc., 2015, ISBN : 9781479976935, DOI : 10.1109/PTC.2015.7232606.

[84] Xinyu Liang et al., « Relaxed alternating direction method of multipliers for hedging communication packet loss in integrated electrical and heating system », in : *Journal of Modern Power Systems and Clean Energy* 8.*5* (2020), p. 874-883, DOI : 10.35833/MPCE.2020.000163.

[85] Yushuai Li et al., « Event-Triggered-Based Distributed Cooperative Energy Management for Multienergy Systems », in : *IEEE Transactions on Industrial Informatics* 15.*4* (2019), p. 2008-2022, DOI : 10.1109/TII.2018.2862436.

[86] Paritosh Ramanan et al., « Asynchronous Decentralized Framework for Unit Commitment in Power Systems », in : *Procedia Computer Science* 108 (2017), p. 665-674, ISSN : 1877-0509, DOI : 10.1016/J.PROCS.2017.05.038.

[87] F. Moret et al., « Negotiation algorithms for peer-to-peer electricity markets : Computational properties », in : *20th Power Systems Computation Conference, PSCC 2018*, Institute of Electrical et Electronics Engineers Inc., 2018, ISBN : 9781910963104, DOI : 10.23919/PSCC.2018.8442914.

[88] Alyssia Dong et al., « Convergence analysis of an asynchronous peer-to-peer market with communication delays », in : *Sustainable Energy, Grids and Networks* 26 (2021), p. 100475, ISSN : 2352-4677, DOI : 10.1016/J.SEGAN.2021.100475.

[89] Alyssia Dong et al., « Implémentation asynchrone d'un algorithme de marché de l'électricité décentralisé », in : *Symposium de Génie Électrique*, 2020, URL : https://hal.archives-ouvertes.fr/hal-03485938.

[90] Alyssia Dong et al., « Asynchronous algorithm of an endogenous peer-to-peer electricity market », in : *2021 IEEE Madrid PowerTech, PowerTech 2021 - Conference Proceedings* (2021), DOI : 10.1109/POWERTECH46648.2021.9495009.

[91] Alyssia Dong, Roman Le Goff Latimier et Hamid Ben Ahmed, « Asynchronous implementation of a distributed optimal power flow algorithm », 2022.

[92] Alyssia Dong, Roman Le Goff Latimier et Hamid Ben Ahmed, « Asynchronous ADMM based decentralized optimal power flow with edge decomposition », 2022.

[93] James F Kurose et al., *Computer Networking A Top-Down Approach*, 7e éd., 2017, ISBN : 9780133594140, URL : www.pearsoned.com/permissions/..

[94] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez et Robert John Thomas, « MATPOWER : Steady-state operations, planning, and analysis tools for power systems research and education », in : *IEEE Transactions on Power Systems* 26.*1* (2011), p. 12-19, DOI : 10.1109/TPWRS.2010.2051168.

[95] Ben Lauwens, *SimJulia : A discrete event process oriented simulation framework written in Julia*, 2020, URL : https://github.com/BenLauwens/SimJulia.jl.

[96] Stephen Boyd et al., *Distributed optimization and statistical learning via the alternating direction method of multipliers*, 2010, DOI : 10.1561/2200000016.

[97] H S Esch et al., « Online matching and preferences in future electricity markets », in : 1 (2019).

[98] Bartolomeo Stellato et al., « OSQP : an operator splitting solver for quadratic programs », in : *Mathematical Programming Computation* (2020), ISSN : 18672957, DOI : 10.1007/s12532-020-00179-2.

[99] Sean Barker et al., « Smart* : An Open Data Set and Tools for Enabling Research in Sustainable Homes », in : (2012).

[100] R. Le Goff Latimier, T. Baroche et H. Ben Ahmed, « Mitigation of communication costs in peer-to-peer electricity markets », in : *2019 IEEE Milan PowerTech, PowerTech 2019* (2019), DOI : 10.1109/PTC.2019.8810716.

[101] Gurvan Jodin et al., « opENS : un concept de prototype de Smart Grid à puissance réduite, faible coût, libre et ouvert », in : (), URL : www.epfl.ch/labs/desl-pwrs/smartgrid/.

[102] Alyssia Dong, *RPi Asynchronous - Gitlab*, URL : https://gitlab.com/alyssiadong/rpi-asynchronous (visité le 29/03/2022).

[103] MQTT.ORG, *MQTT - The Standard for IoT Messaging*, URL : https://mqtt.org/ (visité le 29/03/2022).

