Negotiation Game for Joint IT and Energy Management in Green Datacenters

Minh-Thuyen Thi[a], Jean-Marc Pierson[a,*], Georges Da Costa[a], Patricia Stolf[a], Jean-Marc Nicod[b], Gustavo Rostirolla[c], Marwa Haddad[b]

[a] *IRIT, University of Toulouse, CNRS, INPT, UPS, UT1, UT2J, France*
[b] *FEMTO-ST, CNRS, Univ. Bourgogne Franche-Comte, UTBM, France*
[c] *LAPLACE, University of Toulouse, CNRS, INPT, UPS, France*

Abstract

As the power demand of datacenters is increasing sharply, a promising solution is to power datacenters locally by renewable energies. However, one of the main challenges when operating such green datacenters is to conciliate the intermittent power supply and the power demand. To deal with this problem, we view the green datacenter as two sub-systems, namely, Information Technology (IT) sub-system which consumes energy, and electrical sub-system which supplies energy. The objective is to find an efficient trade-off between the power demand and power supply, respecting the operational requirements of both sub-systems (i.e., the requirements on *utility*, or monetary gain, which includes monetary revenue and monetary cost). First, we analyze the problem by a black-box approach. In this approach, the models of the two sub-systems are unknown to each other, and the two sub-systems negotiate by exchanging their power preferences. However, we found that the black-box approach cannot guarantee stable solutions in term of execution time and *generational distance* (which is the distance between a solution and the *Pareto front*). Then we introduce a semi black-box approach, in which the two sub-systems are modeled as the

*Corresponding author.
Email addresses: minh-thuyen.thi@irit.fr (Minh-Thuyen Thi), jean-marc.pierson@irit.fr (Jean-Marc Pierson), dacosta@irit.fr (Georges Da Costa), patricia.stolf@irit.fr (Patricia Stolf), jean-marc.nicod@femto-st.fr (Jean-Marc Nicod), gustavo.rostirolla@laplace.univ-tlse.fr (Gustavo Rostirolla), marwa.haddad@femto-st.fr (Marwa Haddad)

buyer and the supplier in a buyer-supplier negotiation game. We propose an algorithm that allows the buyer and supplier to negotiate, seeking for an efficient trade-off between the power demand and power supply. The analytical results show that the semi black-box algorithm converges to equilibrium, and these results are then confirmed by experimental results. We conduct the experiments by implementing a middleware of a datacenter powered by renewable energies. The experimental results show that the semi black-box algorithm improves significantly the stability, quality of service (QoS) and utility of the datacenter, compared to other algorithms. In term of stability, compared to the black-box algorithm, the semi black-box algorithm reduces the standard deviation of execution time and generational distance by 23 and 27 times, respectively. In term of QoS and utility, the semi black-box algorithm outperforms the algorithms that do not consider joint IT-energy management, as well as the algorithms that do not utilize a semi black-box design.

Keywords: Datacenter, Renewable energy, Game theory, Negotiation,

---

1. Introduction

As the demand for cloud services has been growing over recent years, the energy consumption of datacenters is increasing rapidly. A number of studies have showed intensive data and reports on this increase Van Heddeghem et al. (2014), Gill and Buyya (2018), Andrae and Edler (2015), Shehabi et al. (2016). The consumed electricity of datacenters worldwide can reach 8000 billion kilowatt hours (kWh) in 2030 if efficient control methods are not developed Andrae and Edler (2015). In the US, the datacenters consumed 100 billion kWh of electric energy in 2015, and this consumption is expected to be 150 billion kWh in 2022 Gill and Buyya (2018). On the other hand, the traditional/brown energy sources are becoming less preferable, due to economic and environmental concerns. One promising solution is to use renewable energies to locally power datacenters, avoiding both greenhouse gas emission and electricity distribution loss. We consider a green datacenter that is entirely supplied by renewable

sources (namely, wind turbines (WT) and photovoltaic panels (PV)), and storage devices (namely, batteries (BT), electrolyzers (EZ) and fuel cells (FC)). However, one of the main challenges of building this datacenter is to conciliate the intermittent energy production and the continuous operation of the datacenter. The production of renewable energies such as WT and PV highly depends on environmental conditions, so we need to coordinate this intermittent energy production with the energy demand, in order to guarantee a high quality of service (QoS) for cloud users.

To deal with this problem, we model the datacenter with one Negotiation Module (NM) connecting to two sub-modules, namely Information Technology Decision-support Sub-module (ITDM) and Power Decision-support Sub-module (PDM). The ITDM manages the scheduling of datacenter workload, while the PDM manages the scheduling of electrical sources. The scheduling of both ITDM and PDM are considered jointly, with the support of the NM. This NM manages the negotiation between the PDM's power supply and the ITDM's power demand. When the power supply and power demand are mismatched, the NM aims to find a compromise between them. We provide two generic models for the ITDM scheduler and PDM scheduler; in this way, the proposed negotiation algorithms can work with any scheduler that is implemented based on these generic models.

As a straightforward approach toward a distributed design, we first propose a black-box negotiation algorithm, named Scheduling Based Negotiation (SAN). We define that a *power profile* is a set of power values associated with a time interval. In this black-box approach, each Decision-support Sub-module (DM) learns about the other DM through exchanging power profiles. These profiles, called *hints*, allow each DM to learn about the preferred power of the other DM. Each DM is expected to gradually propose more relevant profiles based on the similarity to the hints. However, we found that the black-box approach cannot guarantee stable solutions in term of execution time and *generational distance*. In brief, *generational distance* is the distance between a solution and the *Pareto front*, where *Pareto front* in a multi-objective problem is the boundary defined

by the set of non-dominated solutions. Then, we show that a semi black-box approach is more relevant to deal with this kind of problem. We model the problem as a buyer-supplier negotiation game, and based on this game, we propose a negotiation algorithm named Game Theory Based Negotiation (GAN). In this game, the ITDM and PDM are modeled to become two game players, named IT-Player and PD-Player. These two players negotiate with each other as an energy buyer and an energy supplier. Our goal is to find an efficient compromise between the players, respecting their operational requirements. The final solution is a mutually acceptable profile for both players.

We use a buyer-supplier model because this model reflects the buying-supplying relationship between the IT sub-system and the electrical sub-system. Moreover, in a negotiation problem, when two parties negotiate on a common resource, each party should consider its willingness when compromising. This is because a same amount of resource may be beneficial differently to each party. This property can be addressed by the *pricing* process in a buyer-supplier partnership.

We utilize a semi black-box model in order to retain the benefits of both black-box and non-black-box approach. In a semi black-box approach, the IT-Player and PD-Player are modeled independently, rather than integrally as in a centralized approach. However, unlike the black-box approach, the players can exchange more specific information to learn about the direction to negotiate. On the other hand, our problem is a large-timescale problem, in which the decision can be made every several days. However, we found that a semi black-box approach is more beneficial than a centralized approach. Firstly, each player is not required to gather a lot of information from the other player. The independence between two players facilitates the design process and reduces the involvement in case of modifying one player. Secondly, we propose to run the negotiation algorithm regularly in a smaller timescale, e.g., every 6 hours, even when there is no request from the players. In this way, the negotiation algorithm can be run in an overlap manner, i.e., it is run every 6 hours, to find a negotiation solution for 3 days. Finally, a decentralized approach facilitates the

design of a *multi-timescale negotiation*, which is one of our future works. In this *multi-timescale negotiation*, we first find a long-term solution, then based on this solution, we find multiple short-term solutions, within the long-term period.

The proposed game is a sequential and alternate-move game, rather than a one-shot game. The players' strategies (i.e., scheduling solutions) do not belong to a deterministic space, so it is highly complex to solve the game analytically using a one-shot non-cooperative approach. Moreover, this sequential game has perfect information, since each player knows the decision taken by the other player. However, the game is neither a complete nor incomplete information game, because even though the players know the rules of the game (e.g., *pricing* information), the players' payoffs are not common knowledge.

The contributions of this research are as follows.

- We model the problem of joint IT and energy management for datacenters entirely powered by renewable energies, then analyze that problem using a black-box and a semi black-box approach. We propose two generic models for the ITDM and PDM (section 3), then any scheduler that is implemented based on this generic model can work with the negotiation algorithms.

- After showing the instability of the black-box approach (section 4.2), we propose a buyer-supplier negotiation game for the problem (section 5), then introduce a negotiation algorithm to solve the game (section 6 and section 7). We show that, analytically and experimentally, the proposed algorithm converges to equilibrium.

- We setup a middleware to verify the proposed algorithms, and to evaluate the performance of the whole system. We show that the proposed negotiation algorithm can achieve efficient trade-offs between the utilities (i.e., monetary gains) of the IT and electrical sub-systems.

2. Related works

Recently there are a number of studies about datacenters partially or entirely powered by renewable energy. Some of them consider only energy management, some others consider only IT management, and some others jointly consider IT and energy management. Some methodologies of the energy management problem are power source coordination Li et al. (2014), Sheme et al. (2016), and power provisioning Liu et al. (2015). Some methodologies of the IT management problem are IT job scheduling Goiri et al. (2015), Lei et al. (2015), Kassab et al. (2018), virtual machine migration Wang et al. (2015), shifting demand in time/demand response management Cioara et al. (2015), Paul et al. (2017), assigning IT jobs to computational resources, Chonglin et al. (2015), Gu et al. (2015), and Dynamic Voltage and Frequency Scaling (DVFS) Wu et al. (2014). There is also some literature that proposes to combine multiple methodologies, e.g., scheduling IT jobs over time, assigning IT jobs to servers/VMs, controlling the states of servers/VMs, and assigning VMs to hosts Iturriaga and Nesmachnow (2016), Beldiceanu et al. (2017). From the perspective of *demand response*, we can also categorize those studies by temporal load balancing (e.g., shifting demand in time), spatial load balancing (e.g., assigning IT jobs to multiple computational resources), equipment state management (e.g., DVFS), and additional storage management.

However, there are not many studies that provide a detailed consideration of joint IT and energy management in green datacenters. Some studies that have certain levels of that consideration are Goiri et al. (2013), Goiri et al. (2014), Courchelle et al. (2018), Li et al. (2017), Roche et al. (2017), Caux et al. (2018), Haddad et al. (2019). Among them, the articles Goiri et al. (2013) and Goiri et al. (2014) are from a same research work; also, the articles Roche et al. (2017), Caux et al. (2018), and Haddad et al. (2019) are from a same research work.

Some studies of Goiri et al. focus on IT scheduling with respect to predicted renewable sources Goiri et al. (2011), or focus on developing a research platform for green datacenters Goiri et al. (2013), Goiri et al. (2014). The authors

proposed Parasol, which is a prototype of datacenter powered by solar energy, batteries, and net metering. The authors also introduced GreenSwitch, a scheduler for workload and energy sources. That research provides two main contributions: (1) the analysis of main trade-offs in the datacenters that are powered by solar and/or wind energy, and (2) the design of Parasol and GreenSwitch. The authors analyze three trade-offs, namely *grid-centric approach and self-generation approach*, *space and cost of solar energy*, and *space and cost of wind energy*. In Goiri et al. (2014), the GreenSwitch tries to minimize the cost of grid energy, with respect to the workload and the battery lifetime. The experiments of Parasol and GreenSwitch prove that an intelligent management of IT workload and energy source can reduce operation cost significantly. However, only solar panel is considered in Goiri et al. (2013) and Goiri et al. (2014). Moreover, the IT scheduling in that research is limited. At each time period, the scheduling algorithm selects which energy source (i.e., renewable, battery, and/or grid) and which storage medium (i.e., battery or grid) to use.

The research in Li et al. (2017), Courchelle et al. (2018), Aksanli et al. (2011) and Grange et al. (2018) also considers joint IT and energy management, though the energy management is limited. Li et al. Li et al. (2017) presented two methods to maximize the utilization of renewable energy in a small/medium-sized datacenter. The first method is an opportunistic scheduling, which suggests to run more jobs when renewable energy is available. The second method is to store renewable energy surplus to use later when the renewable energy supply is low. The experiments are setup with real-world job workload and solar energy traces. The authors also show that the proposed methods can reduce the demand for energy storage. However, the proposed methods have simplified the management of power sources. This management focuses on controlling the storage devices with respect to their characteristics, e.g., battery depth-of-discharge, battery charging rate limit. In Courchelle et al. (2018), the authors introduced a priority-based scheduling, considering the battery state and the renewable energy forecast. The scheduling uses genetic algorithm to allocate jobs, taking into account the storage capacity and the available solar energy. In general,

that research focuses on virtual machine scheduling, taking into consideration the solar panel production. In Aksanli et al. (2011), the authors also introduce an adaptive job scheduler with respect to the energy production forecast but of both solar and wind sources. The objective is to decrease the number of canceled or violated jobs, and increase the efficient usage of the green energy. Similar to Courchelle et al. (2018), the research in Aksanli et al. (2011) focuses on jobs scheduling with regard to renewable energy production. The jobs are either web services or batch jobs; and the authors assume that each server has one web services request queue, and one or more batch jobs slots. Web services are executed when there are available computing resources and there is enough brown energy to maintain these services. In Grange et al. (2018), Leo et al. proposed a heuristic IT scheduling algorithm for datacenter powered by renewable energies and the grid. The algorithm takes into account a limited knowledge of the power sources. Specifically, each source is supposed to have a predicted function that provides the estimation of the available energy over time. Then the algorithm takes this prediction as an input in order to schedule IT jobs for the datacenter.

The project *DATAZERO* [1] Roche et al. (2017), Caux et al. (2018), Haddad et al. (2019) is among the first studies that proposes the architecture of datacenters entirely powered by renewable energy, with a detailed consideration of joint IT and energy management. The works in Roche et al. (2017) introduced the design of the architecture, while the research in Caux et al. (2018) presented the management of power demand, and the technical report Haddad et al. (2019) presented the management of power supply. In this paper, we propose the negotiation between the power demand and power supply. As a part of the project, this paper presents the results of negotiation as well as the performance of the whole system.

[1] www.datazero.org

3. Datacenter model

3.1. Generic model

We define that a *power profile* (or *profile*) x is a set of power values, associated with a time window, denoted as $x = \{x_1, ..., x_T\}$, where T is the length of the time window. Depending on each specific context, a profile may have other names, e.g., *candidate profile*/*candidate* or *hint profile*/*hint*. Table 1 is the list of main notations used in our datacenter model.

We design a generic model for the DMs, then any scheduler whose implementation follows this model can work with the proposed negotiation algorithms. In the generic model, the ITDM is responsible for scheduling the IT workload in data center. Similarly, the PDM is responsible for scheduling the energy sources and storage devices. The DMs run scheduling algorithms to find multiple feasible scheduling solutions, and the negotiation module helps to find a compromised solution from those scheduling solutions (Fig. 1). The proposed negotiation algorithm is a turn-based approach, i.e., when the ITDM schedules, the PDM does not, and vice versa.

The two schedulers of the two DMs are described as follows.

- ITDM scheduler: this is an IT job scheduler; with each scheduling solution, this scheduler generates a corresponding power profile. Specifically, with respect to each scheduling solution, the scheduler computes the power profile needed to execute the scheduled jobs (including both service and batch jobs).
- PDM scheduler: this is a power sources scheduler; with each scheduling solution, this scheduler generates a corresponding power profile, which represents the power output from the electrical components (including energy sources and storage devices).

