Algorithmes hybrides pour la gestion intelligente de l'énergie dans les smart grids

Robin Roche, Lhassane Idoumghar, Benjamin Blunier, Abdelkader Miraoui

Algorithmes hybrides pour la gestion intelligente de l'énergie dans les smart grids

Robin ROCHE[1], Lhassane IDOUMGHAR[2], Bejamin BLUNIER[†], Abdellatif MIRAOUI[1]

[1] IRTES-SeT, Université de Technologie de Belfort-Montbéliard,
13 rue Thierry Mieg 90000 Belfort France
robin.roche@utbm.fr, abdellatif.miraoui@utbm.fr

[2] Laboratoire LMIA, Université de Haute-Alsace
4 rue des Frères Lumière 68093 Mulhouse, France
lhassane.idoumghar@uha.fr

Résumé :
L'Union Européenne a fixé des objectifs en termes d'émissions de CO_2, d'efficacité énergétique et de production d'énergie d'origine renouvelable. Pour atteindre ces objectifs, un changement de paradigme du secteur de l'énergie s'impose et cela se traduit par la nécessité d'évoluer vers un réseau électrique plus intelligent, dit smart grid. Les travaux présentés dans cet article proposent un système de gestion d'énergie flexible et efficace, ouvrant de larges possibilités d'applications dans ces nouveaux réseaux, où la communication et l'intelligence seront centraux. Un système multi-agents est hybridé avec des algorithmes d'optimisation, basés sur des métaheuristiques, pour obtenir une gestion d'énergie qui satisfait divers objectifs et contraintes. Plusieurs cas d'application sont également présentés, notamment sur une centrale de production et un micro-réseau.

Mots-clés : Systèmes multi-agents, Métaheuristiques, Gestion d'énergie, Algorithmes hybrides.

1 Introduction

Le réchauffement climatique et la raréfaction des sources d'énergie fossiles ont conduit les États à prendre des mesures pour favoriser l'émergence de pratiques propres dans le secteur de l'énergie. À titre d'exemple, l'Union Européenne s'est fixée pour objectifs d'augmenter l'efficacité énergétique de 20%, de réduire les émissions de CO_2 de 20% et de faire passer la part des énergies renouvelables à 20% d'ici 2020. Pour atteindre ces objectifs de lourdes modifications dans les processus de production et consommation de l'énergie sont nécessaires. Au cœur de ces modifications, le réseau électrique est amené à évoluer profondément vers ce que l'on nomme “smart grid", ou réseau électrique intelligent (Simoes *et al.*, 2011) (voir figure 1).

Dans ce réseau modernisé, le réseau électrique est doublé d'un réseau de communication et de contrôle, devant apporter une réelle intelligence à l'ensemble. Il permet d'intégrer les actions de tous les acteurs qui y sont connectés (consommateurs, producteurs, stockeurs), pour fournir de l'énergie électrique de façon durable, économique et sûre.

Les travaux présentés dans cet article s'inscrivent dans cette perspective, et cherchent à lever en partie certains des verrous liés à l'émergence de systèmes de gestion d'énergie intelligents. Ils visent à proposer une solution de gestion d'énergie à la fois efficace et flexible, en alliant les avantages offerts par deux technologies : les systèmes multi-agents, et les algorithmes d'optimisation par métaheuristiques. Des exemples d'applications seront présentés afin d'illustrer les propos développés.

2 Architectures de contrôle décentralisées

La flexibilité du système de gestion proposé est permise par l'utilisation d'un système multi-agents comme base de construction de l'architecture du système.

[†] Une pensée particulière à Benjamin qui nous a quitté le 22 Février 2012.

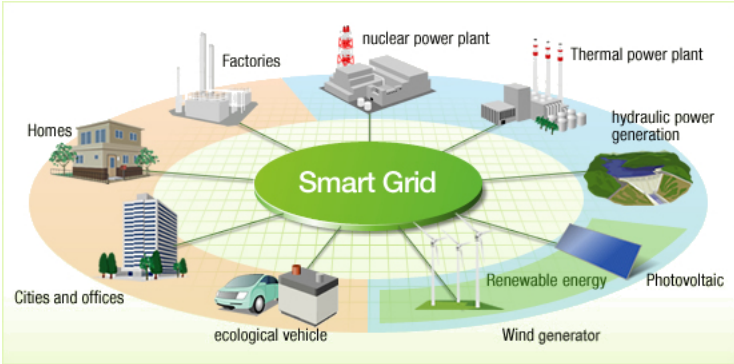

FIG. 1 – Un réseau smart grid.

2.1 Architecture d'un système multi-agents

Un des principaux objectifs du Système de Gestion d'Energie (SGE) est d'atteindre un très haut niveau de flexibilité, non seulement au cours de son fonctionnement, mais aussi pendant les pannes et durant tous son cycle de vie : le système conçu doit être capable de s'adapter à la plupart des changements dans l'architecture du micro-réseau, qu'ils soient intentionnels (par exemple, ajout d'une turbine pour augmenter la capacité de production, ou son arrêt pour la maintenance) ou non (par exemple, après une panne). Les systèmes multi-agents, grâce à leurs propriétés et caractéristiques, permettent d'atteindre ce niveau de flexibilité.

2.1.1 Les Systèmes Multi-Agents

Les systèmes multi-agents (SMA) représentent une forme d'intelligence artificielle distribuée (Ferber, 1999; Hilaire *et al.*, 2008; Mazigh *et al.*, 2011). Ils permettent une approche systémique des problèmes qui peuvent se décomposer en plusieurs entités en interaction. Un SMA est composé de plusieurs entités appelées agents, qui ont leur propre intelligence et autonomie (Ferber, 1999). Ces agents situés dans un environnement peuvent interagir en se basant sur leur perception (par exemple, des mesures) et en exécutant des actions (voir figure 2). Chaque agent a sa propre perception de l'environnement et prend ses décisions sur cette base. En interagissant les uns avec les autres, par exemple grâce à des échanges de messages, les agents peuvent poursuivre leurs propres objectifs et contribuer ainsi à la réalisation du but global visé par le SMA. Ces interactions peuvent être de nature coopératives ou compétitives, c'est-à-dire, ces objectifs peuvent être contradictoires ou non. Les agents peuvent avoir différents niveaux d'intelligence, certains peuvent être dotés d'une intelligence simple tandis que d'autres peuvent disposer de mécanismes de pensée beaucoup plus complexes (par exemple, utilisation des techniques d'apprentissage).