[104] HiveMQ, *MQTT Essentials part 2 : publish-subscribe*, URL : https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/ (visité le 29/03/2022).

[105] InfluxData, *InfluxDB 1.X : Open Source Time Series Platform*, URL : https://www.influxdata.com/time-series-platform/ (visité le 29/03/2022).

[106] Carleton Coffrin et al., « PowerModels.jl : An Open-source framework for exploring power flow formulations », in : *20th Power Systems Computation Conference, PSCC 2018*, Institute of Electrical et Electronics Engineers Inc., 2018, ISBN : 9781910963104, DOI : 10.23919/PSCC.2018.8442948.

[107] Stephen Frank, Ingrida Steponavice et Steffen Rebennack, « Optimal power flow : A bibliographic survey I Formulations and deterministic methods », in : *Energy Systems* 3.*3* (2012), p. 221-258, ISSN : 18683967, DOI : 10.1007/S12667-012-0056-Y.

[108] Andreas Wächter et Lorenz T. Biegler, « On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming », in : *Mathematical Programming 2005 106 :1* 106.*1* (2005), p. 25-57, ISSN : 1436-4646, DOI : 10.1007/S10107-004-0559-Y, URL : https://link.springer.com/article/10.1007/s10107-004-0559-y.

[109] Junyao Guo, Gabriela Hug et Ozan K. Tonguz, « On the Role of Communications Plane in Distributed Optimization of Power Systems », in : *IEEE Transactions on Industrial Informatics* 14.*7* (2018), p. 2903-2913, ISSN : 15513203, DOI : 10.1109/TII.2017.2774243.

[110] George Karypis et Vipin Kumar, « A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs », in : *http ://dx.doi.org/10.1137/S1064827595287997* 20.*1* (2006), p. 359-392, ISSN : 10648275, DOI : 10.1137/S1064827595287997.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

école normale supérieure rennes

Titre : Algorithmes asynchrones pour la gestion décentralisée des réseaux électriques soumis aux aléas de communication

Mot clés : Gestion décentralisée, ADMM, marché de l'électricité, OPF, algorithme asynchrone, réseau de communication

Résumé : La thématique des smart grids est fondée sur l'interaction d'un réseau électrique et d'un réseau de communication reliant des agents distribués, « intelligents » et communicants. Dans le cas d'un réseau électrique classique, les événements imprévus (perte d'un équipement, ligne saturée, etc) sont principalement amortis et compensés par l'inertie du réseau et ses centrales de réserves. Un smart grid se propose de réduire cette dépendance à l'aide de stratégies de gestion distribuées intelligentes faisant appel à des échanges d'information et donc à un réseau de communication. La question de l'impact de la fiabilité du réseau de communication est donc ici cruciale. Le présent projet se propose donc d'analyser finement l'impact des aléas de communication sur la résilience d'un smart grid et de développer des stratégies de gestion prenant en compte ces aléas pour garantir l'opérabilité et l'efficience du smart grid. On se penche en particulier sur trois types d'algorithmes : un algorithme de marché pair à pair, un algorithme d'Optimal Power Flow qui prend en compte les contraintes physiques du réseau électrique, et enfin un algorithme de marché pair à pair endogène. Nous avons étudié de manière approfondie le principe d'algorithme asynchrone, qui permet de limiter les délais d'attente à chaque itération, ainsi que les effets des aléas du réseau de communication sur la résolution des versions asynchrones de ces algorithmes.

Title: Asynchrounous algorithms for decentralized management of electrical networks subject to communication hazards

Keywords: Decentralized management, ADMM, electrical market, OPF, asynchronous algorithm, communication network

Abstract: Smart grids are based on the interaction of an electrical network and a communication network between distributed, "intelligent" and communicating agents. In the case of a traditional electrical network, unforeseen events (loss of an equipment, line congestion, etc.) are mainly absorbed and compensated by the inertia of the network and its operating reserve. Smart grids aim to reduce this dependency by using intelligent distributed management strategies using information exchange and therefore based on a communication network. The impact of the communication network's reliability is a critical issue that we must address. This project aims at analyzing the impact of communication hazards on the resilience of a smart grid and at developing strategies that take these hazards into account in order to guarantee the operability and efficiency of the smart grid. In particular, we investigate three types of algorithms: a peer-to-peer market algorithm, an Optimal Power Flow algorithm that takes into account the physical constraints of the power grid, and finally an endogenous peer-to-peer market algorithm. We have studied in depth the principles of asynchronous algorithms, which help to minimize the waiting time at each iteration, as well as the effects of communication network hazards on the asynchronous resolution of these algorithms.