3.1.1. ITDM utility

The schedulers are able to output multiple feasible scheduling solutions. Each solution corresponds to a power profile, called *candidate*. Then, the nego-

Symbol	Name	Description
T	time window	T is a constant; T is equal to the number of time steps in each profile; in experiment, $T = 72$ (hours)
x	power profile	power profile has the form $x = \{x_1, ..., x_i, ..., x_T\}, i = 1, ..., T$, where x_i is the power value at i-th time step; the unit of power value is Watt (W)
$u(\cdot)$	utility	the utility of a DM; the utility in the black-box approach is normalized to [0, 1] and has no unit, whereas the utility in the semi black-box approach is the monetary gain, which has the unit *Euro* and is not normalized
$r(\cdot)$	revenue	the revenue of a DM; similar to the utility, the revenue in the black-box approach has no unit, whereas the revenue in the semi black-box approach has the unit *Euro*
$c(\cdot)$	cost	the cost of a DM; the cost in the black-box approach has no unit, whereas the cost in the semi black-box approach has the unit *Euro*
$J^{(b)}$	number of batch jobs	number of batch jobs that are being processed by ITDM
$J^{(s)}$	number of service jobs	number of service jobs that are being processed by ITDM
J	number of all jobs	number of service and batch jobs that are being processed by ITDM; $J = J^{(b)} + J^{(s)}$
$d(x, y)$	distance	distance between two profiles x and y; distance can be measured by Mean Square Error or Pearson correlation
ε	distance threshold	threshold used in stopping criteria of the negotiation algorithms; this threshold is set through experiment parameters

Table 1: Notations of datacenter model

tiation solution is selected based on the utility of the candidates. We explain in detail this selection in the next sections. For each profile x, the ITDM's utility $u(x)$ is computed based on the Amazon pricing [2] for on-demand instances, which is given as

$$u(x) = r(x) - c(x), \tag{1}$$

where the revenue $r(x) = g(x) - h(x)$, with $g(x)$ is *job execution time* multiplied by *instances cost*, and $h(x)$ is the service-level agreement (SLA) *violation compensation*; the cost $c(x)$ of the black-box and the semi black-box approach are given differently, which will be described in corresponding sections. The SLA violation compensations of IT batch job and service job are computed differently. First, with a set of $J^{(b)}$ batch jobs, we compute their average due date violation $v^{(b)}(x)$ as

$$v^{(b)}(x) = \frac{1}{J^{(b)}} \sum_{i=1}^{J^{(b)}} \frac{t_i^{(f)} - t_i^{(d)}}{t_i^{(d)} - t_i^{(s)}}, \tag{2}$$

where $t_i^{(s)}$, $t_i^{(d)}$ and $t_i^{(f)}$ are the starting time, due date and finishing time of i-job, respectively. Note that in equation 2, for simplicity, we abandon x on the right-hand side expression, implicitly implying that the expression is computed with respect to x. Second, for service jobs, we calculate the ratio between the amount of utilized computing resources (i.e., CPU and memory) and the amount of reference computing resources. Specifically, when a set $J^{(s)}$ of service jobs fail to receive their reference amount computing resource, and hence, undergo QoS degradation, we calculate the under-provisioning ratio $v_{cpu}^{(s)}(x)$ of CPU and the under-provisioning ratio $v_{mem}^{(s)}(x)$ of memory as

$$v_{cpu}^{(s)}(x) = v_{mem}^{(s)}(x) = \frac{1}{J^{(s)}} \sum_{i=1}^{J^{(s)}} \frac{r_i^{(utiz)}}{r_i^{(ref)}}, \tag{3}$$

[2] www.aws.amazon.com/ec2/pricing/on-demand

where $r_i^{(utiz)}$ is the computing resource utilized, and $r_i^{(ref)}$ is the reference computing resource. In case of CPU,

$$r_i^{(utiz)} = \sum_{\xi \in \Xi} \sigma_{i,\xi} f_\xi, \tag{4}$$

and

$$r_i^{(ref)} = \sum_{\xi \in \Xi} \sigma_{i,\xi}^{(ref)} f_\xi^{(ref)}, \tag{5}$$

with $\sigma_{i,\xi}$ is the percentage of the processing element/processing core ξ that is assigned to job i, f_ξ is the frequency of processing core ξ, $\sigma_{i,\xi}^{(ref)}$ is the percentage of reference processing core ξ that the job i requests, f_ξ is the reference frequency of processing core ξ, and Ξ is the set of processing cores. Similar to equation 2, in equation 3, we abandon x on the right-hand side expression for simplicity. Finally, we compute the SLA violation compensation as

$$h(x) = v^{(b)}(x) + \frac{1}{v_{cpu}^{(s)}(x) + v_{mem}^{(s)}(x)}, v_{cpu}^{(s)}(x) + v_{mem}^{(s)}(x) \neq 0. \tag{6}$$

Note that for comparability, the values are normalized before applying the computation.

3.1.2. PDM utility

The PDM computes its utility value, associated with each profile x, as

$$u(x) = r(x) - c(x, S), \tag{7}$$

where $S = \{WT, PV, BT, EZ, FC\}$ is the set of utilized components; the revenue $r(x)$ of the black-box and the semi black-box approach are given differently, which will be described in corresponding sections; the cost $c(x, S)$ is normalized to [0,1], after being computed as

$$c(x, S) = \sum_{i=1}^{T} \sum_{j \in S} \frac{P_{j,i} \Delta t}{E_j^{(max)}} (c_j^{(op)} + c_j^{(cap)}), \tag{8}$$

where $x_i = \sum_{j \in S} P_{j,i}, i = 1, ..., T$, with $P_{j,i}$ is the output power of component j at time step i (this output power is obtained from the solution of PDM scheduling); $c_j^{(op)}$, $c_j^{(cap)}$ and $E_j^{(max)}$ are respectively the operational cost, the capital

cost (i.e., replacement cost) and the maximum energy produced by the power source component j during its lifetime; Δt is the duration of one time step. Note that $P_{j,i}$ can be positive or negative depending on the status of the source j (i.e, charging or discharging). We define the operational cost $c_j^{(op)}$ based on the characteristics of each component. This cost is the fixed amount of maintenance cost during the entire lifetime of a component. Note that the cost to purchase hydrogen for fuel cell is not considered. In this way, we only use fuel cell when hydrogen tank is not empty.

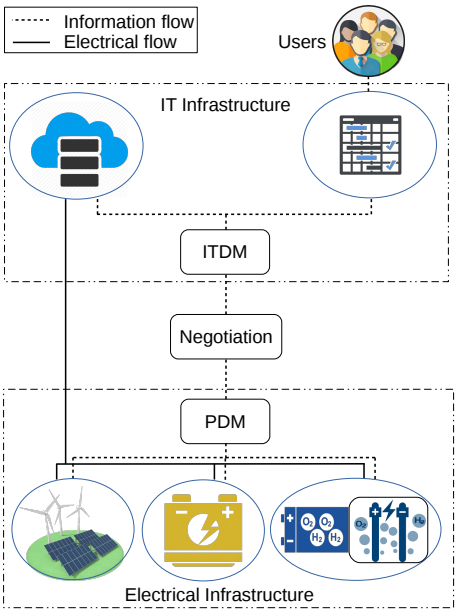


Figure 1: Datacenter model

3.1.3. Difference between two profiles

In order to quantify the difference between two profiles x and y, we define the distance between them, denoted as $d(x, y)$. Though $d(x, y)$ can be implemented by any method, in the experiment, we use Mean Square Error (MSE) and inverse Pearson correlation to implement this distance.

3.2. A specific implementation of DMs

Based on the above generic model, we implement one ITDM scheduler and one PDM scheduler, which were described partially in our previous works Caux et al. (2018), Haddad et al. (2019). As in Caux et al. (2018), the ITDM scheduler is implemented with three versions of Best Fit algorithm; these versions are different on the way that the jobs are sorted: (i) due date, closest job first; (ii) arrival time, first come first served; and (iii) job size, longest one first. Each of those algorithms takes one profile as an input, called *power constraint* Caux et al. (2018). We use the PDM's profile with highest utility for that power constraint. However, the scheduling algorithms are not required to strictly respect the power constraint, because each PDM's profile has a relaxation value $\bar{\rho}$, indicating how much the ITDM is allowed to violate the power constraint. For instance, when $\bar{\rho} = 0.2$, the power demand can be arbitrarily lower than the power supply, but the power demand can only exceed the power supply by 20%. In other words, denoting the power constraint as $l^{(PD)}$, the output profile of the scheduling algorithm must be lower than or equal to $(1+\bar{\rho}) \times l^{(PD)}$. The ITDM has three techniques to generate multiple scheduling solutions, and therefore multiple profiles. The first technique is to vary the relaxation value. The second technique is to use different versions of Best Fit algorithm. As the third technique, the ITDM supports multiple service grades, i.e., multiple QoS levels of a job. For example, video streaming service can provide multiple encoding quality levels. We define that the degraded QoS of a batch job is associated with delay violation, whereas the degraded QoS of a service job is associated with resource under-provisioning. In this way, when a job has multiple service grades, the ITDM finds one scheduling solution for each service grade. In other words, the ITDM generates multiple scheduling solutions, each solution corresponds to a service grade.

As described in Haddad et al. (2019), the PDM scheduler is based on the following integer linear program.

$$
\begin{aligned}
&\max \sum_{i=1}^{T} P_i^{(prod)} \\
&s.t.\ P_i^{(prod)} \leq P_{WT,i} + P_{PV,i} + (P_{FC,i} + P_{BT,i}^{(out)})\eta \\
&\qquad\qquad - (P_{EZ,i}^{(in)} + P_{BT,i}^{(in)})\eta, \quad i = 1, ..., T, \\
&\quad (1 - \bar{\phi}) \times l_i^{(IT)} \leq P_i^{(prod)} \leq (\bar{\phi} + 1) \times l_i^{(IT)}, \quad i = 1, ..., T, \\
&\quad \text{state of charge equations,} \\
&\quad \text{electrolyzer equations,} \\
&\quad \text{fuel cell equations,} \\
&\quad \text{level of hydrogen equations,} \\
&\quad \text{bounds of FC, EZ,} \\
&\quad \text{bounds of state of charge,} \\
&\quad \text{bounds of level of hydrogen,}
\end{aligned}
\tag{9}
$$

where corresponding to the time step i, $P_i^{(prod)}, l_i^{(IT)}$, $P_{WT,i}$, $P_{PV,i}$, $P_{FC,i}$, and $P_{BT,i}^{(out)}$, are respectively the produced power, *load power*, wind turbine power, photovoltaic power, power delivered by fuel cell, and power discharged from battery; $P_{EZ,i}^{(in)}$ and $P_{BT,i}^{(in)}$ are the power put into electrolyzer, and the power used to recharge battery, respectively; η is the inverter efficiency. The resolution of the integer linear program takes the load $l^{(IT)}$ as an input Haddad et al. (2019). We use the ITDM's profile with highest utility for that input. To output multiple scheduling solutions, the PDM varies the relaxation value $\bar{\phi}$. The meaning of relaxation in PDM is similar as in ITDM, except the case when the power supply is higher than the power demand. For example, when $\bar{\phi} = 0.2$, the power supply cannot be arbitrarily higher than the power demand, but can only be higher by 20%.

Inside the scheduling algorithms, the distance between two profiles is implemented by MSE and inverse Pearson correlation. Denoting T as the number of time steps in each profile, the MSE distance between $x = \{x_1, ..., x_T\}$ and $y = \{y_1, ..., y_T\}$ is given as

$$d(x,y) = \frac{1}{T}\sum_{i=1}^{T}(x_i - y_i)^2. \tag{10}$$

Note that, similar to Euclidean distance, MSE is invariant if we change the order of power values inside the profiles. In contrast, inverse Pearson correlation can recognize that change, because it can realize the trends and evolution of power values. As a result, we implement both methods and compare their performance. When using inverse Pearson correlation, the distance between x and y is given as

$$d(x,y) = \frac{\sqrt{\sum_{i=1}^{T}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{T}(y_i - \bar{y})^2}}{\sum_{i=1}^{T}(x_i - \bar{x})(y_i - \bar{y})}, \tag{11}$$

where $\bar{x}$ and $\bar{y}$ are the averages of the sets $\{x_1, ..., x_T\}$ and $\{y_1, ..., y_T\}$, respectively.

4. Black-box approach

As a straightforward approach, we introduce and analyze a black-box negotiation algorithm, named Scheduling Negotiation Algorithm. Instead of considering the whole system by a global model, we consider the sub-systems of NM, ITDM and PDM separately. The sub-problems are solved with as little specific information as possible. The only exchanged information is power profiles.

In SAN, at the first negotiation round, each Decision-support Sub-module generates its feasible profiles, called *candidates*, and sends them to the NM. Each DM generates its profiles without considering the other DM. As described in section 3, each profile has an associated *utility*. The NM selects half of the candidates of each DM as hints, and abandons the other half. The selection is based on the weighted sum of utilities and similarity, which is called *weighted similarity*. We will describe this selection in detail in the next subsection. In the next negotiation round, the NM requests a DM to generate a new half of candidates. The NM selects the DM to request based on the *negotiation mode*, which will be also described in the next subsection. If the ITDM is selected,

Symbol	Name	Description
$\check{x}$	the set of ITDM hints	$\check{x} = \{\check{x}^1, ..., \check{x}^m, ...\check{x}^M\}$, where $\check{x}^m$ is an ITDM hint
$\check{y}$	the set of PDM hints	$\check{y} = \{\check{y}^1, ..., \check{y}^n, ...\check{y}^N\}$, where $\check{y}^n$ is a PDM hint
$\dot{x}$	one variable in the binary integer program (12)	$\dot{x} = \{\dot{x}^1, ..., \dot{x}^M\}$ indicates which ITDM hint is selected for the matched pair, for example, $\dot{x} = \{0, 1, 0\}$ indicates that $\check{x}^2$ is selected for the matched pair
$\dot{y}$	one variable in the binary integer program (12)	$\dot{y} = \{\dot{y}^1, ..., \dot{y}^N\}$ indicates which PDM hint is selected for the matched pair, for example, $\dot{y} = \{1, 0, 0\}$ indicates that $\check{y}^1$ is selected for the matched pair
$\delta(\check{x}, \check{y})$	minimum distance between two sets $\check{x}$ and $\check{y}$	this minimum distance is defined as $min\{d(\check{x}^1, \check{y}^1), d(\check{x}^1, \check{y}^2), ..., d(\check{x}^M, \check{y}^N)\}$
ϕ	the relaxation values of ITDM hints	$\phi = \{\phi^1, ..., \phi^m, ..., \phi^M\}$, where $\phi^m \in (0, 1]$ is the relaxation value of the hint $\check{x}^m$; the ITDM assigns one relaxation value to each hint
ρ	the relaxation values of the PDM hints	$\rho = \{\rho^1, ...\rho^n, ..., \rho^N\}$, where $\rho^n \in (0, 1]$ is the relaxation value of the hint $\check{y}^n$; PDM assigns one relaxation value to each hint

Table 2: Main notations of black-box approach

the NM requests this ITDM by sending to it the PDM hint with highest utility as *power constraint* (i.e., $l^{(PD)}$). Similarly, if the PDM is selected, the NM requests this PDM by sending to it the ITDM hint with highest utility as *load* (i.e., $l^{(IT)}$). As mentioned in the previous section, $l^{(PD)}$ and $l^{(IT)}$ serve as the inputs of ITDM scheduler and PDM scheduler, respectively. After receiving the request, the DM reschedules to generate its new half of candidates. Then the DM sends this new half to the NM. The NM combines this new half of the DM with the hints of this DM in the previous round, in order to form a new candidate set for this DM. Again, the NM selects new hint set based on the new candidate set. The process is repeated until the NM found a pair of {1 ITDM hint, 1 PDM hint} that are *matched*. A pair is called matched when it consists of two profiles that have maximized summation of utilities, while the distance between these two profiles is below a given threshold ε. Note that, in this black-box approach, we set the ITDM's cost $c(\cdot) = 0$, and the PDM's revenue $r(\cdot) = 1$.

The proposed negotiation algorithm has two stages, namely, *checking for matched pair* and *negotiating*. After executing stage 1, the algorithm checks whether to continue the stage 2 or not. We describe these two stages as follows.

- Stage 1 - *Checking for matched pair*: after having a new hint set, the NM checks whether there is a pair of {1 ITDM hint, 1 PDM hint} that approximately matches with each other. If this matched pair exists, the NM returns these two hints to the DMs as the final negotiation solution. Then it is not necessary to continue the stage 2.

- Stage 2 - *Negotiating*: if the NM cannot find any matched pair, stage 2 is executed. We propose a turn-based mechanism in which the two DMs do not reschedule at the same time. In each negotiation round, the NM determines the *negotiation mode*, which indicates which DM is allowed to reschedule at the next negotiation round. To do this, the NM monitors the quality of the rescheduling, in order to decide which DM should reschedule at the next round. In the next subsection, we describe this negotiation

algorithm in detail.