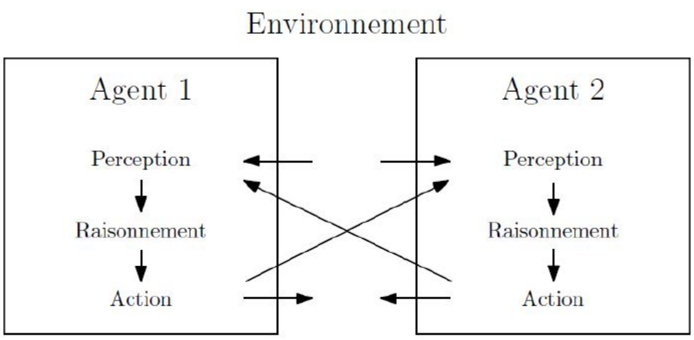

FIG. 2 – Un SMA est constitué d'agents en interaction avec leur environnement.

2.1.2 Intérêt pour un système de gestion d'énergie

Chaque élément composant le réseau électrique peut être considéré comme un agent, avec des rôles, des contraintes, des degrés de perception de l'environnement et moyens d'action propres. Les agents correspondant aux sources, aux moyens de stockage, aux charges électriques, ou encore aux transformateurs, forment donc un SMA qui est l'image du réseau électrique physique.

L'utilisation du concept de SMA dans la problématique des smart grids est motivée par les 3 raisons (Roche *et al.*, 2010) suivantes :

– De par leur structure distribuée, les SMA sont facilement adaptables à un système distribué dans l'espace tel qu'un réseau électrique. Ceci permet notamment de réduire les besoins en bande passante et en puissance de calcul : les problèmes (coupures, instabilités, etc.) peuvent au moins en partie être traités localement, permettant une plus grande réactivité. Au besoin, ils peuvent également coopérer ou entrer en compétition.
– Les SMA sont flexibles, grâce à leur architecture distribuée, et peuvent donc s'adapter aux évolutions du réseau, qu'il s'agisse de pannes, d'ajout ou de suppression de composants, et ce, que ces évènements aient été prévus à la conception du système ou non. Ils sont donc tolérants aux fautes et permettent un fonctionnement dégradé et *plug & play*.
– Les SMA sont proactifs, en cherchant à satisfaire leurs besoins ou objectifs en fonction de leurs rôles et contraintes. Une charge cherchera donc à être alimentée, tout comme un gestionnaire de centrale cherchera à minimiser ses coûts tout en respectant ses contraintes opérationnelles et légales. En cas de besoin, les agents peuvent également aller chercher les informations dont ils ont besoin pour prendre des décisions et planifier des actions.

On remarque donc une adéquation entre les caractéristiques des SMA et celles requises par les smart grids, montrant l'intérêt de leur utilisation.

2.1.3 Architecture sélectionnée

Dans un SMA, les agents peuvent être organisés de nombreuses manières : équipes, hiérarchies, coalitions, fédérations, marchés, etc. Dans ce travail nous avons opté pour une forme de structure en fédérations. Celle-ci est la plus proche de la structure que l'on peut imaginer que les smart grids prendront à terme, à savoir un ensemble de micro-réseaux partiellement autonomes, interconnectés entre-eux et s'échangeant de l'énergie. La Figure 3 l'illustre, et présente plusieurs micro-réseaux. À cet ensemble de micro-réseaux on rajoute un système de gestion, hiérarchisé, qui sera à la fois une interface de coordination des composants et un acteur de *trading* d'énergie avec les autres systèmes de gestion.

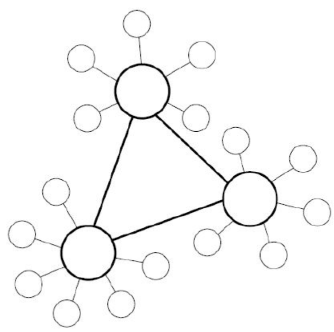

FIG. 3 – Exemple de plusieurs micro-réseaux qui forment un smart grid.

L'architecture de SMA proposée (figure 4) se compose de plusieurs agents, répartis en trois catégories :

– Agents de contrôle des composants physiques : turbines, stockage, charge et réseau. Un agent *turbine* est créé pour chaque turbine dans le micro-réseau. L'agent réside dans le système de contrôle propre à la turbine. Il communique avec l'agent *SCADA*[1] en lui envoyant son état actuel grâce à ses mesures, et reçoit des consignes en retour. Cet agent est le seul à avoir accès à toutes les données disponibles sur

[1]Supervisory Control And Data Acquisition

la turbine. De même pour les autres agents, ils peuvent dialoguer avec l'agent *SCADA* en lui envoyant leurs mesures actuelles et reçoivent des recommandations en retour.

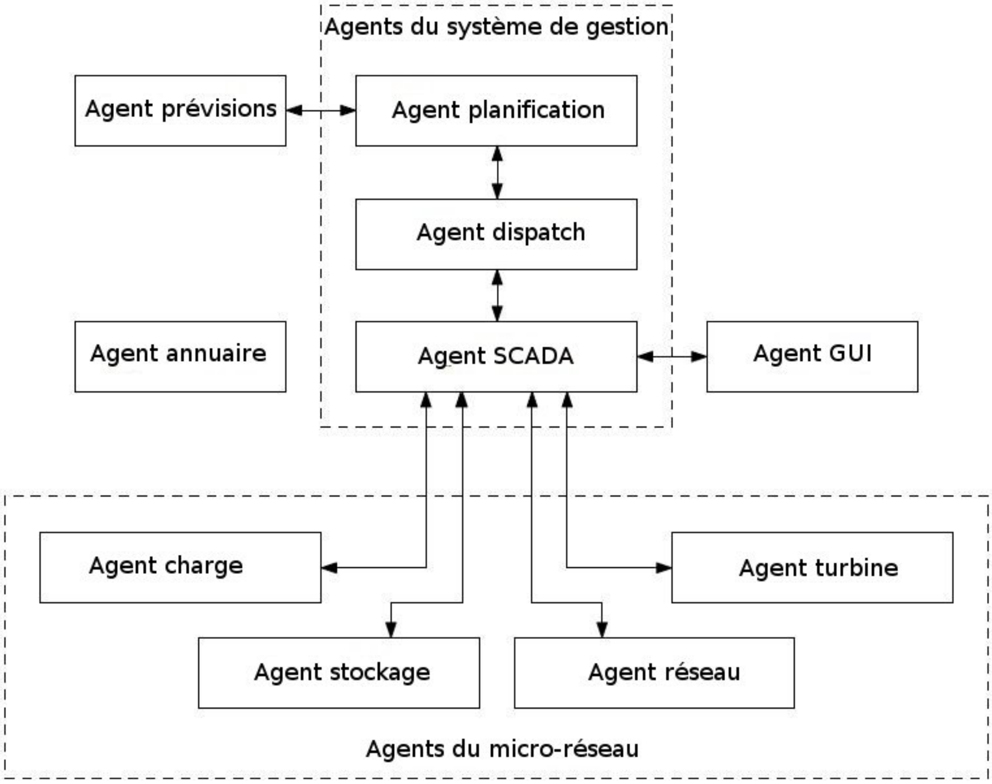

FIG. 4 – Le système multi-agents proposé est constitué de plusieurs types d'agents, avec un fonctionnement propre à chacun d'entre eux. Chaque flèche correspond à un canal de communication habituel entre les agents en mode de fonctionnement normal.

– Agents du système de gestion : *SCADA*, *dispatching* et *planification*. L'agent *SCADA* (système d'acquisition et de supervision) joue le rôle d'un superviseur et contrôleur commercial. Il recueille les données des différents agents qui forment le micro-réseau, et leur transmet les consignes reçus auprès de l'agent chargé de dispatching. Il fournit également aux autres agents les informations dont ils ont besoin pour fonctionner, à chaque fois qu'ils le demandent et si ils sont autorisés à le faire. L'agent *dispatch* calcule, à partir des données reçues de l'agent *SCADA*, le point de fonctionnement optimal pour chaque agent du micro-réseau. Il maintient également une base de données contenant les informations dont il a besoin pour déterminer la bonne répartition des puissances en utilisant des données économiques et environnementales. L'agent *planification* est chargé de la planification à l'avance, si possible optimale, du fonctionnement de l'ensemble, à l'aide de prévisions de production, de demande et de prix émis par des agents.
– Agents auxiliaires : formés principalement de l'agent *annuaire* et *GUI*. L'agent *annuaire* sert de serveur de noms et de services, il fournit aux agents qui en font la demande les noms et adresses des agents correspondant à leurs besoins (tous les agents peuvent communiquer avec lui). L'agent *GUI*, gère l'interface homme-machine, reçoit les mises à jour sur l'état actuel du système, et permet aux opérateurs humains d'assurer la supervision. Il transmet également les commandes émises par les opérateurs à l'agent *SCADA*.

2.1.4 Structure d'un agent

Tous les agents sont basés sur une structure générique (Figure 5). Quand un agent est créé, il exécute un ensemble d'instructions d'initialisation. Ces instructions lui permettent d'obtenir des informations sur le composant dont il est en charge (par exemple la turbine), et de s'inscrire auprès des autres agents avec lesquels il interagira. Il se met en mode d'attente d'un message en provenance d'un autre agent. Ce message peut être une demande ou peut contenir une information dont l'agent aurait besoin. En fonction du contenu du message, l'agent exécute une séquence donnée d'instructions, comme par exemple lancer des mesures, interagir avec les autres agents. A l'issu de cette séquence, l'agent se remet en mode d'attente d'un message, et peut exécuter des tâches d'arrière-plan jusqu'à ce qu'un nouveau message soit reçu. Si le message indique à l'agent de s'arrêter, l'agent exécute quelques instructions de nettoyage et s'arrête après s'être désinscrit auprès des autres agents. Lorsqu'un problème de communication est détecté, l'agent a également la possibilité de basculer en mode dégradé, dans lequel il peut choisir de s'arrêter progressivement pour assurer la sécurité du système, s'il ne parvient pas à rétablir la communication.

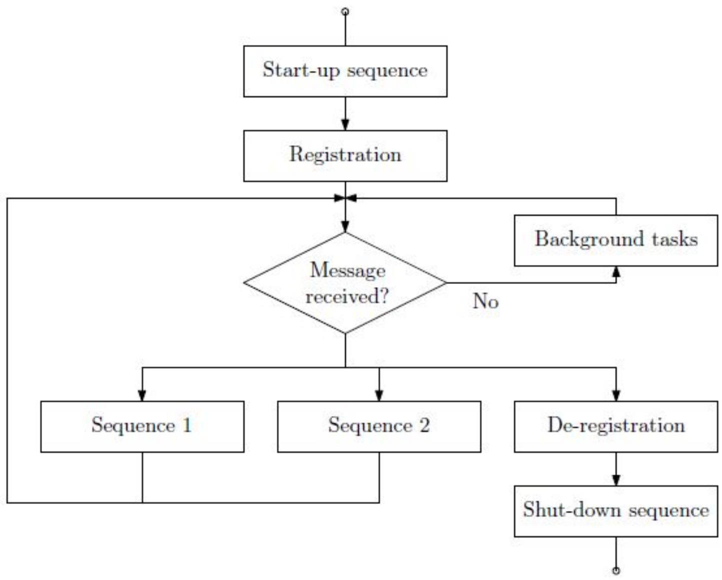

FIG. 5 – Diagramme générique de cycle de vie d'un agent, de sa création à son arrêt. Les séquences d'instructions sont choisies en fonction des interactions avec les autres agents.

2.2 Interactions entre les agents

Les interactions des agents dans notre SMA dépendent de l'évolution du système. En effet, deux situations se présentent : la modification de l'architecture et le fonctionnement normal du système.

2.2.1 Changement dans l'architecture SMA

Lors du changement de l'architecture du système, par exemple, au démarrage ou lorsqu'une turbine est ajoutée, la structure de SMA a besoin de s'adapter. Dans le premier cas, l'agent *GUI* et l'agent du système de gestion sont lancés, initialisés et paramétrés par l'opérateur. Le système attend alors que les agents micro-réseau s'inscrivent. Le fonctionnement du système peut alors commencer. Pour un agent type turbine, il suit le processus décrit ci-après (voir les trois premiers messages de la figure 6) :

1. L'agent de la turbine est créé, comme *turbine*, et son système de contrôle est démarré.

2. Pour son initialisation, il charge ou mesure les caractéristiques de la turbine (modèle, puissance max en sortie, type de carburant, etc.).

3. Il s'inscrit auprès de l'agent *annuaire*, pour permettre aux autres agents de le trouver dans le cas où ils auront besoin d'une turbine.

4. Il s'enregistre ensuite auprès de l'agent *SCADA*, et attend une demande pour transmettre des informations sur son état actuel. L'agent a alors atteint son état normal de fonctionnement.

5. L'agent *SCADA* enregistre enfin le nouvel agent auprès de l'agent *dispatch*. La turbine est désormais pleinement opérationnelle.

De même, quand il faut déconnecter un composant, soit par l'opérateur ou par le système, l'agent correspondant commence par se désinscrire auprès de l'agent *annuaire* et des bases de données des autres agents, de sorte qu'ils ne tentent plus de communiquer avec lui. Une série d'instructions relatives à l'arrêt de l'agent, par exemple l'enregistrement des données mesurées, sont ensuite exécutées avant son retrait effectif du système. Dans le cas où l'agent doit être temporairement déconnecté, l'agent peut être suspendu en passant à un mode *veille*. En cas de besoin, l'agent peut "rebasculer" en mode *normal*. Cette capacité du système à s'adapter aux changements dans sa structure lui permet de s'adapter à une variété très large de micro-réseaux, avec un nombre variable de composants (turbine, piles à combustion, etc.).

2.2.2 Fonctionnement normal du système

Dans un fonctionnement normal du système, c'est-à-dire lorsque la structure du système ne change pas, le système exécute les mêmes instructions en permanence à une fréquence donnée choisie par l'opérateur (voir les étapes 6 à 14 de la figure 6) :

6. L'agent *dispatch* demande à l'agent *SCADA* de lui transmettre les mesures sur l'état actuel de l'ensemble du système.