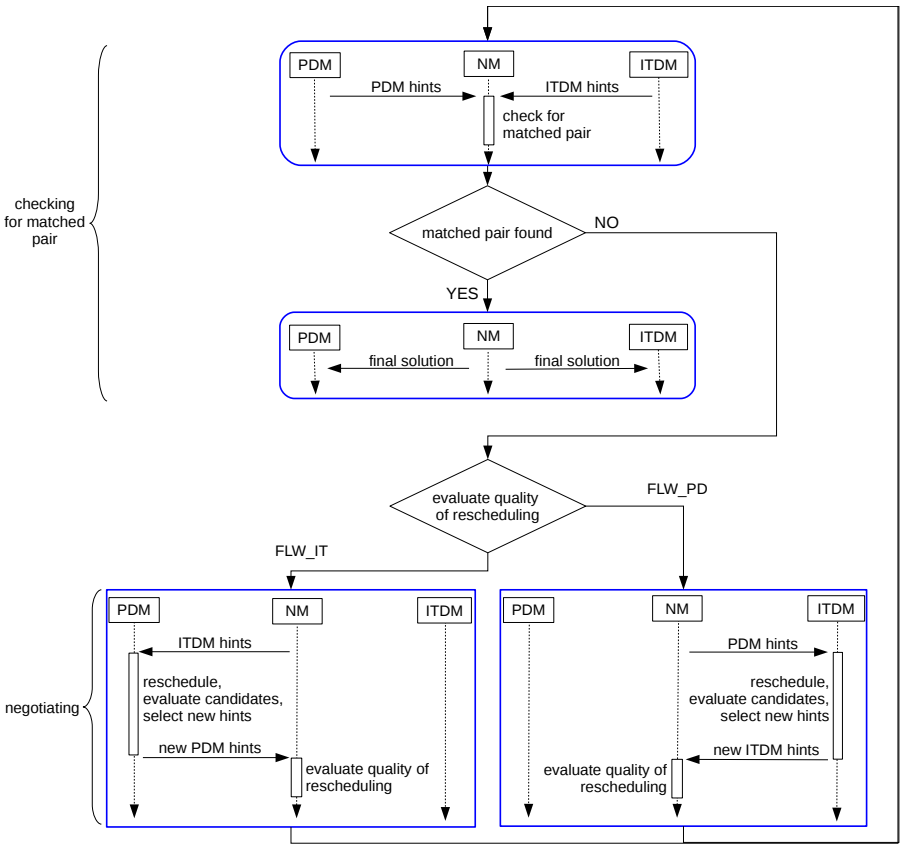


Figure 2: Scheduling-based negotiation algorithm. How to *evaluate quality of rescheduling* is explained in subsection 4.1.2.

4.1. Details of algorithm

In Fig. 2, we show diagram of the proposed algorithm; stage 1 is depicted at the upper part, and stage 2 is depicted at the lower part of the figure. In stage 1, if a matched pair is found, the final solutions are sent to the DMs, and the negotiation stops. If a matched pair is not found, stage 2 is performed. In stage 2, the algorithm is in one of these two *negotiation modes*: (i) following ITDM (named FLW_IT), and (ii) following PDM (named FLW_PD). In the FLW_IT

mode, the NM requests the PDM to reschedule; then the PDM reschedules to generate a new half of candidates, and sends back to the NM. Similarly, in the FLW_PD mode, the NM requests the ITDM to reschedule, then the ITDM reschedules to generate a new half of candidates. In brief, a negotiation round includes three tasks: (i) scheduling, (ii) evaluating the candidates using *weighted similarity*, and (iii) selecting new hints. In the FLW_IT mode, after receiving new half of candidates from the PDM, the NM evaluates the quality of rescheduling in order to decide the mode for the next negotiation round. The procedures in the FLW_PD mode follow the same processes.

The details of the two stages are provided in the following subsections.

4.1.1. Stage 1 - Checking for matched pair

We denote the set of ITDM hints as $\check{x} = \{\check{x}^1, ...\check{x}^m, ..., \check{x}^M\}$, and the set of PDM hints as $\check{y} = \{\check{y}^1, ..., \check{y}^n, ...\check{y}^N\}$, where M and N are respectively the number of ITDM hints and PDM hints. We describe the main notations of the black-box approach in table 2. Stage 1 checks whether there is a pair of {1 hint $\check{x}^m$, 1 hint $\check{y}^n$} that matches. In order to find that pair, we solve a binary integer program with two variables $\dot{x}$ and $\dot{y}$. These variables are two binary vectors: $\dot{x} = \{\dot{x}^1, \ldots, \dot{x}^m, \ldots, \dot{x}^M\}$, $\dot{y} = \{\dot{y}^1, \ldots, \dot{y}^n, \ldots, \dot{y}^N\}$, where $\dot{x}^m \in \{0, 1\}$, $m = 1, \ldots, M$, and $\dot{y}^n \in \{0, 1\}$, $n = 1, \ldots, N$. Each $\dot{x}^m$ is a binary value, indicating that $\check{x}^m$ is selected or not.

Stage 1 is represented by the following binary integer program.

$$\begin{aligned} \max_{\dot{x},\dot{y}} \quad & \sum_{m=1}^{M} u(\check{x}^m)\dot{x}^m + \sum_{n=1}^{N} u(\check{y}^n)\dot{y}^n, \\ \text{s.t.} \quad & \sum_{m=1}^{M}\sum_{n=1}^{N}(1-\phi^m)(1-\rho^n)\dot{x}^m\dot{y}^n d(\check{x}^m, \check{y}^n) < \varepsilon, \\ & \sum_{m=1}^{M} \dot{x}^m = 1, \sum_{n=1}^{N} \dot{y}^n = 1, \\ & \dot{x}^m, \dot{y}^n \in \{0, 1\}, \end{aligned} \tag{12}$$

where

- $u(\check{x}^m)$ and $u(\check{y}^n)$ are the utilities of the hints $\check{x}^m$ and $\check{y}^n$, respectively,
- $d(\check{x}^m, \check{y}^n)$ is the distance between $\check{x}^m$ and $\check{y}^n$,
- ϕ^m and ρ^n are the relaxation values of the hints $\check{x}^m$ and $\check{y}^n$, respectively,
- ε is the distance threshold, which is set through experiment parameters.

The binary integer program finds a pair of profiles that maximizes the summation of utilities, while the distance between these two profiles is lower than the distance threshold ε.

4.1.2. Stage 2 - Negotiating

In this stage, the NM continues to use the hints of the previous stage. This stage includes the scheduling process of the DMs and the evaluating process of the NM. The scheduling process is for generating multiple feasible scheduling solutions, corresponding to multiple candidates. The evaluating process is for assessing the candidates against the hints, based on *weighted similarity*.

We describe the FLW_PD mode as follows. Note that the FLW_IT mode undergoes similar processes. After receiving the request with $l^{(PD)}$ from the NM, the ITDM reschedules to find a new half of candidates. Then the ITDM sends these new candidates to the NM. The NM combines the ITDM hints of previous round with the newly received candidates to form the new ITDM candidate set. Then, the NM computes the quality w of each ITDM candidate x based on *weighted similarity*, as follows.

$$w = \sum_{n=1}^{N} \frac{u(x) + u(\check{y}^n)}{d(x, \check{y}^n)}, \tag{13}$$

where

- $d(x, \check{y}^n)$ is the distance between the candidate x and the hint $\check{y}^n$,
- $u(x)$ is the utility of the candidate x, and $u(\check{y}^n)$ is the utility of the hint $\check{y}^n$.

The NM selects half of ITDM candidates to become the ITDM's new hints. Then the NM decides whether to continue the FLW_PD mode or switch to the FLW_IT mode. To this end, the NM evaluates the quality of the ITDM's reschedule by comparing two distances: (i) the minimum distance $\delta(\cdot)$ between the PDM hints and the ITDM hints of previous negotiation round, and (ii) the minimum distance $\delta(\cdot)$ between the PDM hints and the ITDM's new hints of current negotiation round. If the former is greater than the latter, we say that the quality of the ITDM's reschedule is satisfactory. Then the NM allows the ITDM to execute another reschedule at the next negotiation round. If the former is smaller than the latter, the PDM is allowed to reschedule. The intuition behind this comparison is that, when the former is greater than the latter, we expect that the ITDM will generate new candidates that get closer to the PDM hints. To compute the minimum distance $\delta(\cdot)$ between two sets, we compute the distance of every pair between two sets, then select the smallest distance.

The negotiation process repeats until the NM finds a pair of matched profiles, or the number of negotiation rounds reaches a given threshold. This threshold is set via an experiment parameter. When the NM cannot find a matched pair, the pair with the smallest distance is selected as the final solution. Then the PDM's profile in that pair is implemented as the power supply, and there is a possibility that the performance of the datacenter is degraded when the power supply is lower than the power demand.

4.2. Stability of black-box approach

We implement a middleware of datacenter powered by renewable energies in order to carry out experiments for our research, as described in Sayah et al. (2017). With SAN, we evaluate and analyze the execution time, as well as the *generational distance* of each individual execution, as showed in Fig. 3 and Fig. 5. *Generational distance* is first introduced by Van Veldhuizen and Lamont Van Veldhuizen and Lamont (1998), for estimating the average Euclidean distance between a solution and the nearest point on Pareto front. We measure

this distance in order to evaluate the trade-off of our final solution. To to this, we trace the generated solutions and approximate their Pareto front by finding the set of non-dominated solutions; then we measure the generational distance of our final solution to this Pareto front (Fig. 4). We note that the Pareto front is the best known approximation, because in order to compute the complete Pareto front, we need to have the set of all feasible solutions, which is highly complex to generate.

In Fig. 3, we depict the execution time, grouped by the number of hints. Each gray dot represents the execution time of an individual execution. The vertical line and the middle bar, respectively, represent the standard deviation and the mean, which are computed from the values of gray dots. We can see that, when the number of hints is 4, the execution time varies widely. When the number of hints is high, the execution time becomes more consistent but also grows higher. In Fig. 5 and Fig. 6, we show the generational distance with respect to the number of negotiation rounds and the number of hints, respectively. In these experiments, we set the distance threshold $\varepsilon = 1.12$. Fig. 6 shows that the black-box model cannot guarantee stable results, in term of generational distance.

The experiments show that SAN is not able to provide highly consistent and stable results. In the next section, we propose a semi black-box approach for negotiation.

5. Semi black-box approach

To deal with the instability of the black-box approach, in this section, we propose a semi black-box approach, which includes a game model and a negotiation algorithm, named Game Theory Based Negotiation. The game has two players, namely *IT-Player* and *PD-Player*, corresponding to the two sub-systems. The PD-Player controls the electrical sub-system, playing the role of a supplier; the IT-Player controls the IT sub-system, playing the role of a buyer (Fig. 7). The IT-Player and PD-Player have the functionality of game players,

Symbol	Name	Description
π	price	price is in Euros/kWh, proposed by the PD-Player, indicating the per-unit payment that the IT-Player has to pay to the PD-Player, having the form $\pi = \{\pi_1, ..., \pi_T\}$
$\hat{x}$	order	an order is a power profile, proposed by the IT-Player, indicating how much power the IT-Player wants to buy from the PD-Player, having the form $\hat{x} = \{\hat{x}_1, ..., \hat{x}_T\}$
π^{IT}	IT incentive price	π^{IT} is the price that the IT-Player can offer to the PD-Player, i.e., *willing-to-pay* price; π^{IT} has the vector form like π
π^{PD}	PD incentive price	π^{PD} is the price that PD-Player can offer to IT-Player; π^{PD} has the vector form like π
x^{IT}	aspiration order	x^{IT} is the profile that the IT-Player desires, with respect to users' demand; x^{IT} has the vector form like $\hat{x}$
x^{PD}	aspiration supply	x^{PD} indicates the power that the PD-Player wants to supply to the IT-Player, with respect to the states of electrical components; x^{PD} has the vector form like $\hat{x}$
α	sacrifice variable	this variable is used in sacrifice mechanism, making the DMs sacrifice their utility in order to continue negotiation
γ	sacrifice step-size	γ indicates how much α is increased every time we use sacrifice mechanism; this step-size is set though experiment parameters

Table 3: Main notations of semi black-box approach

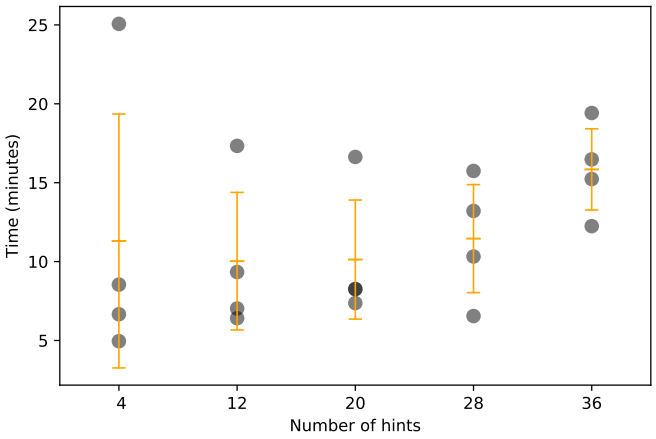


Figure 3: Execution time with respect to the number of hints

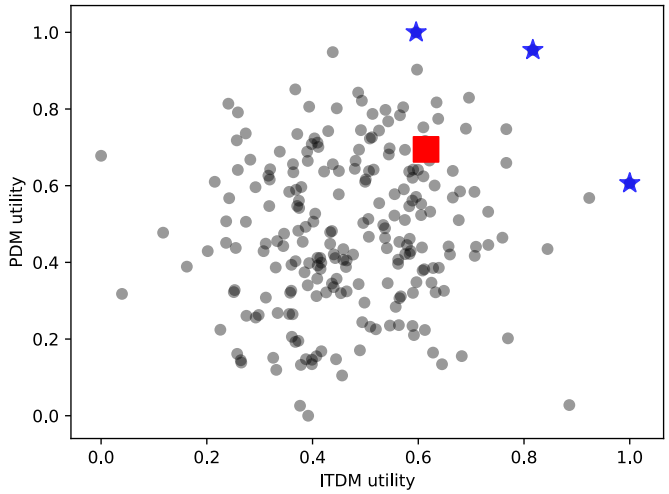


Figure 4: Illustration of solutions (gray dots), final solution (red square), and Pareto front (blue stars)

and they only use ITDM and PDM as their schedulers. With this game design, we abandon the role of the NM.

We introduce the new terms *order*, *aspiration order*, *aspiration supply*, and *price*. Among them, *order*, *aspiration order*, and *aspiration supply* are profiles. We will provide the detailed definitions of *aspiration supply* and *aspiration order* in the next section. The definitions of *order* and *price* are as follows.

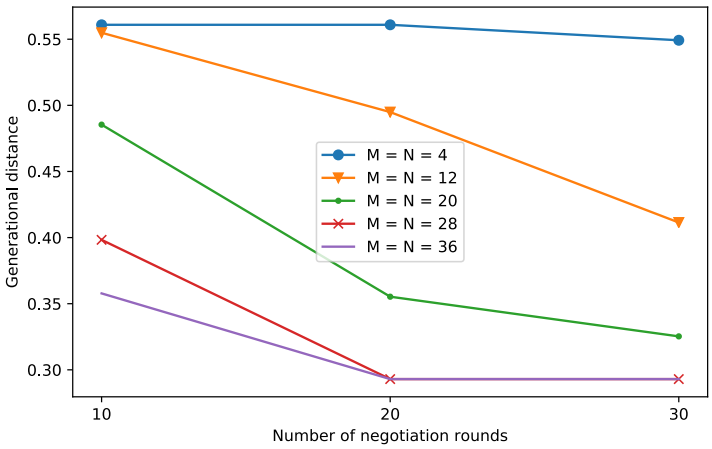


Figure 5: Generational distance with respect to the number of negotiation rounds

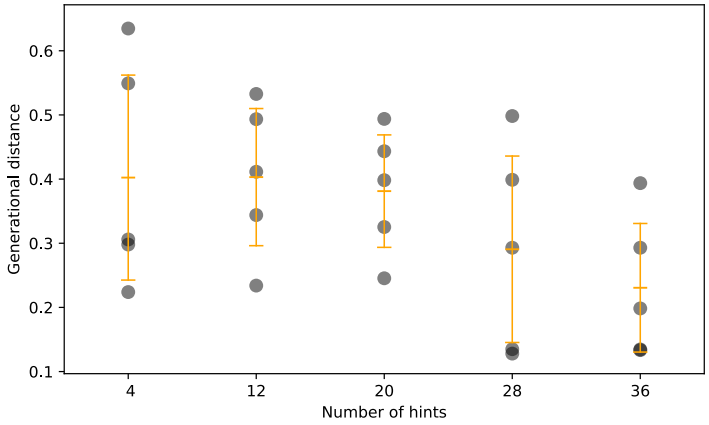


Figure 6: Generational distance with respect to the number of hints

- *Order*: a power profile, which indicates how much power the IT-Player plans to buy from the PD-Player.
- *Price*: similar to *opening price* in Krause et al. (2006), *price* is the per-unit payment that the IT-Player has to pay to the PD-Player, *price* is in Euros/kWh.