7. L'agent *SCADA* demande à chaque agent, inscrit dans sa base de données, de lui transmettre les dernières informations de son état, par exemple les mesures en cours.

8. Chaque agent, réalise ses mesures et met à jour ses caractéristiques.

9. Les données mises à jour sont ensuite envoyées à l'agent *SCADA*, qui centralise les mesures et met à jour sa propre base de données.

10. L'agent *SCADA* envoie les données demandées à l'agent *dispatch*.

11. L'agent *dispatch* calcule les points de fonctionnement optimaux.

12. Ces points de fonctionnement sont enfin envoyés à l'agent *SCADA*, qui se charge de les transmettre à leurs destinataires respectifs.

13. Les résultats sont également envoyés à l'agent *GUI*, qui les affiche.

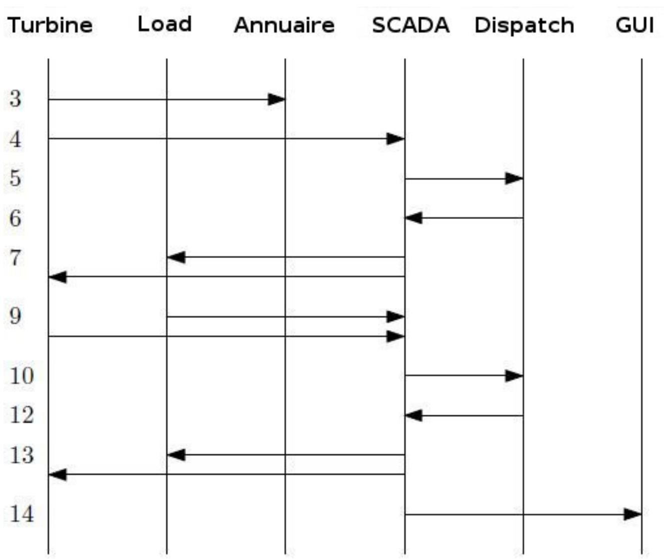

FIG. 6 – Durant leurs opérations, les agents interagissent les uns avec les autres en échangeant des informations par le biais des messages. Chaque flèche décrit la direction de transmission des messages entre deux agents. Cet exemple est limité à deux composantes, une turbine et la charge, pour faciliter la compréhension. Les étapes 3 à 5 se réfèrent à la section 2.2.1 et les étapes 6 à 13 se réfèrent à la section 2.2.2.

3 Optimisation du fonctionnement des micro-réseaux et des centrales

L'efficacité du système est quant à elle obtenue grâce à des algorithmes d'optimisation basés sur des métaheuristiques. Ils sont au cœur des deux couches supérieures du système de gestion de la figure 4 et leur donnent leur intelligence.

3.1 Algorithmes d'optimisation

3.1.1 Algorithme *MPSOM* (Metropolis Particle Swarm Optimization with Mutation Operator)

Dans cette sous-section, nous présentons l'algorithme hybride que nous avons proposé dans (Idoumghar *et al.*, 2010) et qui est noté *MPSOM*. Cet algorithme se base sur l'algorithme PSO auquel on incorpore un opérateur de mutation et la règle de *Metropolis* afin de sortir d'un optimum local. Le principe de *MPSOM* fonctionne comme illustré dans l'algorithme 1, où :

Algorithm 1 Algorithme hybride *MPSOM*

```
Initialize swarm_size particles
Evaluate (Swarm)
stop_criterion ← maximum number of function evaluations
nonImprove ← 0
Initialize inertia factor w ← w_0
Initialize initial temperature T ← T_0
while Not stop_criterion do
  Sort (Swarm)
  if nonImprove < k then
    for each particle i ← 1 to swarm_size do
      Accept (cbest, X, T)
      Update velocity according to Equation (2)
      Enforce velocity bounds
      Update particle position according to Equation (5)
      Enforce position bounds
    end for
  else
    {Mutation operator}
    nonImprove ← 0
    for each particle i ← 1 to swarm_size do
      Initialize its velocity to maximum velocity allowed
    end for
  end if
  Evaluate (Swarm)
  if there is no improvement of global best solution then
    nonImprove ← nonImprove + 1
  else
    Update global best solution
    nonImprove ← 0
  end if
  Update inertia factor w by using Equation (4)
  Update (T)
  Update (stop_criterion)
end while
```

- **Particule** : chaque particule (solution) est représentée par sa :
  - position courante $X \in S$ représentée par ses $n > 0$ composants, $i.e.$ $X = (x_1, x_2, ..., x_n)$ où n représente la dimension du problème d'optimisation à résoudre.
  - meilleure solution obtenue précédemment (appelée $cbest$).

– vélocité v qui correspond au taux de changement de la position.
- **Essaim initial** : correspond à la population de particules qui vont évoluer. Chaque particule x_i est initialisée avec une valeur uniformément tirée entre les bornes inférieure et supérieure de l'intervalle définissant le problème d'optimisation.
- **Fonction *Evaluate*** : correspond à la fonction objectif que l'algorithme *MPSOM* doit minimiser.
- **Tri** : toutes les particules de l'essaim sont triées dans l'ordre décroissant de leur fonction objectif.
- **Fonction *Accept*** : $Accept(cbest, X, T)$ est déterminée par la probabilité d'acceptation donnée par l'équation 1, qui est la probabilité d'accepter la position courante d'une particule comme étant son $cbest$.

$$p = \begin{cases} 1 & \text{si } f(X) \leq f(cbest) \text{ ou} \\ & \frac{1}{2} rand \times (1 + e^{\frac{f(X)-f(cbest)}{T}}) < 1 \\ 0 & \text{sinon} \end{cases} \tag{1}$$

  Dans l'équation 1, T est la température. Elle joue le même rôle que la température dans la méthode du recuit simulé. $rand$ est un nombre aléatoire indépendamment généré dans l'intervalle $[0, 1]$.
- **Mise à jour de la vélocité** : nous proposons ici, un voisinage topologique qui est déterminé dynamiquement durant le processus de recherche selon la fonction objectif de l'essaim. Après avoir trié la population de l'essaim pour chaque $i^{\text{ème}}$ particule, un ensemble N_i de ses voisins est défini comme : $N_i = \{particule\ k\ /\ k \geq i\ et\ k \leq swarm\_size\}$. Ensuite, afin d'ajuster la vélocité de la $i^{\text{ème}}$ particule, nous utilisons l'équation 2 suivante :

$$\begin{aligned} v_{ij} &= \omega \times v_{ij} + c_1 \times rand \times (cbest_{ij} - x_{ij}) \\ &+ min(V_{max}, \sum_{k=i}^{swarm\_size} \frac{cbest_{kj} - x_{ij}}{k}) \end{aligned} \tag{2}$$