The roles of the IT-Player and PD-Player are different, specifically, the PD-Player first proposes *price*, then the IT-Player places *order* based on that price.

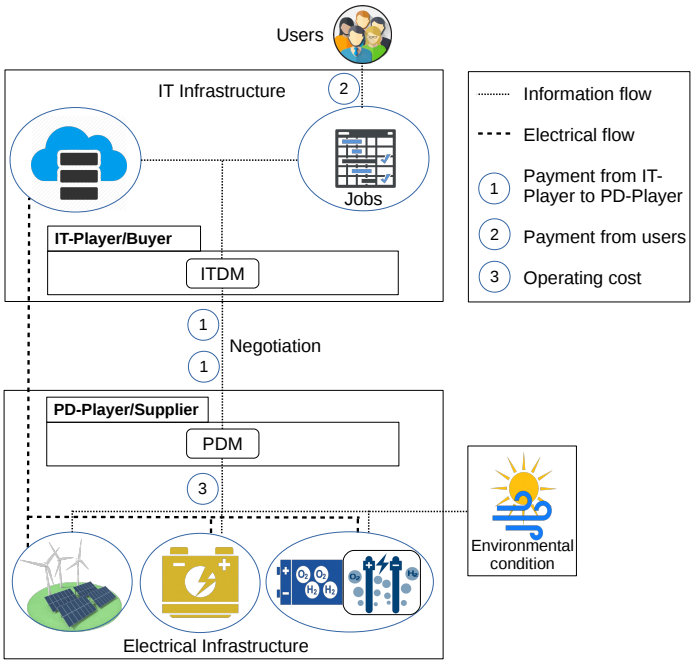


Figure 7: Game model

In other words, the PD-Player controls the *price*, while the IT-Player controls the *order*. In this way, the PD-Player is able to reflect the availability of energy in the *price*.

In the proposed game, each player has its own objective and constraints, which are described as follows.

- IT-Player: maximizes its utility (i.e., monetary gain), while satisfying the users' demand. Unlike SAN where the utility is an abstract value and normalized to [0, 1], the utility in GAN is the amount of money a DM earns. Specifically, the IT-Player utility is defined as the difference between: (i) the payment that the users give to the IT-Player, and (ii) the payment that the IT-Player gives to the PD-Player. The first payment is $r(x)$, corresponding to an ITDM profile x, computed based on the Amazon pricing. Each ITDM profile indicates how much power the IT-Player will buy from the PD-Player. This power provides a certain computing capacity to the

users; then, the users pay to the IT-Player.

- PD-Player: maximizes its utility (i.e., monetary gain), while considering the environmental conditions and operating cost (i.e., operational and capital cost, as in section 3). That utility is defined as the difference between: (i) the payment from the IT-Player and (ii) the operating cost of the electrical infrastructure.

We depict the two players and their relationship as in Fig. 7.

The decision variables of the PD-Player are *price* and *energy source scheduling*, whereas those of the IT-Player are *order* and *job scheduling*. The players obtain scheduling solutions from the ITDM and PDM. The PD-Player obtains the energy source scheduling solution from the PDM; similarly, the IT-Player obtains the IT job scheduling solution from the ITDM. We described the DMs' scheduling algorithms in section 3.

The proposed game is not completely a cooperative game or a non-cooperative game. The players are partially selfish, i.e., each player maximizes its own utility, however, at some points, the players sacrifice their utility, in order to reach a negotiation agreement. We will define this *sacrifice mechanism* in subsection 6.2.

6. Overview of GAN algorithm

6.1. Terms definitions

Fig. 8 shows the variables and procedures of the proposed algorithm. In the figure, we introduce some new terms.

- *Aspiration order*: the power profile that the IT-Player desires to order, after considering users' demand. We also use aspiration order as the *load* $l^{(IT)}$ for the PDM's scheduler.

- *Aspiration supply*: the power profile that the PD-Player desires to supply, after considering the environmental conditions and operating cost. The

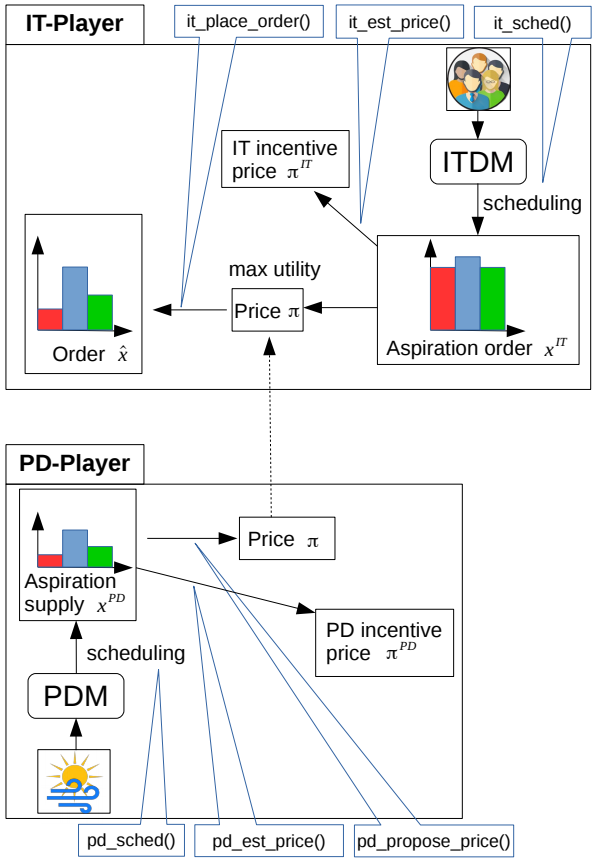


Figure 8: Variables and procedures in the algorithm

final solution of the negotiation is the aspiration supply in the last negotiation round. We also use aspiration supply as the *power constraint* $l^{(PD)}$ for the ITDM's scheduler.

The aspiration order and aspiration supply are used as two reference points Krause et al. (2006). Also, we introduce two other reference points, namely, IT incentive price and PD incentive price. In a buyer-supplier game, reference points are important; and the players can revise these points during the negotiation, in order to reach an agreement. That revision is necessary because there is a possibility that the game has negative bargaining zone Krause et al. (2006).

Fig. 9 is an example of the aspiration supply, order, and aspiration order.

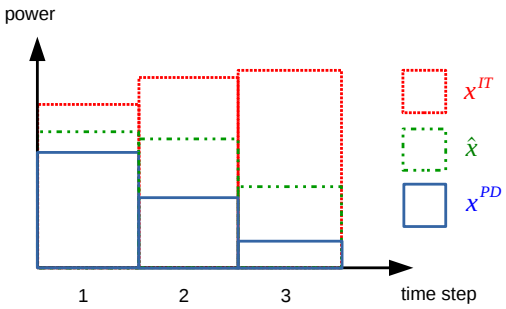
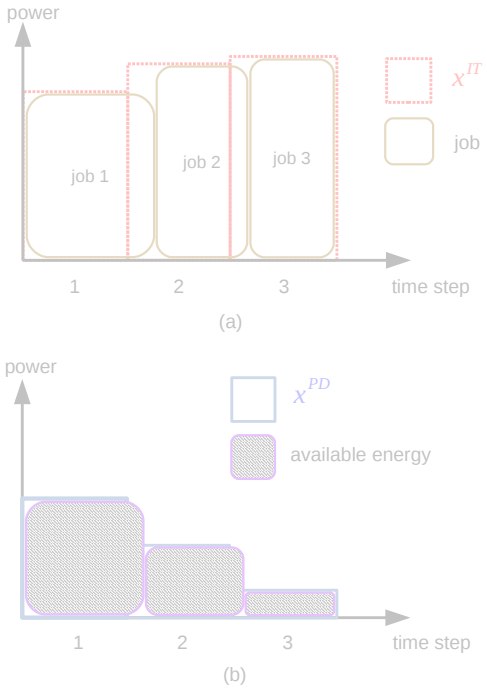


Figure 9: An example of power profiles



proportional to x^{PD}, we can expect that $\hat{x}$ has similar curve with x^{PD}. Fig. 10 shows how to generate the aspiration order x^{IT} and the aspiration supply x^{PD}. The IT-Player generates x^{IT} based on the scheduling solution of job 1, job 2 and job 3. The difference between x^{IT} and $\hat{x}$ is that x^{IT} is the direct result of a scheduling solution, whereas $\hat{x}$ is the output of another computation after we already had a scheduling solution. Similar to IT-Player, the PD-Player generates x^{PD} based on the scheduling solution of available energy. We will describe in detail these generations in the formulations of IT-Player and PD-Player. Table 3 summarizes the main notations in semi black-box approach.

In GAN, the two modes FLW_IT and FLW_PD correspond to *follow IT-Player* and *follow PD-Player*. Due to our turn-based design, at a time, the algorithm follows only IT-Player or PD-Player. The decision of which mode to follow is not made by any player, but by the global variable *mod*, as will be explained in section 7. On the other hand, the players are selfish, and they negotiate just because they foresee their benefit. Specifically, the IT-Player wants the PD-Player to follow it, i.e., the IT-Player wants the PD-Player to reschedule, in order to propose a more attractive supply. Similarly, the PD-Player wants the IT-Player to reschedule and propose a more attractive order. In this way, a *problem of selfishness* may occur: both players do not want to follow each other, and they stop negotiating without reaching any agreement. To deal with this problem, we introduce the mechanism of *incentive pricing*. In this mechanism, each player proposes an *incentive price* that is possibly attractive to the other player. However, the players cannot freely propose this price; they must guarantee that their utilities are not reduced if this price is used. The intuition behind this mechanism is as follows.

- *IT incentive pricing*: if the PD-Player is attracted by the IT incentive price, and the PD-Player wants this price to be used, then the PD-Player must supply a power profile that is equal to the aspiration order.
- *PD incentive pricing*: if the IT-Player is attracted to the PD incentive price, and the IT-Player wants this price to be used, then the IT-Player

must place an order that is equal to the aspiration supply.

The incentive prices are the signals of the willingness to cooperate. Incentives can be a powerful tool for the buyer to seek for agreement with the supplier Terpend and Krause (2015). In our model, we allow both supplier and buyer to use this tool. We utilize *cooperative incentive*, instead of *competitive incentive* Terpend and Krause (2015), since cooperative incentives enable players to share cost Giunipero (1990) and/or revenue Modi and Mabert (2007).

6.2. Sacrifice mechanism

We found that, even with incentive pricing mechanism, the problem of selfishness may still occur, i.e., both players stop negotiating while an agreement is not reached. The problem of selfishness occurs when the incentive is not attractive enough for both players to follow each other. From the system-wide perspective, this situation is unacceptable, since the system will stop working. To deal with this problem, we introduce *sacrifice mechanism*. At first, both players negotiate without sacrificing. If the problem of selfishness occurs, the players sacrifice their utility to continue negotiating, trying to reach an agreement.

A player sacrifices by giving more attractiveness to the incentive price, even though the utility of that player is reduced. The intuition behind this is that, if the players stop negotiating without reaching any agreement, we can assume that the players and the end-users receive a very low utility (e.g., negative infinity utility); therefore, the players need to continue negotiating, even though their utility decreases. We introduce the sacrifice variable α, which indicates how much utility the players sacrifice. Every time the problem of selfishness occurs, we increase α, raising the sacrifice quantity that each player should abide by.

6.3. Two modes of negotiation

Fig. 11 depicts two sequential diagrams of two modes FLW_IT (i.e., following IT-Player) and FLW_PD (i.e., following PD-Player). Those modes are

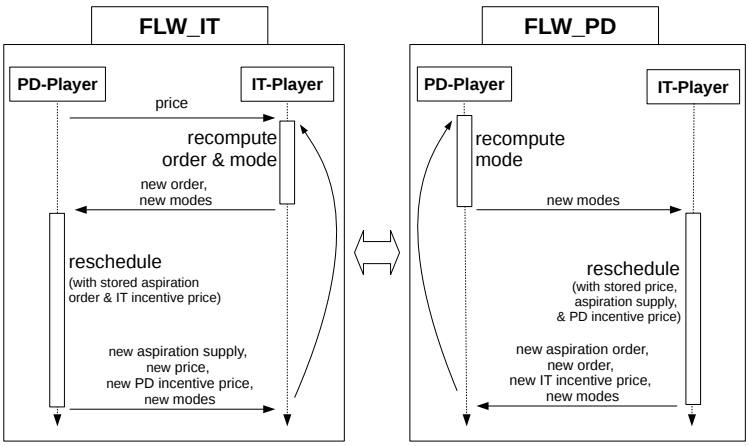


Figure 11: Two modes of GAN algorithm

described as follows.

- FLW_IT: The PD-Player reschedules and computes new aspiration supply, new price, new PD incentive price, and new mode. After that, the PD-Player sends these new information to the IT-Player. The IT-Player computes new order and new mode, then sends to the PD-Player.

- FLW_PD: The IT-Player reschedules and computes new aspiration order, new order, new IT incentive price, and new mode. Then the IT-Player sends these new information to the PD-Player. The PD-Player computes new mode, then sends to the IT-Player.

We note that, as showed in Fig. 11, if some information is not modified, it can be stored and reused. For example, in the FLW_IT mode, the PD-Player can reuse the aspiration order and the IT incentive price.

6.4. Mode controlling

In order to control the mode, we use 3 variables: IT-Player local variable $it\_pre \in \{FLW\_IT,$
$FLW\_PD\}$, PD-Player local variable $pd\_pre \in \{FLW\_IT, FLW\_PD\}$, and global variable $mod \in \{FLW\_IT, FLW\_PD\}$. Both players always run the

Algorithm 1 Procedures of IT-Player

procedure follow_it()

  $\hat{x} \leftarrow$ it_place_order()

  Send a message of $\{\hat{x}, it\_pre, mod\}$, and wake up the PD-Player Loop

procedure follow_pd()

  $x^{IT} \leftarrow$ it_sched()

  $\hat{x} \leftarrow$ it_place_order()

  $\pi^{IT} \leftarrow$ it_est_price()

  Send a message of $\{x^{IT}, \hat{x}, \pi^{IT}, it\_pre, mod\}$, and wake up the PD-Player

Loop

Algorithm 2 Mode Updating

```
1: procedure update_mode(i_pre, p_pre, g_mod)
2:     if i_pre = FLW_IT and p_pre = FLW_IT then
3:         return FLW_IT
4:     else
5:         if i_pre = FLW_PD and p_pre = FLW_PD then
6:             return FLW_PD
7:         else
8:             return g_mod
```

Algorithm 3 IT-Player Loop

```
1: α ← 0, τ ← 1
2: while d(x^PD, x̂) > ε and τ ≤ ITER do
3:     if it_pre = FLW_IT then
4:         if pd_pre = FLW_PD then
5:             α ← α + γ
6:         else
7:             follow_it()
8:     else
9:         if pd_pre = FLW_PD then
10:            follow_pd()
11:        else
12:            if mod = FLW_IT then
13:                follow_it()
14:            else
15:                follow_pd()
16:    if c(x^PD, π^PD) − α < c(x̂, π) and d(x^PD, x̂) > ε then
17:        it_pre ← FLW_PD
18:    else
19:        it_pre ← FLW_IT
20:    mod ← update_mode(it_pre, pd_pre, mod)
21:    τ++
22:    Sleep until received new message from PD-Player
```

if			**then**	
checking current values			**updating new value**	**updating system mode**
variable *it_pre*	**variable *pd_pre***	**variable *mod***	**variable *mod***	**system mode**
FLW_IT	FLW_IT	(*)	FLW_IT	follow IT-Player
FLW_PD	FLW_PD	(*)	FLW_PD	follow PD-Player
FLW_PD	FLW_IT	FLW_IT	FLW_IT (keep unchanged)	follow IT-Player
		FLW_PD	FLW_PD (keep unchanged)	follow PD-Player
FLW_IT	FLW_PD	(*)	*mod* cannot be determined	system mode cannot be determined, the problem of selfishness occurs; we have to use sacrifice mechanism to solve this problem

Table 4: The determining of system mode; (*) means either FLW_IT or FLW_PD

same mode, called *system mode*; this mode depends only on *mod*. And *mod*, in turn, is jointly controlled by two local variables. The two local variables cannot directly control the system mode, but instead, indirectly through the global variable. The local variable *it_pre* indicates the mode that the IT-Player prefers; similarly, the local variable *pd_pre* indicates the mode that the PD-Player prefers. Based on the current values of the variables, we update the new value of *mod* and the system mode. Specifically, if $it\_pre = FLW\_IT$ and $pd\_pre = FLW\_IT$, we will set $mod = FLW\_IT$ regardless of current value of *mod*. Similarly, if $it\_pre = FLW\_PD$ and $pd\_pre = FLW\_PD$,

we will set $mod = FLW\_PD$ regardless of the current values of mod. If $it\_pre = FLW\_PD$ and $pd\_pre = FLW\_IT$, we keep the mod unchanged. If $it\_pre = FLW\_IT$ and $pd\_pre = FLW\_PD$, we cannot determine either mod or system mode. This means that we encounter the problem of selfishness, and negotiation cannot be continued. We have to use sacrifice mechanism to deal with this problem. We summarize how to determine system mode in table 4.