  Dans l'équation 2, le composant social de la $i^{\text{ème}}$ particule est calculé comme étant la moyenne pondérée des meilleures particules dans N_i. En procédant de la sorte, une particule n'est pas seulement influencée par la meilleure solution globale, mais elle ajuste sa vitesse en fonction de la moyenne pondérée des solutions qui sont meilleures que les siennes. Cela signifie simplement qu'il est mieux pour la particule de suivre un groupe de particules plutôt que de suivre une seule particule.
- **Mise à jour de la température** : afin d'éviter à l'algorithme d'être coincé dans un minimum local, le taux de réduction de la température doit être lent. Nous utilisons pour cela l'équation 3 où $i = 0, 1, ...$ et $\gamma = 0.99$.

$$T_{i+1} = \gamma\ T_i \tag{3}$$

- **Mise à jour du facteur d'inertie** : le facteur d'inertie est défini par l'équation 4 où :

$$w_k = w_0 \times \left(1 - \frac{T_0 - T_k}{T_0}\right) \tag{4}$$

  - $w_0 = 0.9$ correspond à la valeur du poids au démarrage de l'algorithme. Notons qu'avec de faibles valeurs de w, la région de recherche de l'algorithme peut se trouver autour de la meilleure solution, alors qu'avec des valeurs élevées de w, l'algorithme peut améliorer l'exploration (recherche globale).
  - T_0 est une température initiale et T_k représente la température à la $k^{\text{ème}}$ itération.

- **Opérateur de mutation** : s'il n'y a pas d'améliorations de la meilleure solution globale pendant les dernières K itérations, cela signifie que l'algorithme est coincé dans un optimum local. Afin d'en échapper, notre algorithme *MPSOM* utilise l'opérateur de mutation en se basant sur l'idée suivante : en attribuant une vélocité maximale permise à chaque particule, l'équation 5 assure que toutes les particules vont sortir de l'optimum local et *MPSOM* peut avoir une large capacité d'exploration.

$$x_{i+1} = x_i + v_{i+1} \tag{5}$$

3.1.2 Algorithme ICA (Imperialist Competitive Algorithm)

ICA (Imperialist Competitive Algorithm) est une approche récente développée par (Atashpaz-Gargari & Lucas, 2007). Elle tente de reproduire le comportement des empires. Il met en place un mécanisme de compétition afin d'affaiblir puis de supprimer les empires les moins puissants (c'est-à-dire les empires trouvant les moins bonnes solutions).

Dans cette section nous allons décrire le fonctionnement de l'algorithme ICA. Imperialist Competitive Algorithm (ICA) est une méthode qui utilise l'impérialisme et la compétition impérialiste comme source d'inspiration. L'algorithme 2 décrit le fonctionnement de l'algorithme ICA dont les principales étapes sont :

Algorithm 2 Imperialist Competitive Algorithm

```
Initialize and Evaluate the empires
while Stop condition is not satisfied do
    Move the colonies toward their relevant imperialist
    Revolution. Make some changes in the characteristics of some of the countries
    if there is a colony in an empire which has lower cost than that of imperialist then
        Exchange the positions of that colony and the imperialist
    end if
    Compute the total cost of all empires
    if the distance between two empires is less than UnitingThreshold then
        Unite the two empires
    end if
    Imperialistic competition
    if there is an empire with no colonies then
        Eliminate this empire
    end if
end while
```

– **Initialisation :** dans cet algorithme chaque individu de la population représente un pays, la première étape est de les initialiser puis de les évaluer. Les meilleurs pays sont sélectionnés et deviennent les impérialistes et ceux qui ne le sont pas deviennent les colonies des impérialistes. Les colonies sont ensuite réparties sur les empires en fonction de leur puissance.

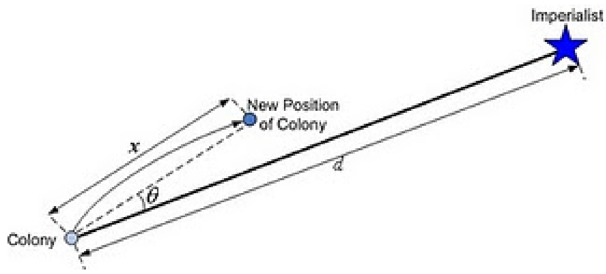

FIG. 7 – Déplacement d'une colonie

– **Mouvement :** après avoir réparti les colonies entre les différents impérialistes, chaque colonie se déplace en direction de son impérialiste. La figure 7[2] illustre ce mouvement, d est la distance entre la colonie et l'impérialiste. θ et x sont des nombres aléatoire. Le nombre θ, tiré aléatoirement, permet de faire dévier légèrement la colonie. Après chaque mouvement d'une colonie, il se peut que le coût d'une colonie soit inférieur au coût de son impérialiste, dans ce cas, la colonie devient l'impérialiste.

– **Coût total d'un empire :** le coût total d'un empire dépend du coût de l'impérialiste et de ses colonies. La formule (6) donne l'expression qui permet de calculer le coût total d'un empire n.

$$TC_n = Cost(Imprialiste_n) + \xi \cdot mean(colonies\ of\ empire_n) \tag{6}$$

avec ξ est un nombre inférieur à 1.

– **Compétition impérialiste :** la compétition impérialiste permet à un empire plus fort de s'emparer d'une colonie de l'empire le plus faible. Cette compétition permet ainsi d'augmenter le pouvoir des empires forts et d'éliminer les empires faibles.
La figure 8 illustre cette compétition, l'empire 1 le plus faible est sur le point de perdre une colonie au profit d'un des empires. Chaque empire a une probabilité de possession P_i calculée à partir de sa puissance.

[2]Figure extraite de http ://www.atashpaz.com/p/imperialist-competitive-algorithm-ica.html

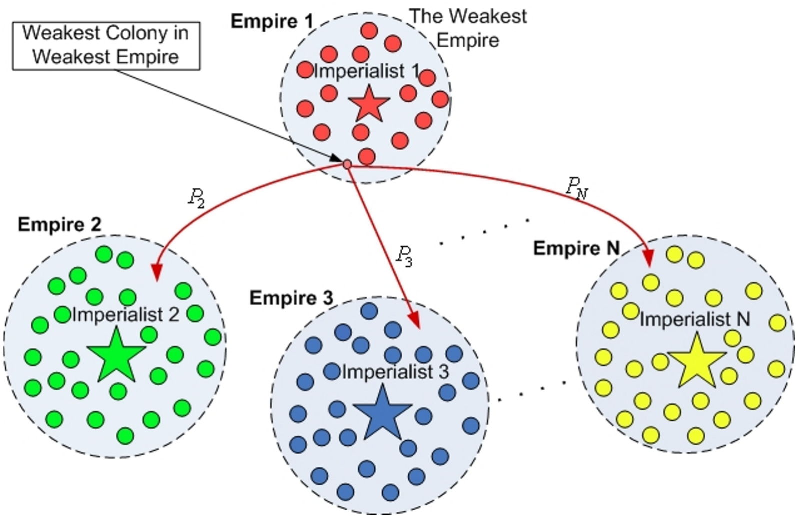

FIG. 8 – Principe de la compétition des empires : l'empire le plus puissant a plus de chance de posséder la plus faible colonie de l'empire le plus faible.

$$P_{p_n} = \left| \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \quad (7)$$

avec $NTC_n = TC_n - \max_i TC_i$, TC_n et NTC_n correspondent respectivement au coût total et au coût total normalisé du $n^{\text{ème}}$ empire.