If the players are completely selfish, then each player always wants the other player to follow itself, regardless of whether it foresees benefit from following the other player or not. In this way, there is high possibility that we encounter the situation when $it\_pre = FLW\_IT$ and $pd\_pre = FLW\_PD$, meaning that the negotiation cannot be continued. In order to reduce the possibility of this situation, instead of designing the players to be completely selfish, we design them to be partially selfish, i.e.,

- when the IT-Player foresees benefit from following the PD-Player, the IT-Player sets $it\_pre = FLW\_PD$
- when the PD-Player foresees benefit from following the IT-Player, the PD-Player sets $pd\_pre = FLW\_IT$.

In this way, the problem of selfishness only occurs when both players cannot foresee any benefit of following each other.

7. Details of GAN algorithm

Corresponding to two modes, the algorithm of each player has two procedures, namely *follow_ it()* and *follow_ pd()*. Moreover, each player has a main loop, named *IT-Player Loop* (algorithm 3) and *PD-Player Loop* (algorithm 5). Note that, in the loops, the parameter $ITER$ is the maximum number of iterations. The two loops have the same values of $ITER$, distance threshold ε, and sacrifice step-size γ; other variables and parameters are different; especially, each player has its own value of α. The parameters $ITER$, ε, and γ are constants, and they are set via experiment parameters. We note that, two players

have the procedures with same names, but their implementations are different. Two procedures of IT-Player are showed in algorithm 1; two procedures of PD-Player are showed in algorithm 4. When the negotiation starts, the two main loops run in parallel. Within a loop, depend on the system mode, a procedure can be called accordingly. A player, after finishing a procedure, sends a message containing updated information to the other player. On the other hand, a loop goes to sleep after reaching its end, and wakes up after receiving a new message from the other player's procedure (i.e., line 22 in algorithm 3 and algorithm 5).

In the proposed game, the players have the choice to stop negotiating. However, both players have the opportunity to increase their utility if they continue to negotiate. We define that if both players stop negotiating before reaching an agreement, each player receives a very low utility. That is why we introduce the incentive pricing mechanism and the system mode controlling. In this way, no player is allowed to unilaterally stop the negotiation process.

7.1. IT-Player algorithm

7.1.1. Overview of IT-Player algorithm

Two IT-Player procedures *follow_ it()* and *follow_ pd()* are showed in algorithm 1. During one iteration, the IT-Player Loop can call one procedure; after finishing the iteration, the IT-Player Loop sleeps and waits for new message from PD-Player. Meanwhile, the called procedure starts to run; when this procedure finishes, this procedure wakes up the PD-Player Loop (algorithm 5). The two procedures are described as follows.

- *follow_ it()*: the IT-Player runs this procedure when $mod = FLW\_IT$. In this procedure, the IT-Player only needs to compute and sends to PD-Player the new order $\hat{x}$ and new modes (Fig. 11). This computation uses the price proposed by the PD-Player. The IT-Player computes $\hat{x}$ by the procedure *it_ place_ order()*.

- *follow_ pd()*: the IT-Player runs this procedure when $mod = FLW\_PD$. In this procedure, the IT-Player must reschedule to compute the new

aspiration order x^{IT}, the new order $\hat{x}$, and the new IT incentive price π^{IT} (Fig. 11). The IT-Player computes x^{IT}, $\hat{x}$, and π^{IT} by the procedure *it_sched()*, *it_place_order()* and *it_est_price()*, respectively.

7.1.2. IT-Player loop

The main loop of IT-Player is showed in algorithm 3. We terminate the loop when: (i) the distance between x^{PD} and $\hat{x}$ is equal or less than a threshold ε, or (ii) the maximum number of iterations is reached, i.e. $\tau >= ITER$. We verify these stopping criteria at line 2 of the algorithm 3. The criterion (i) means that the power demand is approximately equal to the power supply. When the criterion (ii) is satisfied, we stop the negotiation without obtaining a compromise solution. In this case, the power supply from the PD-Player is x^{PD}, and there is a possibility that the performance of the datacenter is degraded if the power supply is lower than the power demand.

In the loop, the IT-Player first checks and updates the sacrifice variable α. This check and update are done from line 3 to line 5. We need to increase α when both IT-Player and PD-Player stop following each other. Specifically, α is increased when the loop is not terminated but it_pre = FLW_IT and pd_pre = FLW_PD, meaning that IT-Player wants to follow itself and PD-Player also wants to follow itself. We increase α in order to help the players have more incentive to follow each other. We use α in the condition at line 16. In that condition, $c(\cdot)$ is the cost function, indicating the IT-Player payment to the PD-Player. This condition means that, if the PD incentive price and the aspiration supply are used, the IT-Player pays lower cost than when the current price and order are used. With larger α, this condition is more likely to hold, then it_pre is more likely to be set to FLW_PD (line 17).

In the loop, the IT-Player calls the procedure *follow_it()* or *follow_pd()* based on the system mode; and the system mode depends on the values of *mod*, it_pre and pd_pre (from line 7 to line 15). We summarize the behaviors of the loop, depending on the modes, as in table 5.

At the end of the loop (line 20), the IT-Player updates the global variable

if	**then**
updated value *mod*	**updating behavior**
FLW_IT	call *follow_it()*
FLW_PD	call *follow_pd()*
mod cannot be determined; the problem of selfishness occurs	increase α; use sacrifice mechanism to solve the problem of selfishness

Table 5: Behaviors of the IT-Player Loop, depending on the updated value of *mod*

mod using algorithm 2. This variable only switches its value when both local variables *it_pre* and *pd_pre* have switched to the same value. In this way, a player cannot unilaterally change the system mode, which results in conflicting.

The sacrifice mechanism supports the convergence of the algorithm. When the players stop following each other but the algorithm still has not converged, the players are required to sacrifice their utility by α, then the negotiation has more possibility to continue. Every time the players stop following each other, we increase α by an amount of the sacrifice step-size, which is denoted as γ (line 5). The sacrifice mechanism is applied until the algorithm converges or the maximum number of iterations is reached. The value of γ is set via experiment parameters. We summarize the behaviors of IT-Player Loop in table. 5.

7.1.3. IT-Player formulation

With T is the size of time window, we denote $x = \{x_1, x_2, ..., x_T\}$ as an ITDM profile. The price proposed by the PD-Player is $\pi = \{\pi_1, \pi_2, ..., \pi_T\}$. Utility function of IT-Player is

$$u(x, \pi) = r(x) - c(x, \pi) = r(x) - \sum_{k=1}^{T} \pi_k x_k, \tag{14}$$

where

- $r(x)$ is the revenue of the IT-Player, indicating the payment that the users

pay to the IT-Player; as mentioned in section 3, $r(x)$ is computed based on Amazon pricing,

- $c(\cdot)$ is the payment that the IT-Player pays to the PD-Player. Unlike in SAN where the cost is set $c(\cdot) = 0$, the cost in GAN is $c(x, \pi) = \sum_{k=1}^{T} \pi_k x_k$.

7.1.4. IT-Player procedures

Considering a negotiation round with the current aspiration order is x^{IT} and the current IT incentive price is π^{IT}, we denote the aspiration order and IT incentive price of the previous negotiation round as $\ddot{x}^{IT}$ and $\ddot{\pi}^{IT}$, respectively. The three procedures of IT-Player are described as follows.

- *it_ sched()*:
  - At the first negotiation round, the IT-Player finds x^{IT} by generating multiple feasible scheduling solutions (called candidates) then selects the best solution based on their utility. The generation of scheduling solutions is described in section 3.
  - At later rounds, the selection needs to satisfy another condition: $d(x, x^{PD}) < d(\ddot{x}^{IT}, x^{PD})$. This condition implies that the new aspiration order must be closer to x^{PD} than the aspiration order of previous negotiation round.
- *it_ place_ order()*: the IT-Player finds $\hat{x}$ that maximizes its utility as in equation 14. The computation of $\hat{x}$ is given as

$$\hat{x} = \arg\max_{x} u(x, \pi). \tag{15}$$

- *it_ est_ price()*: the IT-Player proposes a more attractive price to the PD-Player, while keeping the IT-Player's total utility non-decreased. The price is proposed based on the following rationales.
  - We have the fact that, from our definition, the aspiration order is the IT-Player desired order; hence, the revenue associated with this order is higher than the revenue associated with other orders.

– Based on this fact, the IT-Player estimates the revenue that it can gain if it runs the users' jobs using the power equal to the *aspiration order*, instead of the *placed order*.

Based on those two rationals, the IT-Player computes an incentive price that possibly increases the IT-Player cost, but the IT-Player total utility has to be non-decreased, i.e., $u(x^{IT}, \pi^{IT}) \geq u(\hat{x}, \pi)$. This incentive price implies that the IT-Player is willing to pay this price, if the PD-Player supplies the power equal to the aspiration order. Denoting $p = \{p_1, p_2, ..., p_T\}$ as a temporary variable, the pseudo-code for computing π^{IT} is as follows.

$$
\begin{aligned}
& p = \pi^{IT} = \ddot{\pi}^{IT} \\
& \textit{while} \;\; u(x^{IT}, p) \geq u(\hat{x}, \pi) \\
& \qquad \pi^{IT} = p \\
& \qquad p_i = p_i + \frac{p_i}{R}, \qquad i = 1, ..., T,
\end{aligned}
$$

where $\ddot{\pi}^{IT}$ is the IT incentive price of the previous negotiation round; R is a positive integer, which is set through experiment parameters.

7.2. PD-Player algorithm

Algorithm 4 Procedures of PD-Player

procedure follow_pd()

  Send a message of $\{pd\_pre, mod\}$, and wake up the IT-Player Loop

procedure follow_it()

  $x^{PD} \leftarrow$ pd_sched()

  $\pi \leftarrow$ pd_propose_price()

  $\pi^{PD} \leftarrow$ pd_est_price()

  Send a message of $\{x^{PD}, \pi, \pi^{PD}, pd\_pre, mod\}$, and wake up the IT-Player Loop

We show two procedures *follow_pd()* and *follow_it()* of the PD-Player in algorithm 4; and the PD-Player loop in algorithm 5. The PD-Player loop can

Algorithm 5 PD-Player Loop

$\alpha \leftarrow 0$, $\tau \leftarrow 1$
while $d(x^{PD}, \hat{x}) > \varepsilon$ and $\tau \leq ITER$ **do**
    if pd_mode = FLW_PD **then**
        if it_pre = FLW_IT **then**
            $\alpha \leftarrow \alpha + \gamma$
        else
            follow_pd()
    else
        if it_pre = FLW_IT **then**
            follow_it()
        else
            if mod = FLW_IT **then**
                follow_it()
            else
                follow_pd()
    if $r(x^{IT}, \pi^{IT}) + \alpha > r(\hat{x}, \pi)$ **and** $d(x^{PD}, \hat{x}) > \varepsilon$ **then**
        $pd\_pre \leftarrow$ FLW_IT
    else
        $pd\_pre \leftarrow$ FLW_PD
    $mod \leftarrow$ update_mode(it_pre, pd_pre, mod)
    $\tau++$
    Sleep until received new message from IT-Player

be described similarly to the IT-Player loop.

Denoting $x = \{x_1, x_2, ..., x_T\}$ as a PDM profile, the utility function of the PD-Player is given as

$$u(x, \pi, S) = r(x, \pi) - c(x, S) = \sum_{i=1}^{T} \pi_i x_i - c(x, S), \tag{16}$$

where $c(x, S)$ is computed as in equation (8). Unlike in SAN where the revenue is set $r(\cdot) = 1$, the revenue in GAN is $r(x, \pi) = \sum_{i=1}^{T} \pi_i x_i$.

7.2.1. PD-Player procedures

Similar to the IT-Player procedures, we denote the aspiration supply, price and PD incentive price of the previous negotiation round as $\ddot{x}^{PD}$, $\ddot{\pi}$ and $\ddot{\pi}^{PD}$, respectively. The three procedures of the PD-Player are described as follows.

- *pd_ propose_ price()*: each value π_i is generated such that it is inversely proportional to the value x_i^{PD}. We denote Z as the base price of electricity; in experiment, we set $Z = 0.17$ Euros/kWh. The computation of π is given as follows.

$$\begin{aligned} &if \;\; x_i^{PD} >= \bar{P}_i \\ &\qquad \pi_i = Z \\ &else \\ &\qquad \pi_i = Z \frac{\bar{P}_i}{x_i^{PD}}, \quad i = 1, ..., T, \end{aligned}$$

  where $\bar{P} = \{\bar{P}_1, ..., \bar{P}_T\}$ is the PD-Player average power supply; these values are stored and updated regularly by the PD-Player. In this computation, $x_i^{PD}, i = 1...T$ and $\bar{P}_i, i = 1...T$ are normalized to (0,1].

- *pd_ est_ price()*: similar to the procedure *it_ est_ price()*, the PD-Player proposes an incentive price that is attractive to the IT-Player, while keeping PD-Player's total utility non-decreased. We have defined that the aspiration supply is the PD-Player desired supply, hence the cost associated with this supply is lower than the cost associated with other supplies. Based on this fact, the PD-Player estimates the amount of cost

that it can reduce if it provides the IT-Player with the power equal to *aspiration supply*, instead of the *placed order*. Then the PD-Player computes an incentive price such that its total utility is non-decreased, i.e., $u(x^{PD}, \pi^{PD}) \geq u(\hat{x}, \pi)$. This computation implies that the PD-Player is willing to offer this price, if the IT-Player purchases the power equal to the aspiration supply. Denoting $p = \{p_1, p_2, ..., p_T\}$ as a temporary variable, the pseudo-code for computing π^{PD} is as follows.

$$
\begin{aligned}
& p = \pi^{PD} = \ddot{\pi}^{PD} \\
& while \;\; u(x^{PD}, p) \geq u(\hat{x}, \pi) \\
& \qquad \pi^{PD} = p \\
& \qquad p_i = p_i - \frac{p_i}{R}, \qquad i = 1, ..., T,
\end{aligned}
$$

where $\ddot{\pi}^{PD}$ is the PD incentive price of previous negotiation round; R is a positive integer as in the IT-Player algorithm.