Le mouvement des colonies vers l'impérialiste et la compétition permet de voir émerger un état où il n'y a plus qu'un seul empire et où l'algorithme a trouvé idéalement une solution proche de la solution optimale.

3.2 Problème du *dispatching*

La résolution du problème du *dispatching* optimal est réalisée à l'aide des deux algorithmes $MPSOM$ (Roche *et al.*, 2011b) et (Roche *et al.*, 2011a) précédemment détaillés. Le réseau évoluant en permanence, il s'agit donc d'un problème dynamique, contraint et potentiellement à objectifs multiples. Une solution correspond à un vecteur composé des puissances des composants que l'ont souhaite calculer et prendre en compte. L'espace d'optimisation pour chaque dimension est défini par l'agent lui-même, en fonction de ses propres contraintes.

L'objectif principal est généralement le coût de fonctionnement, que l'on cherche à minimiser. Ce coût correspond notamment au coût du combustible et aux achats et ventes d'énergie sur le réseau principal. On cherche donc à minimiser :

$$f(t) = \sum_{i=1}^{n_{gen}} c(P_{gen,i}(t)) + \sum_{i=1}^{n_{enr}} c(P_{enr,i}(t)) + \sum_{i=1}^{n_{sto}} c(P_{gen,i}(t)) + \sum_{i=1}^{n_{res}} c(P_{res,i}(t)) \quad (8)$$

Soumis à :

$$\sum_{i=1}^{n_{gen}} P_{gen,i}(t) + \sum_{i=1}^{n_{enr}} P_{enr,i}(t) + \sum_{i=1}^{n_{sto}} P_{gen,i}(t) + \sum_{i=1}^{n_{res}} P_{res,i}(t) = \sum_{i=1}^{n_{dem}} P_{dem,i}(t) \quad (9)$$

L'équation 8 correspond au coût total de fonctionnement, qui est la somme des coûts c de chaque composant du réseau, dont les puissances sont notées avec les indices *gen* (générateurs), *enr* (énergies renouvelables), *sto* (stockage), *res* (réseau) et *dem* (demande). Quant à 9, elle transcrit l'équilibre entre offre et demande, qui doit être en permanence maintenu, et est donc une des contraintes principales à prendre en compte. D'autres contraintes physiques ne sont pas détaillées ici mais peuvent être consultées dans (Roche *et al.*, 2011b).

Le réseau étant en constante évolution, ce problème est résolu en continu, toutes les minutes ou secondes, soit quasiment en temps-réel. Les équations 8 et 9 sont construites dynamiquement à partir des mesures reçues et leurs solutions, comportant des points de fonctionnement, sont renvoyées aux agents qui peuvent décider ou non de les appliquer.

Les émissions de gaz à effet de serre étant de plus en plus encadrées, les émissions des générateurs classiques, par exemple à gaz, peuvent être considérées comme un second objectif. Moyennant une adaptation[3] de $MPSOM$ lui permettant de prendre en compte plusieurs objectifs et de retourner un ensemble de solutions dites Pareto-optimales, ce critère peut être intégré. Il requière toutefois de choisir une solution suivant un critère à définir par l'opérateur du système.

Un ensemble de règles permet par ailleurs d'implémenter la stratégie de gestion définie par l'opérateur du système : conditions d'utilisation du stockage, réserves de puissance, etc.

Enfin, l'algorithme de *dispatching* peut être combiné à un système expert[4], baptisé ASSS (*Automatic Start & Stop System*), qui décide quand et combien de sources contrôlables doivent être démarrées ou arrêtées pour minimiser encore les coûts.

3.3 Problème de la *planification*

Le problème de la *planification* est une extension du problème du dispatching. Au lieu de considérer simplement les points de fonctionnement à un temps t donné, on considère des profils sur plusieurs heures voire jours. Cela permet de déterminer quand démarrer telle ou telle source, quand il est optimal de stocker de l'énergie en prévision d'un pic de demande, quand il est préférable de délester une partie de la charge, etc. La résolution de ce problème passe d'une part par l'obtention et l'utilisation de prévisions, aussi précises que possible, de la production intermittente, de la demande et des prix d'achat et de revente d'énergie, et d'autre part sur une légère modification de l'algorithme $MPSOM$. En effet, celui-ci est adapté pour que chaque solution comporte chaque point de fonctionnement à trouver à chaque temps t de la période considérée. La solution comportera donc des variables binaires déterminant si une unité est à démarrer ou non. D'autres contraintes, telles que les durées minimales de fonctionnement ou d'arrêt des sources thermiques, sont également à ajouter.

La dimension du problème étant alors bien plus grande, il est également bien plus long à résoudre. Toutefois, il est résolu en parallèle de celui du *dispatching*, et ne nécessite pas d'être résolu en temps réel (une à deux résolutions par jour peuvent suffire, suivant la précision des prévisions).

4 Exemples d'applications

Deux configurations de réseau sont utilisées pour tester les performances de notre approche. La première est composée d'un micro-réseau où la stabilité de nos algorithmes d'optimisation sera testée et la seconde représente une centrale électrique composée de piles à combustible, où l'optimisation portera sur le coût total de la simulation. Les deux tests sont effectués sur une période de cinq jours et sont basés sur les profils de charge réel [5]. Les tests ont été effectués à plusieurs reprises et retournent des résultats très similaires. Côté implémentation, le système présenté dans les parties précédentes a été implémenté en Java sur la base du middleware pour SMA JADE, de façon à créer un simulateur autonome, entièrement paramétrable et adaptable à divers problèmes et structures.

4.1 Micro-réseau

Le premier exemple que nous présentons est celui d'un micro-réseau, où des piles à combustible, des panneaux photovoltaïques et des éoliennes sont installés. Une connexion au réseau de distribution est également disponible. Un profil de charge d'une durée de 5 jours est adapté à partir d'un relevé fourni par un distributeur d'électricité [6]. Notons qu'aucune modification n'a été apportée au système de gestion, à part lui

[3] travail en cours

[4] travail en cours

[5] http ://www.sce.com/AboutSCE/Regulatory/loadprofiles

[6] www.sce.com/AboutSCE/Regulatory/loadprofiles

indiquer les caractéristiques du nouveau micro-réseau. Les résultats, visibles sur la figure 9 [7], montrent que le système a été capable de répondre à la demande tout au long de la simulation, malgré la forte intermittence de la production renouvelable.