- *pd_ sched()*:
  - Similar to the IT-Player, at the first negotiation round, the PD-Player finds x^{PD} by generating multiple feasible scheduling solutions (called candidates) then selects the best solution based on their utility. The generation of the scheduling solutions is based on equation (9) of section 3.
  - At later rounds, this procedure needs to satisfy two other conditions: (i) x^{PD} must be closer to x^{IT} than the aspiration supply of previous negotiation round $\ddot{x}^{PD}$, i.e., $d(x, x^{IT}) < d(\ddot{x}^{PD}, x^{IT})$, and (ii) the new price π must be closer to π^{IT} than the price of previous negotiation round $\ddot{\pi}$, i.e., $d(\pi, \pi^{IT}) < d(\ddot{\pi}, \pi^{IT})$.

In order to provide more explanations on the algorithms, we present a simplified numerical example of the negotiation process in Appendix B.

7.3. Properties of the proposed game

7.3.1. General properties

As mentioned in section 5, the proposed game is neither a cooperative game nor a non-cooperative game. This game can also be categorized as an *integrative*, *win-win* or *problem solving* bargaining game Pruitt (1981), Graham (1986), Lewicki et al. (2011), Lewicki (1981). In this integrative bargaining game, the benefits of both players are addressed by a cooperative and information-exchange oriented negotiation process. This is a well-known method to find a win-win agreement between suppliers and buyers. Without defining target and minimal outcomes, the players view the negotiation as a "problem to be solved", then mutually seek for a win-win agreement through negotiation Lewicki et al. (2011), Lewicki (1981). As a results, one player's gain does not necessarily come at the expense of the other player's gain.

The proposed game has some similar properties to the *centipede* game Binmore (1987), Osborne and Rubinstein (1994). In brief, *centipede* game is a sequential game in which two players play alternatively, choosing their strategy on a shared resource. The two main similar properties between the proposed game and centipede game are as follows. First, in our game, both players alternately have the option to stop negotiating Osborne and Rubinstein (1994). When a player prefers to stop negotiating, if it does stop, its utility is kept non-decreased. Second, the players have the chance to achieve higher utility if they continue. A player continues because it expects to obtain higher utility, in contrast, a player stops because it is afraid that the next move of the opponent will not be beneficial to it. Although there are two similar properties, the proposed game and the centipede game have a major difference. The players in the centipede game have the same set of two predefined discrete strategies, whereas the players in the proposed game have different continuous strategy sets.

7.3.2. Equilibrium

Definition 1: *The incentive to unilaterally deviate*: a player has *incentive to unilaterally deviate* if and only if the condition $d(x^{PD}, \hat{x}) > \varepsilon$ holds.

Definition 2: *Equilibrium*: the game reaches *equilibrium* when the condition $d(x^{PD}, \hat{x}) > \varepsilon$ does not hold for both players.

We note that, the equilibrium in Definition 2 is *Nash equilibrium* Osborne and Rubinstein (1994) because of followings. When the $d(x^{PD}, \hat{x}) > \varepsilon$ does not hold for both players, we reach equilibrium (Definition 2), and at the same time, the two players have no incentive to unilaterally deviate (Definition 1). On the other hand, the state when each player does not want to unilaterally deviate (given the current strategies of other players are unchanged) is called Nash equilibrium Osborne and Rubinstein (1994). This means that the equilibrium we have reached is the Nash equilibrium.

Proposition 1: Consider algorithm 3 and algorithm 5,

- (i) given that the scheduling relaxation can be arbitrarily varied,
- (ii) if these two algorithms run for a large enough *number of iterations*,

then the DMs' scheduling solution is ergodic in $[x^{IT}, x^{PD}]$ given a pre-defined granularity, and these algorithms converge to equilibrium.

Proof See Appendix A.

8. Experimental results

8.1. Setup

To evaluate GAN, we conduct various experiments using our middleware of datacenter powered by renewable energies. The middleware system is composed of a negotiation controller, an IT sub-system, and an electrical sub-system; they connect with each other through ActiveMQ [3]. We implement the ITDM using DCWorms Kurowski et al. (2013), a tool for simulating distributed computing systems. More description about this middleware can be found in our previous article Pierson et al. (2019). When running the negotiation algorithm, at each negotiation round, two representative profiles from IT-Player and PD-Player are

[3] www.activemq.apache.org

logged for evaluation. To have these two profiles, we use the placed order and the aspiration supply. The time window is set to 72 (hours), corresponding to 3 days. We run the experiments on Taurus cluster of Grid5000 platform [4] and on a local computer. The local computer is used to run the experiments relating to execution time. This local computer has one Intel ® processor 2.20GHz with 4 cores, and 8.27GB memory. Pearson correlation is used as the default implementation of profile distance. With some abuse of notation, we use M and N to denote the number of candidates in the scheduling of ITDM and PDM, respectively. In case there is not specific information provided, the default number of candidates is $M = N = 36$, and the default value of γ is 50. The default maximum number of iteration is $ITER = 18$.

The workload and weather information is trace data. The workload is a set of jobs, each has the information of arrival time, resource consumption over time, due date, and service grades. The traces of the workload are generated from a generator that is introduced in our previous research Caux et al. (2018). We use Google based workload generator to generate jobs for 72-hours time window. This means that the datacenter has to execute these jobs within 72 hours, otherwise, the jobs are expired. However, we use a different workload set with Caux et al. (2018) (which has only batch jobs); our workload includes 312 jobs, with 156 batch jobs and 156 service jobs. The trace of solar radiation is from the National Solar Radiation Database [5], and the trace of wind is from the wind prospect database [6], both traces are of Los Angeles in August 2002. The parameters of electrical infrastructure are summarized in table 6. Our research targets the datacenters that have 1MW peak of power demand, which were mentioned in our previous article Pierson et al. (2019). This kind of datacenter is popular in enterprises and public clouds. As a result, the sizing of the experimented datacenter is 1MW. This sizing is based on our other re-

[4] www.grid5000.fr

[5] www.nrel.gov/rredc/solar_data.html

[6] www.maps.nrel.gov/wind-prospector

Components	Parameters	Values
wind turbines	number of turbines	2
	maximum power per turbine	1800W
photovoltaic panels	number of panels	14
	area per panel	7.8 m^2
fuel cell	maximum power	2740W
	minimum power	0W
electrolyzer	maximum power	2740W
	minimum power	600W
batteries	number of batteries	2
	capacity per battery	2500Wh
	maximum charging/discharging power	1486W

Table 6: Setup parameters of electrical infrastructure

search Haddad et al. (2017), which focuses on the sub-topic *Infrastructure sizing* inside the DATAZERO project. We also note that the negotiation algorithm works based on the relative relationship between the power of ITDM and PDM, instead of absolute values of power. In other words, the absolute values of both ITDM and PDM power can be scaled up or down together. As a result, with other sizes of datacenter, as long as the ITDM and PDM are sized relatively in the proper order of magnitude, the algorithm can work. Other information about the setup and configurations of PDM and ITDM can be seen in Haddad et al. (2019), Caux et al. (2018) and Pierson et al. (2019).

We conduct various experiments to verify the convergence (section 8.2) and stability (section 8.3) of the proposed algorithms, then compare their performance with other algorithms (section 8.4). In order to verify the convergence,

generally we show the distance between ITDM and PDM profiles over time. To verify the stability, we perform multiple executions, then estimate the standard deviation among those executions in term of execution time and generational distance. Finally, we compare the performance of SAN and GAN with two other algorithms, namely *SAN-IT* and *GreenSlot* Goiri et al. (2011). The algorithm *SAN-IT* is a modified version of SAN, in which the scheduling of PDM is not considered. *GreenSlot* is an algorithm that schedules workload in time based on the information of predicted available renewable energy and grid electricity price.

8.2. Convergence

Fig. 12 depicts the performance of Pearson and MSE in term of distance between ITDM and PDM profiles, over negotiation rounds, in 3 different values of the sacrifice step-size γ. In the experiments about convergence, we set the distance threshold ε to an extremely low value, in order to prevent the algorithm from stopping before reaching the maximum number of iteration, i.e., $ITER = 18$. In general, the curves drop gradually over time, but with different paces. We can see that the curve $\gamma = 100$ fluctuates dramatically, while the curve $\gamma = 20$ drops slowly, and therefore, reaches stable state slowly. The explanation is that when γ is small, α is reduced slowly, then the sacrifice mechanism needs more time to take effect. When $\gamma = 50$, the algorithm provides more proper result than two others.

We define *power violation* between an ITDM profile and a PDM profile is the total amount of power that the ITDM profile exceeds the PDM profile. Fig. 13 presents the power violation of two implementations of distance, namely Pearson and MSE. We show the results of three scenarios: $M = N = 4$, $M = N = 12$, and $M = N = 20$. We can see that the curves generally drop over time. When $M = N = 12$ or $M = N = 20$, the curve fluctuates more than when $M = N = 4$, but reaches stable state faster. With higher number of candidates, we reach the stable state faster at the cost of running time. We can see that, with proper settings, the algorithm converges in about 12 negotiation rounds (Fig. 12 and

Fig. 13).

In Fig. 12 and Fig. 13, we also see that Pearson has better performance than MSE in term of convergence. This can be explained by the fact that, as discussed earlier, the Pearson correlation is able to capture the change in the order of power levels. Note that, to be comparable, all the curves showed in the figure are in Pearson measurements, even with the curve of MSE implementation. Specifically, in the implementation of MSE, we only use MSE measurement when comparing two profile inside the algorithm; after obtaining the solution for each round, we apply Pearson measurement on that solution.

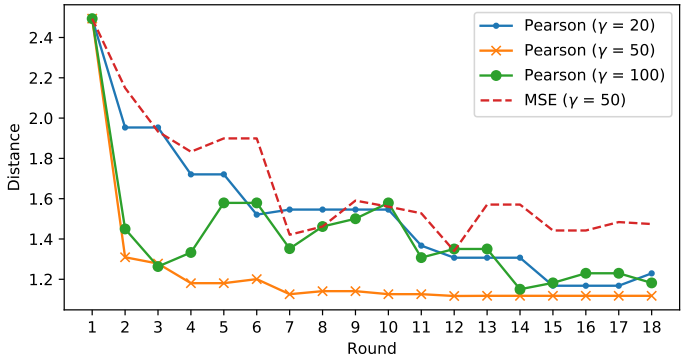


Figure 12: Profile distance over negotiation round

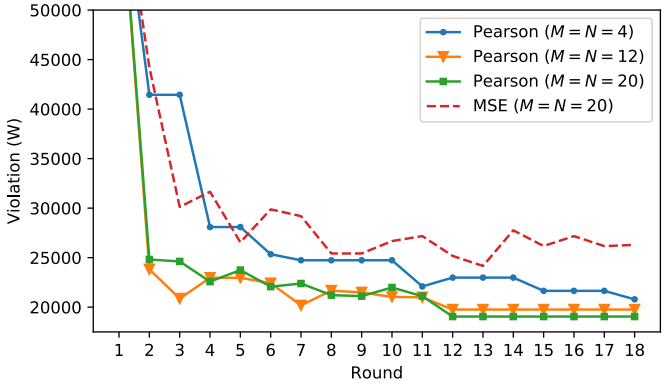


Figure 13: Power violation over negotiation round

In Fig. 14, we illustrate the evolution of sacrifice variable α over time. In this

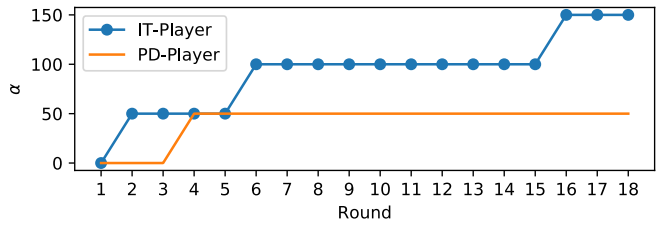


Figure 14: The change of α over negotiation round

experiment, we set $\gamma = 50$, so α is increased by 50 each time. The corresponding utilities of two players are showed in Fig. 21. We note that, due to the sacrifice mechanism, the utility is not always increased. When the IT-Player utility increases, the PD-Player utility tends to be decreased (Fig. 21).

8.3. Stability

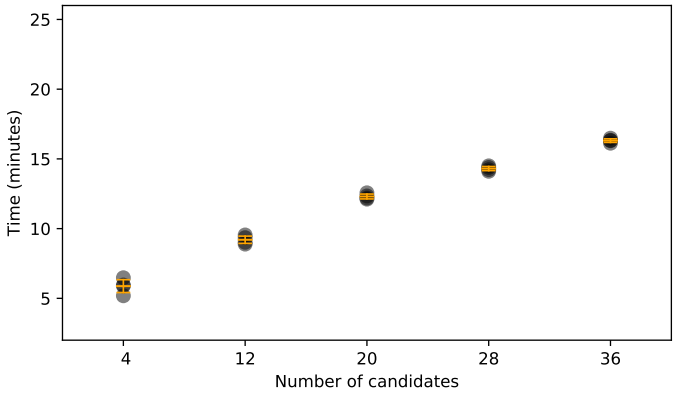


Figure 15: Execution time over the number of candidates

Fig. 15 depicts the execution time of each execution, categorized by the number of candidates. Note that as defined in section 3, a candidate is a profile corresponding to a feasible scheduling solution. When performing scheduling, the schedulers generate multiple feasible scheduling solutions, then select the final solution based on the criteria described in $it\_sched()$ and $pd\_sched()$. In the experiments about execution time, we fix the stopping criteria of the negotiation algorithm by setting the distance threshold $\varepsilon = 1.12$. Table 7 summarizes

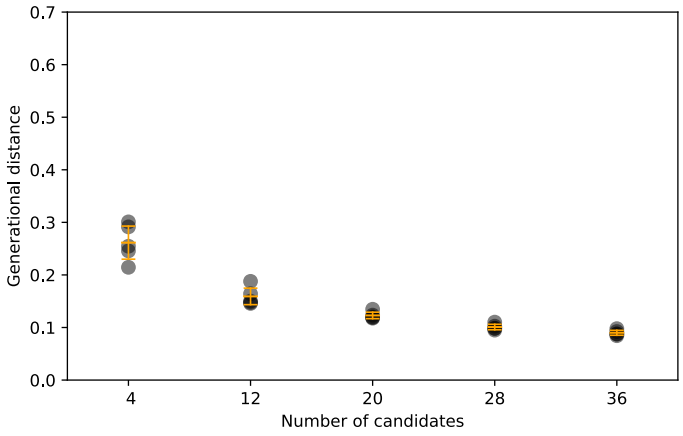


Figure 16: Generational distance over the number of candidates

the mean execution time of GAN in four scenarios of candidate. The table shows that the execution time of GAN does not grow exponentially with the number of candidates; instead, the growth is nearly linear.

Fig. 16 depicts the generational distance of the negotiation solution after 18 rounds. Compared to SAN (Fig. 3 and Fig. 6), the approach GAN (Fig. 15 and Fig. 16) achieves more stable execution time and generational distance. Table 8 summarizes the performance of SAN and GAN in term of standard deviation on execution time (stand. dev. on exe. time) and standard deviation on generational distance (stand. dev. on gen. dist) over different number of candidates (no. of cand.). We generate this table using the data in Fig 3, Fig. 6, Fig. 15, and Fig. 16. As an example, the SAN's standard deviation on execution time of *28 candidates* scenario is 3.4225, and the deviation on generational distance is 0.1453; whereas for GAN, those deviations are 0.1487 and 0.0052, respectively. This means that the deviations of GAN are lower than SAN by 23.0161 and 27.9423 times. In general, with the scenarios of *28 candidates* or higher, the execution time and generational distance of GAN are lower than SAN more than 20 times. Unlike GAN, the DMs in SAN have less information about direction to negotiate; hence, in some cases, the negotiation process may takes long time.

Number of candidates	**4**	**12**	**20**	**28**	**36**
Mean execution time	5.88 mins	9.2 mins	12.3 mins	14.31 mins	16.31 mins

Table 7: Execution time of GAN

Metrics	**No. of cand.**	**SAN**	**GAN**	**SAN÷GAN**
Stand. dev. on exe. time	4	8.0451	0.4609	17,4552
	12	4.3564	0.2608	16,704
	20	3.7724	0.1741	21,668
	28	3.4225	0.1487	23,0161
	36	2.5734	0.1284	20,0421
Stand. dev. on gen. dist.	4	0.1597	0.0315	5,0698
	12	0.1069	0.0157	6,8089
	20	0.0876	0.0062	14,129
	28	0.1453	0.0052	27,9423
	36	0.1002	0.0047	21,3191

Table 8: Execution time and generational distance of SAN and GAN

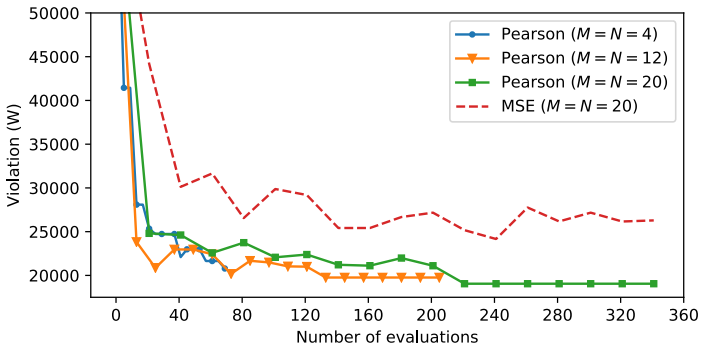


Figure 17: Power violation over number of evaluations

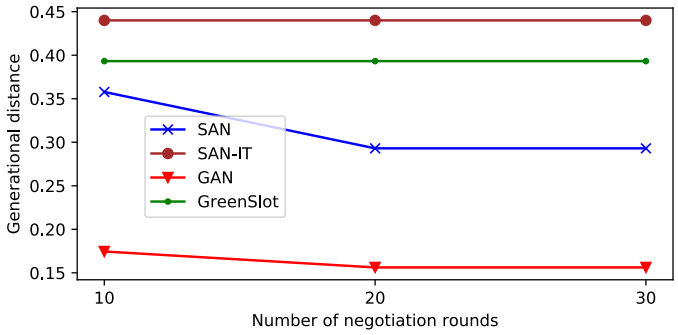


Figure 18: Generational distance over the number of negotiation rounds

When scheduling, the schedulers generate multiple feasible scheduling solutions, called candidates, then assess these candidates to select proper ones. We define that an *evaluation* is an assessment of one candidate when selecting the proper candidate. The number of evaluations is affected by the number of candidates (i.e., M and N) and the number of negotiation rounds. For instant, with $M = N = 4$, the number of evaluation after 18 negotiation round is 72. Fig. 17 shows the number of evaluations of GAN, with different scenarios of M and N. This result can help to understand the complexity of the proposed algorithm, if this algorithm is compared to other algorithms that are also based on generating multiple solutions, e.g., genetic algorithm. From this figure, we

can see that, since the curves "Pearson (M = N = 20)" and "Pearson (M = N = 12)" in Fig. 13 require higher number of evaluations, they reach stability sooner than the curve "Pearson (M = N = 4)". Unlike the Pearson curves, the MSE curve cannot reach stability although it also requires a high number of evaluation.

We compare our approaches with the "GreenVarPrices" version of the algorithm GreenSlot Goiri et al. (2011), which is implemented into our system with several simplifications and modifications. These modifications allow GreenSlot to have the same machine and workload model with ours. The modifications are: (i) the renewable sources are predicted without error, (ii) the job execution time is predefined, there is no time tolerance, (iii) if the due date of a job cannot be guaranteed, we still admit and schedule that job after its due date, and (iv) a machine may run multiple jobs at a same time, instead of only one job as in GreenSlot.

We modify SAN to have the new approach *SAN with only ITDM*, named *SAN-IT*. In SAN-IT, the scheduling solution from the ITDM is used as the final solution, without negotiating with PDM. The NM requests the ITDM to schedule with minimum relaxation, then the ITDM computes and returns the final solution. The power constraint $l^{(PD)}$ of the ITDM scheduler is set to the maximal available energy, meaning that the ITDM is allowed to used maximal available energy without regarding to the PDM cost. We implement the approach *SAN with only ITDM*, instead of *SAN with only PDM*, in order to have comparable results with GreenSlot (which focuses on IT scheduling, while taking into account the prediction of available energy). In Fig. 18, we compare the generational distance of SAN, SAN-IT, GAN, and GreenSlot. Note that the result of SAN in Fig. 18 is identical to the curve $M = 36$ in Fig. 5. The generational distance of SAN-IT and GreenSlot in Fig. 18 are constant because they are one-shot algorithms. We can see that SAN-IT provides lower performance than others because it only considers ITDM scheduling. SAN and GAN have highest performance, and they both converged before 20 negotiation rounds.

In Fig. 21, Fig. 19, and Fig. 20, we compare the utility, revenue and cost of four approaches. Each approach has one ITDM curve and one PDM curve, each curve is named by the the approach name followed by "ITDM" or "PDM"; for example, the ITDM curve of SAN is "SAN - ITDM". With some abuse of notation, we also name the curves of GAN as "GAN - ITDM" and "GAN - PDM", instead of "GAN - IT-Player" and "GAN - PD-Player". To make the performances of four approaches more comparable, we compute the utility, cost and revenue of GreenSlot, SAN-IT and SAN using the same formulation with GAN. Unlike in the SAN formulation where the cost of ITDM is set to zero, and the revenue of PDM is set to 1, in these experiments, we compute this cost and revenue similar as in GAN. Specifically, GreenSlot, SAN-IT and SAN call the GAN's procedures that compute utility, cost and revenue. As defined in section 3, the utility is the difference between the revenue and the cost. Note that the ITDM cost in Fig. 20 is identical to the PDM revenue in Fig. 19, because the ITDM pays to the PDM. The utility, cost and revenue are all measured in *Euro*.

The figures show that, in each negotiation round, if the curve "SAN - ITDM" raised or dropped, the curve "SAN - PDM" unchanged, and vice versa. This is the effect of turn-based negotiation. However, we cannot see this effect in the curves of GAN because of following reasons.

- First, for GAN, a *negotiation round* in the FLW_PD mode is slightly different with a *negotiation round* in the FLW_IT mode. In the FLW_PD mode, a negotiation round is finished after the IT-Player sends a message to the PD-Player. However, in the FLW_IT mode, a negotiation round is finished after the PD-Player sends a message to the IT-Player and the IT-Player finishes recomputing new order $\hat{x}$.

- Second, unlike SAN where each DM shows its utility, revenue and cost with respect to its own profile (i.e., the profile with highest utility), the players in GAN show their utility, revenue and cost with respect to a common profile (i.e., the order $\hat{x}$). Specifically, the revenue and cost of

IT-Player are from $r(\hat{x})$ and $c(\hat{x}, \pi)$, respectively; similarly, the revenue and cost of PD-Player are from $r(\hat{x}, \pi)$ and $c(\hat{x}, S)$, respectively.

In the figures, we can see that, because SAN-IT only considers ITDM scheduling, its ITDM has high revenue and low cost, while, in contrast, its PDM has low revenue, and high cost. As a result, its PDM utility is low, and even negative. Compared to SAN-IT, GAN provides a higher fairness between the utilities of ITDM and PDM. Among all four approaches, GAN, in general, achieves the highest performance in utility, though its cost is not always lowest. On the other hand, as showed in Fig. 14, the IT-Player sacrifices more than PD-Player, so the curves of IT-Player's revenue and IT-Player's cost fluctuate more than PD-Player, making the IT-Player's utility vary more than PD-Player's utility.

In Fig. 22, Fig. 23 and Fig. 24, we show the SLA violation of batch jobs, service jobs and overall violation, respectively. In each figure, we present the SLA violation of the final negotiation solutions of each approach. With batch jobs, we define the SLA violation as the rate at which the jobs finish their execution after their due dates. With service jobs, the SLA violation is the rate at which the computing resource received is lower than the computing resource requested. Denoting the number of violated batch jobs is b, we have the SLA violation as $\frac{b}{J^{(s)}}$. Similarly, denoting the number of violated service jobs is s, we have the SLA violation as $\frac{s}{J^{(b)}}$. In this way, the overall SLA violation is $\frac{b+s}{J}$, where $J = J^{(b)} + J^{(s)}$. Though SAN-IT has generally low performance in other metrics, it has higher performance than GreenSlot in SLA violation of both service and batch jobs, because it only considers ITDM scheduling without regarding to the PDM. However, SAN-IT cannot have higher performance than GAN and SAN because when it disregards PDM scheduling, there is a higher possibility that the power supply mismatches the power demand, then the performance of datacenter is degraded. GAN and SAN have the highest performance, achieving around 0.23% and 0.96% overall violation, respectively.

We summarize the performances of all approaches in table. 9, with different scenarios of distance threshold and number of negotiation rounds. We show the

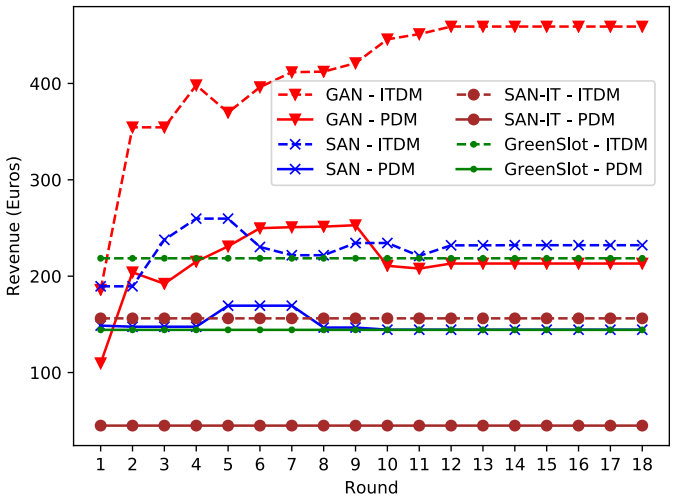


Figure 19: Revenue over negotiation round

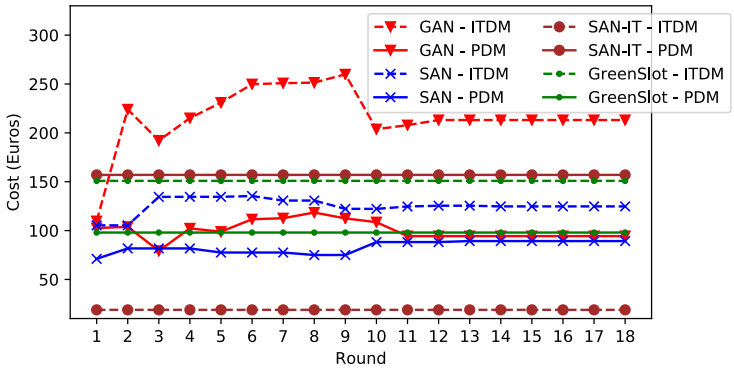


Figure 20: Cost over negotiation round

metrics of average execution time (avg. exe. time), average profile distance (avg. prof. dist.), average generational distance (avg. gen. dist.), average utility (avg. util.), total utility (tot. util.), and average overall SLA violation (avg. over. SLA violat.). We define the total utility as the sum of ITDM utility and PDM utility. Note that the average profile distance of GreenSlot and SAN-IT is not showed in the table because in these approaches, there are not two separate profiles of ITDM and PDM, so we cannot measure their distance. The table shows that GAN outperforms other approaches in most metrics, except the average

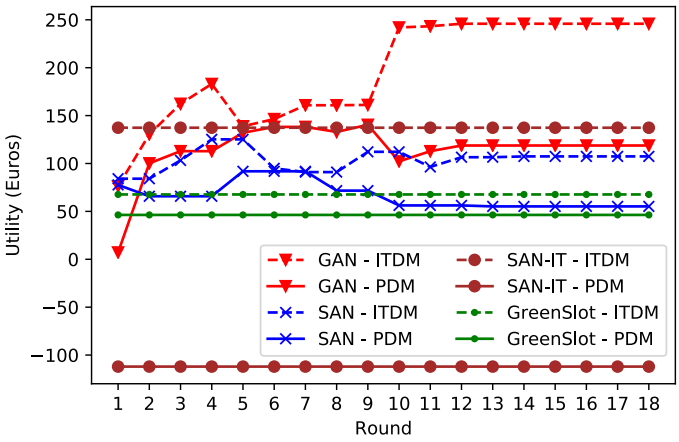


Figure 21: Utility over negotiation round (the higher the better)

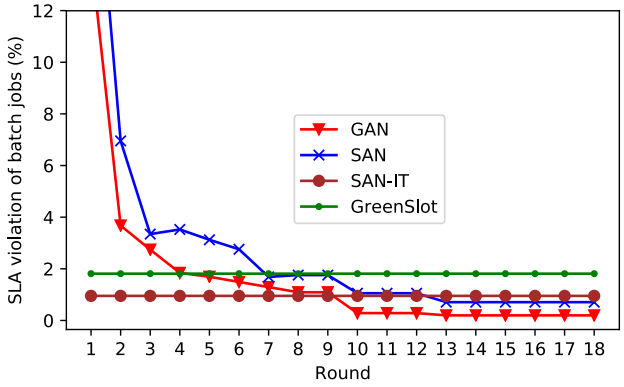


Figure 22: SLA violation of batch jobs

execution time. Specifically, compared to other approaches, GAN achieves lower average profile distance, lower average generational distance, higher total utility, and lower average overall SLA violation. The GreenSlot and SAN-IT have small average execution time because they use a one-shot algorithm, instead of a sequential algorithm. SAN-IT obtains negative PDM utility because it disregards the electrical management. Within a same approach, with lower value of distance threshold ε, the performance tends to be higher but the average execution time tends to be larger.

Scenario	Avg. exe. time (min)	Avg. prof. dist.	Avg. gen. dist.	Avg. util. (ITDM/ IT-Player, PDM/ PD-Player) (Euro)	Tot. util. (Euro)	Avg. over. SLA vio- lat. (%)
GreenSlot	0.43		0.39	(67.67, 46.30)	113.97	2.95
SAN-IT	4.45		0.44	(137.34, -112.05)	25.29	2.57
SAN $\varepsilon = 1.2$, $M = N = 20$	6.08	1.20	0.42	(67.77, 40.29)	108.06	1.90
GAN $\varepsilon = 1.2$, $M = N = 20$	6.80	1.17	0.15	(150.95, 106.67)	257.62	0.88
SAN $\varepsilon = 1.2$, $M = N = 36$	9.49	1.18	0.31	(103.20, 55.40)	158.6	1.25
GAN $\varepsilon = 1.2$, $M = N = 36$	10.69	1.17	0.10	(185.88, 125.92)	311.8	0.81
SAN $\varepsilon = 1.12$, $M = N = 20$	10.13	1.12	0.38	(81.29, 40.01)	121.3	0.98
GAN $\varepsilon = 1.12$, $M = N = 20$	12.30	1.11	0.12	(178.22, 106.03)	284.25	0.49
SAN $\varepsilon = 1.12$, $M = N = 36$	15.84	1.11	0.23	(107.99, 55.17)	163.16	0.91
GAN $\varepsilon = 1.12$, $M = N = 36$	16.31	1.11	0.09	(245.90, 118.77)	364.67	0.23

Table 9: Summarized performance with fixed stopping criteria

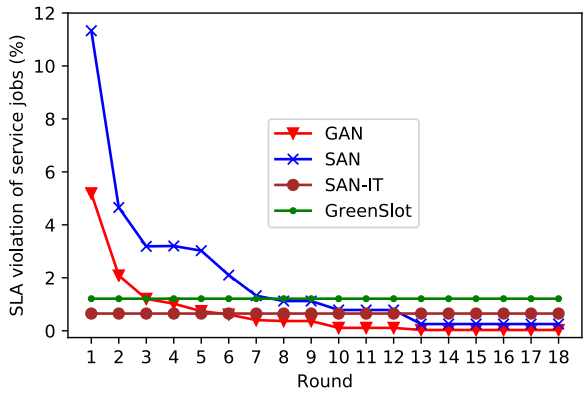


Figure 23: SLA violation of service jobs

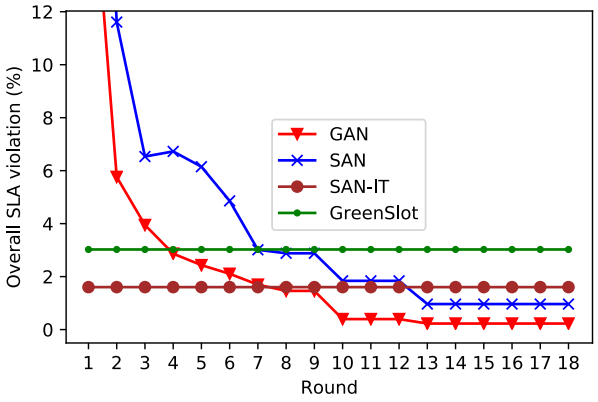


Figure 24: Overall SLA violation

9. CONCLUSIONS

In this paper, we investigate the problem of joint IT and energy management in datacenters entirely powered by renewable energies. The IT and electrical sub-systems, named ITDM and PDM, are modeled separately in order to take advantage of the distributed design, facilitating the developing and maintaining process. We aim to find an efficient compromise between power demand and power supply, while respecting the operational requirements of each sub-system. We propose a black-box and a semi black-box algorithm that allow the ITDM and PDM to negotiate. In addition, we design a generic model for the DMs,

then any scheduler whose implementation follows this generic model can work with the proposed negotiation algorithms. We found that the semi black-box approach achieves higher stability and performance than the black-box approach. Specifically, the black-box approach is prone to unstable performance in term of execution time and generational distance. This can be explained that, in the black-box approach, the DMs are not provided with the information about the direction to negotiate. Therefore, in some cases, this approach takes long time to converge. On the contrary, the semi black-box approach allows the game players to exchange information about the direction for negotiating, based on the framework of a buyer-supplier relationship. The analytical results prove that the proposed algorithm converges and the game reaches equilibrium. These results are confirmed by various experiments in a middleware. The experimental results show that the proposed game-theoretic algorithm provides stable solutions, and outperforms other approaches in term QoS and utility.