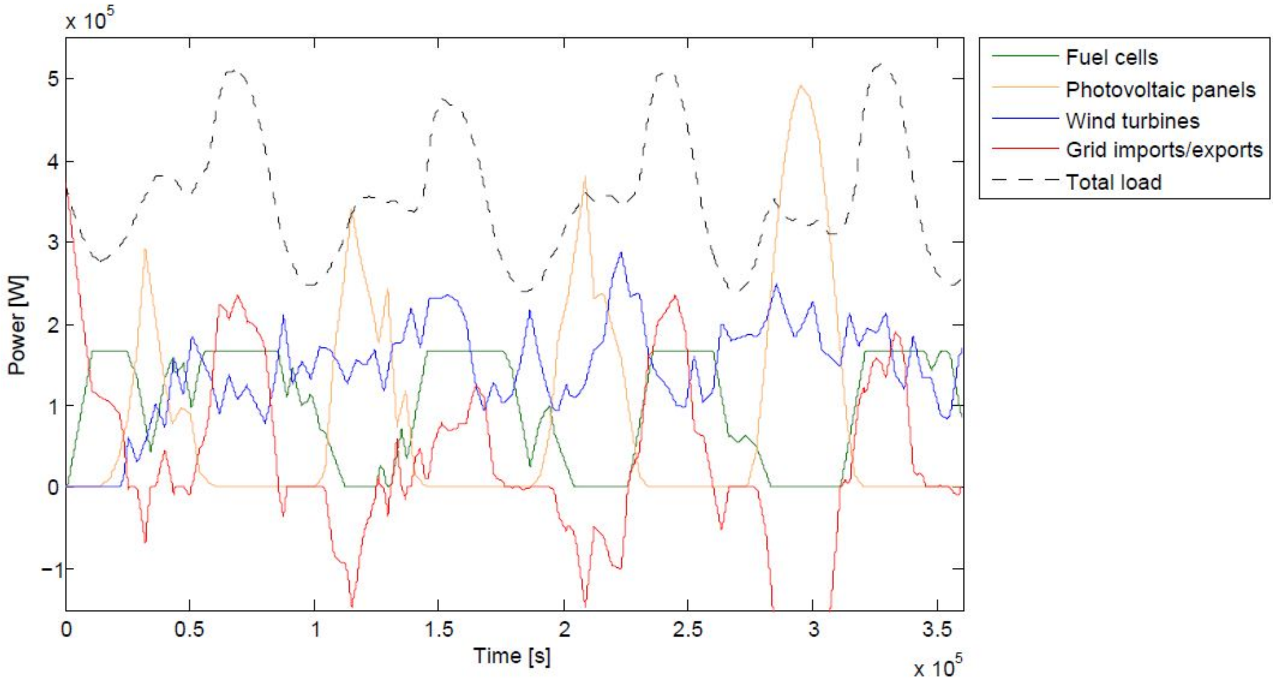

FIG. 9 – Résultats de la simulation pour le micro-réseau, montrant que le système est capable de maintenir l'équilibre production-demande même avec une forte intermittence de la production.

Algorithme	Unité	MPSOM	DE	ICA
Coût Total	€	1075	1155	1088
Déséquilibre moyen	W	-0.0175	0.1344	0.1314
Temps moyen	ms	8.126	4.075	257.0

TAB. 1 – Comparaison des résultats pour le micro-réseau

Le tableau 1 présente les résultats obtenus par les algorithmes $MPSOM$, DE et ICA. L'analyse de ce tableau montre que l'algorithme $MPSOM$ obtient le meilleur coût suivi de l'algorithme ICA.

4.2 Centrale de production

Un second exemple correspond à la gestion de l'énergie appliquée à une centrale de production pouvant comporter différents composants. Ici, on considère qu'elle comporte six piles à combustible, alimentées en hydrogène. Chaque pile possède son rendement propre, dépendant de son historique d'utilisation et de son âge. Le profil de charge utilisé est similaire au précédent [8]. Les résultats du *dispatching* de la figure 10 montrent que l'optimisation parvient à correctement répartir l'utilisation des piles en fonction de leurs caractéristiques. On observe en effet que les piles les plus efficaces sont plus utilisées que les autres (cf. légende : une valeur plus faible indique une meilleure efficacité).

L'analyse du tableau 2 montre les gains obtenus en utilisant trois algorithmes d'optimisation. On peut remarquer que l'algorithme ICA minimise le coût de fonctionnement à 2119 €, contre 4097 € si l'on ne fait que diviser la charge totale par le nombre de piles. Cependant, comme avec le test précédent, il est

[7] Le système a recours au réseau uniquement lorsqu'il ne peut pas satisfaire la demande en utilisant les moyens d'énergies qu'il commande.

[8] l'Ensoleillement et la vitesse du vent sont extraits de : www.unige.ch/cuepe/html/meteo/donnees-csv.php

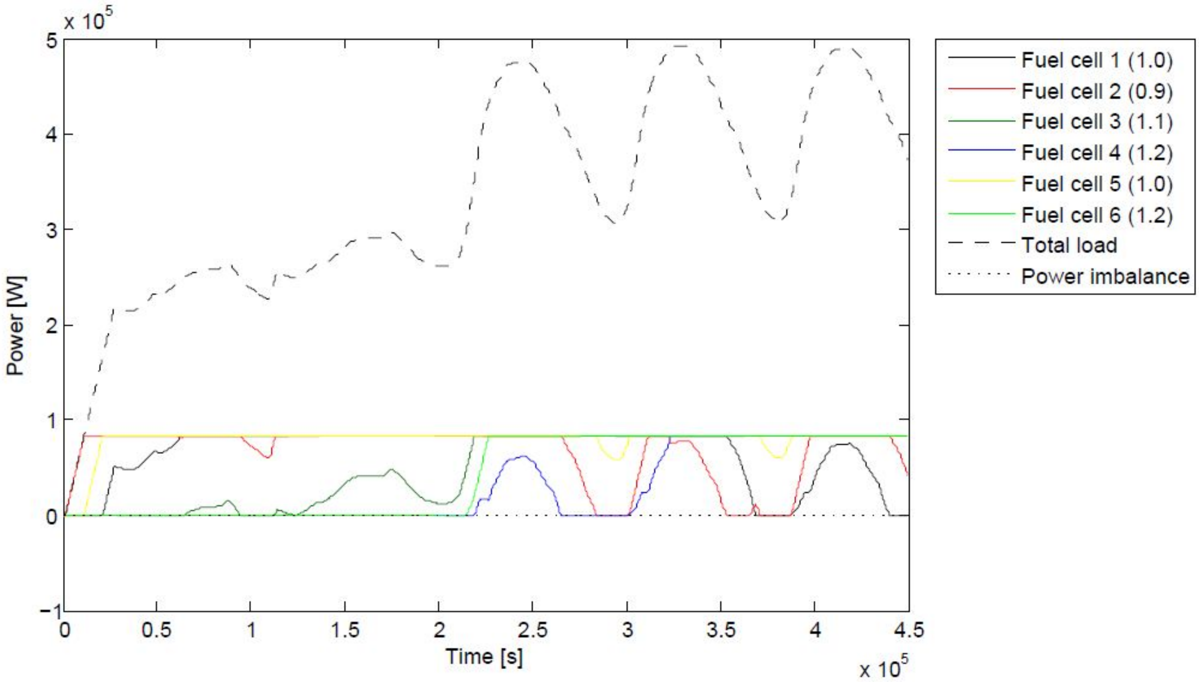

FIG. 10 – Résultats de la simulation pour le micro-réseau avec l'optimisation, ici par ICA (Roche *et al.*, 2011a)). Le résultat obtenu est toutefois très poche de celui obtenu par $MPSOM$ (Roche *et al.*, 2011b).

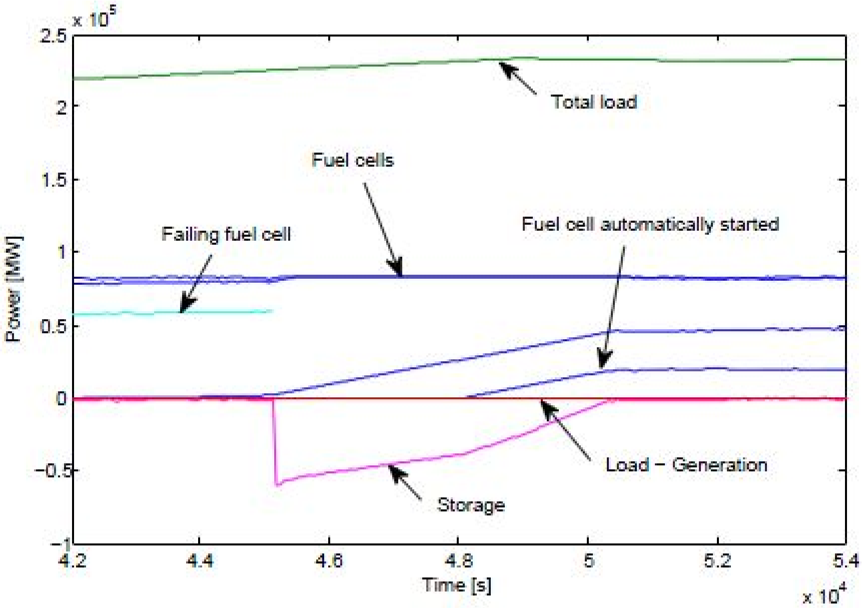

FIG. 11 – Dans cette figure, la puissance est achetée sur le réseau afin de compenser la défaillance de la batterie. La puissance achetée via le réseau est comptée comme négatif.

aussi plus lent que les autres algorithmes. Differential Evolution (DE) est particulièrement rapide, mais est également assez imprécise, bien que le déséquilibre ne représente que 10^{-4} % de la charge totale maximale. Il convient également de noter que le déséquilibre de $MPSOM$ est presque toujours proche de zéro. Par conséquent, il sera le plus privilégié.

D'autres tests montrent que le système est capable de réagir correctement lorsqu'une panne survient, en

Algorithme	Unité	EqD	MPSOM	DE	ICA
Coût Total	€	4097	2165	2160	2119
Déséquilibre moyen	W	≈ 0	-0.099	-2.286	-2.888
Temps moyen	ms	< 1	11.82	8.391	454.8

TAB. 2 – Comparaison des résultats pour la centrale de production

utilisant au mieux les possibilités offertes par les différents éléments du réseau. La figure 11 illustre la réaction du système face à la défaillance d'une batterie.

5 Conclusion

Les travaux présentés dans cet article ont montré qu'un système de gestion d'énergie à la fois flexible et efficace a été développé, en combinant les avantages des systèmes multi-agents et des métaheuristiques. De nombreuses applications sont possibles, et plusieurs sont en cours d'étude : gestion de centrales à gaz avec prise en compte des émissions, trading entre micro-réseaux, etc. Les perspectives incluent la prise en compte de nouveaux éléments (véhicules électriques, utilisateurs, gestion de la demande, pertes en ligne, etc.) ainsi que la simulation en temps réel des réseaux interconnectés.

Références

ATASHPAZ-GARGARI E. & LUCAS C. (2007). Imperialist competitive algorithm : An algorithm for optimization inspired by imperialistic competition. In *Proceedings of the IEEE Congress on Evolutionary Computation*, p. 4661–4667.

FERBER J. (1999). *Multi-Agent Systems : An Introduction to Artificial Intelligence*. Addison-Wesley.

HILAIRE V., KOUKAM A. & RODRIGUEZ S. (2008). An adaptative agent architecture for holonic multi-agent systems. *Journal of ACM Transactions on Autonomous and Adaptive Systems*, **3**(1), 1–24.

IDOUMGHAR L., IDRISSI-AOUAD M., MELKEMI M. & SCHOTT R. (2010). Metropolis particle swarm optimization algorithm with mutation operator for global optimization problems. In *Proceedings of IEEE-ICTAI'2010 : 22th International Conference on Tools with Articial Intelligence*, p. 35 – 42, Arras, France.

MAZIGH B., HILAIRE V. & KOUKAM A. (2011). Formal specification of holonic multi-agent systems : Application to distributed maintenance company. In *Proceedings of the 9th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*.

ROCHE R., BLUNIER B. & MIRAOUI A. (2010). Multi-agent systems for grid energy management : A short review. In *Proceedings of the 36th Annual Conference of the IEEE Industrial Electronics Society (IECON 2010)*.

ROCHE R., IDOUMGHAR L., BLUNIER B. & MIRAOUI A. (2011a). Imperialist competitive algorithm for dynamic optimization of economic dispatch in power systems. In *Proceedings of International Conference on Artificial Evolution (EA 2011)*.

ROCHE R., IDOUMGHAR L., BLUNIER B. & MIRAOUI A. (2011b). Optimized fuel cell array energy management using multi-agent systems. In *Proceedings of the IEEE International Conference on Industrial Applications (IAS 2011)*.

SIMOES M., ROCHE R., KYRIAKIDES E., MIRAOUI A., BLUNIER B., MCBEE K., SURYANARAYANAN S., NGUYEN P. & RIBEIRO P. (2011). Smart-grid technologies and progress in europe and the usa. In *Proceedings of the IEEE Energy Conversion Congress & Exposition (ECCE 2011)*.

Remerciments

Les auteurs souhaitent remercier les rapporteurs pour leurs précieuses remarques et suggestions qui ont permis d'améliorer la qualité de cet article.