For future works, we plan to design a multi-timescale version of GAN, in which the algorithm makes prompt short-term decisions to deal with small-timescale events, while maintaining long-term objectives. This design is important because our system involves multiple sub-systems with different operational timescales. For example, the operational objective of the electrical sub-system may be in the order of years or quarters; while the environmental events can be in the order of months or days; and the IT workload events may be in the order of hours or minutes.

10. Acknowledgment

The work presented in this paper has been funded by the ANR in the context of the project DATAZERO, ANR-15-CE25-0012. Experiments presented in this paper were carried out on Grid5000 testbed, supported by a scientific interest group hosted by Inria, including CNRS, RENATER, several Universities and organizations (www.grid5000.fr).

Appendix A. Proof of Proposition 1

We have following facts.

- First, due to
  - (i),
  - (iii) incentive mechanism, and
  - (iv) the fact that α in algorithm 3 and algorithm 5 can be increased freely,

  the negotiation can be continued for an arbitrary number of iterations, therefore the condition (ii) holds;
- Second, the two constraints $d(x, x^{PD}) < d(\ddot{x}^{IT}, x^{PD})$ and $d(x, x^{IT}) < d(\ddot{x}^{PD}, x^{IT})$ inside the two procedures $it\_sched()$ and $pd\_sched()$ guarantee that x^{IT} approaches x^{PD}.
- Third, x^{IT} approaches x^{PD}, or in other words, $d(x^{PD}, x^{IT})$ always decreases. Moreover, the two algorithms have the same stopping criterion $d(x^{PD}, \hat{x}) \leq \varepsilon$, hence two negotiation loops terminate at the same time.

Based on the first and second facts, the DM scheduling solution is ergodic in $[x^{IT}, x^{PD}]$ given a pre-defined granularity.

Based on the third fact and the two definitions, algorithm 3 and algorithm 5 terminate at the point where no player wants to deviate, meaning that the game reaches equilibrium. ■

Appendix B. Example of Negotiation Process

In Fig. B.25, we provide a simplified example of negotiation process. The information in the *Common Space* is accessible to both players. The time window is set to $T = 2$. In the figure, the first column is the negotiation round, whereas the second column is the row index. In this example, we suppose that the aspiration supply is initially lower than the aspiration order. Specifically, we

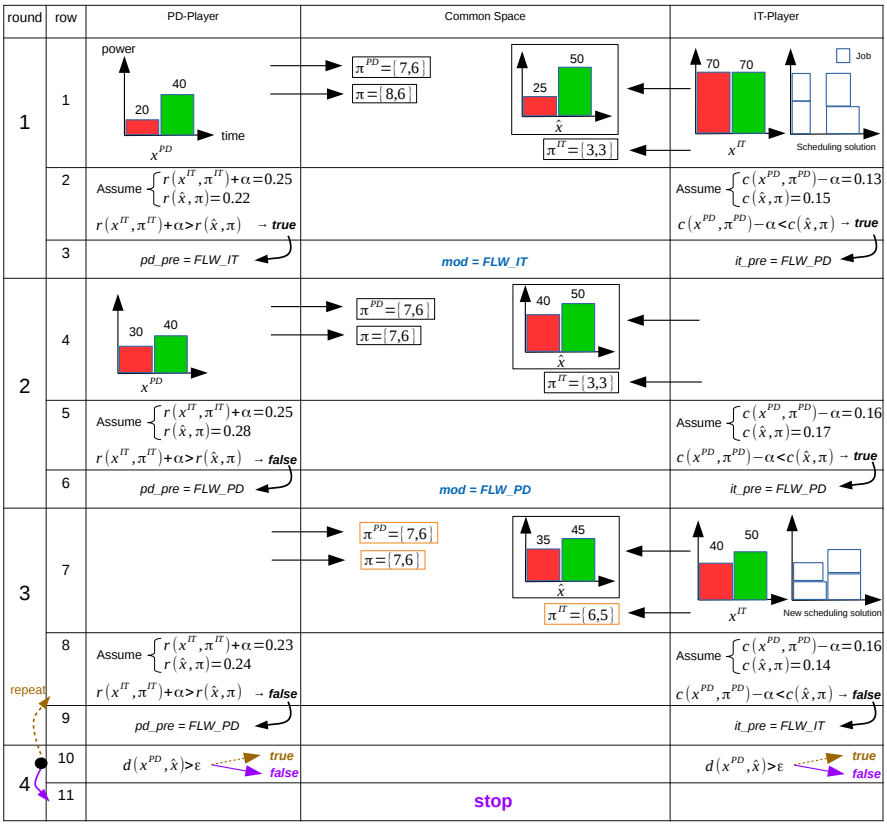


Figure B.25: An example of negotiation

suppose that $x^{PD} = \{20, 40\}$ and $x^{IT} = \{70, 70\}$. After one reschedule of the PD-Player and one reschedule of the IT-Player, $x^{PD} = \{30, 40\}$, $x^{IT} = \{40, 50\}$, and $\hat{x} = \{35, 45\}$. At that time, we assume that the condition $d(x^{PD}, \hat{x}) \leq \varepsilon$ holds, and the negotiation finishes. The details of the example are provided in the following descriptions.

- Round 1: We suppose that, initially, $mod = FLW\_IT$.
  - Row 1
    - PD-Player: We suppose that, after scheduling, the procedure $pd\_sched()$ obtains the aspiration supply $x^{PD} = \{20, 40\}$. On

the other hand, the price π is inversely proportional to its associated aspiration supply. As a result, the output of the procedure *pd_ propose_ price()* is $\pi = \{8, 6\}$. The PD-Player also tries to propose a PD incentive price that is attractive to the IT-Player. We suppose that the PD-Player is able to propose $\pi^{PD} = \{7, 6\}$, using the procedure *pd_ est_ price()*. Finally, the PD-Player sends π and π^{PD} to the IT-Player.

* IT-Player: We assume that, after scheduling (as in the chart *Scheduling solution*), the procedure *it_ sched()* obtains the aspiration order $x^{IT} = \{70, 70\}$. After receiving π from the PD-Player, the IT-Player places order using procedure *it_ place_ order()*. We suppose that the output of this procedure is $\hat{x} = \{25, 50\}$. The IT-Player places this order with the consideration of π and x^{IT}. In this scenario, we can expect that the power levels of $\hat{x}$ are between x^{PD} and x^{IT}; also, the power levels of $\hat{x}$ have similar curve with x^{PD}. Similar with PD-Player, the IT-Player tries to propose an attractive price to the PD-Player, using the procedure *it_ est_ price()*. We suppose that the IT-Player is only able to propose the IT incentive price $\pi^{IT} = \{3, 3\}$. Finally, the IT-Player sends $\hat{x}$ and π^{IT} to the PD-Player.

– Row 2: As in line 16 of the algorithm 3 and algorithm 5, both players verify whether it is beneficial to switch their mode. The PD-Player verifies the revenue, i.e., $r(x^{IT}, \pi^{IT}) + \alpha > r(\hat{x}, \pi)$, and the IT-Player verifies the cost, i.e., $c(x^{PD}, \pi^{PD}) - \alpha < c(\hat{x}, \pi)$. In the example, we assume that $r(x^{IT}, \pi^{IT}) + \alpha = 0.25$, $r(\hat{x}, \pi) = 0.22$, $c(x^{PD}, \pi^{PD}) - \alpha = 0.13$, and $c(\hat{x}, \pi) = 0.15$. As a result, both verifying conditions return *true*. The intuition behind this verification is as follows. We can call x^{IT} and π^{IT} as the offers from the IT-Player; similarly, we can call x^{PD} and π^{PD} as the offers from the PD-Player. The PD-Player verifies the attractiveness of the IT-Player offer by comparing

the revenue $r(x^{IT}, \pi^{IT}) + \alpha$ with the revenue $r(\hat{x}, \pi)$. Similarly, the IT-Player compares the cost $c(x^{PD}, \pi^{PD}) - \alpha$ and the cost $c(\hat{x}, \pi)$. Then, a player will follow the other player if the offers from the other player are attractive.

- Row 3: We suppose that, both comparisons in row 2 return *true*, then $pd\_pre = FLW\_IT$ and $it\_pre = FLW\_PD$. However, the global mode is unchanged, i.e., $mod = FLW\_IT$, since this mode is switched to FLW_PD only when both players switched to FLW_PD.

- Round 2:

  - Row 4:

    * PD-Player: Since the current global mode is FLW_IT, the PD-Player reschedules, trying to find a new aspiration supply whose associated price is closer to π^{IT}. We suppose that the PD-Player finds the aspiration supply $x^{PD} = \{30, 40\}$, and the associated price is $\pi = \{7, 6\}$. Similar to the row 1, the PD-Player also tries to find a PD incentive price that is attractive to the IT-Player. However, the PD-Player is unsuccessful at this time, and the PD incentive price is equal to π, i.e., $\pi^{PD} = \{7, 6\}$. Finally, the PD-Player sends π and π^{PD} to the IT-Player.

    * IT-Player: The IT-Player, after estimating its utility, will place an order $\hat{x} = \{40, 50\}$.

  - Row 5: We suppose that the comparison $r(x^{IT}, \pi^{IT}) + \alpha > r(\hat{x}, \pi)$ returns *false*; this means that the PD-Player no longer foresees benefit in following the IT-Player. Compared to row 2, we see that the term $r(x^{IT}, \pi^{IT}) + \alpha$ is unchanged at 0.25, but $r(\hat{x}, \pi)$ has increased to 0.28. This increase can be explained that, compared to row 2, $\hat{x}$ has grown significantly, whereas π has just reduced slightly. At the side of IT-Player, compared to row 2, we can see that both x^{PD} and $\hat{x}$ have increased. As a result, though π decreases slightly, both IT-Player

costs $c(x^{PD}, \pi^{PD}) - \alpha$ and $c(\hat{x}, \pi)$ increase, keeping the condition $c(x^{PD}, \pi^{PD}) - \alpha < c(\hat{x}, \pi)$ return *true* again as in row 2.

- Row 6: Because of the results in row 5, the PD-Player switches to the mode *FLW_PD*. At this time, both players already switched to the mode *FLW_PD*, so the global mode also switches to *FLW_PD*.

- Round 3:

  - Row 7: Since the global mode is *FLW_PD*, the IT-Player reschedules. In contrast with the objective of the PD-Player rescheduling, the objective of the IT-Player rescheduling is to find an aspiration order that is closer to x^{PD}. We suppose that the IT-Player found the new scheduling solution as in the chart *New scheduling solution*, then the aspiration order is $x^{IT} = \{40, 50\}$. With this new aspiration order, the IT-Player estimates its utility, then places an order $\hat{x} = \{35, 45\}$. Also, with the new aspiration order, the IT-Player recomputes and obtains the new IT incentive price $\pi^{IT} = \{6, 5\}$.
  - Row 8:
    - PD-Player: Compared to row 5, both x^{IT} and $\hat{x}$ have decreased, keeping the condition $r(x^{IT}, \pi^{IT}) + \alpha > r(\hat{x}, \pi)$ return *false* again as in row 5.
    - IT-Player: Unlike row 5, the comparison now returns *false*. We can explain this result as follows. Compared to row 5, x^{PD} and π^{PD} are unchanged, but $\hat{x}$ has decreased, making $c(\hat{x}, \pi)$ decrease from 0.17 to 0.14. As a result, $c(\hat{x}, \pi)$ is no longer larger than $c(x^{PD}, \pi^{PD}) - \alpha$.
  - Row 9: The variables of mode are updated according to the results at row 8.

- Round 4:

  - Row 10: Assume that we set $\varepsilon = 1.1$, now the Pearson distance between $x^{PD} = \{30, 40\}$ and $\hat{x} = \{35, 45\}$ is $d(x^{PD}, \hat{x}) = 1 < \varepsilon$,

meaning that the stopping criteria of both players are met. We note that these criteria are not only verified after round 3, but also after round 1 and round 2. However, these criteria were not met after round 1 and round 2.

- Row 11: The negotiation algorithm stops according to the results at row 10.

After negotiation, the PD-Player will implement the final solution $x^{PD} = \{30, 40\}$, and the IT-Player must accept this solution. In practice, we need to choose ε such that the gap between x^{PD} and x^{IT} is acceptable to IT-Player. We can summarize the effect of the negotiation process as follows. At row 1, through the price $\pi^{IT} = \{3, 3\}$, the IT-Player gives the PD-Player the information about the direction for negotiating, which is to increase the power level of the first time step or to decrease the power level of the second time step in x^{PD}, in order to comply with the aspiration order $x^{IT} = \{70, 70\}$ which has two equal power levels. However, according to the conditions in the procedure *pd_sched*(), the power level of the second time step cannot be decreased. As a result, the PD-Player reschedules and increases the power level of the first time step from 20 to 30. Similarly, through the price $\pi^{PD} = \{7, 6\}$, the IT-Player knows it should reschedule to reduce the power level of the first time step in x^{IT}, because the energy price of the first time step is high. And in fact, the IT-Player has reduced the power level of the first time step from 70 to 40.

References

W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, P. Demeester, Trends in worldwide ICT electricity consumption from 2007 to 2012, Computer Communications 50 (2014) 64–76.

S. S. Gill, R. Buyya, A taxonomy and future directions for sustainable cloud computing: 360 degree view, ACM Computing Surveys (CSUR) 51 (5) (2018) 104.

A. Andrae, T. Edler, On global electricity usage of communication technology: trends to 2030, Challenges 6 (1) (2015) 117–157.

A. Shehabi, S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, W. Lintner, United states data center energy usage report, Research Report LBNL-1005775, Lawrence Berkeley National Laboratory, Berkeley, California, available online at https://escholarship.org/uc/item/84p772fc, 2016.

C. Li, R. Wang, T. Li, D. Qian, J. Yuan, Managing Green Datacenters Powered by Hybrid Renewable Energy Systems, in: The 11th International Conference on Autonomic Computing, USENIX, 261–272, 2014.

E. Sheme, P. Stolf, G. Da Costa, J.-M. Pierson, N. Frashëri, Efficient Energy Sources Scheduling in Green Powered Datacenters: A Cloudsim Implementation, NESUS workshop .

L. Liu, C. Li, H. Sun, Y. Hu, J. Gu, T. Li, J. Xin, N. Zheng, HEB: deploying and managing hybrid energy buffers for improving datacenter efficiency and economy, in: ACM SIGARCH Computer Architecture News, vol. 43, ACM, 463–475, 2015.

I. Goiri, M. E. Haque, K. Le, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, R. Bianchini, Matching renewable energy supply and demand in green datacenters, Ad Hoc Networks 25 (2015) 520 – 534, doi:https://doi.org/10.1016/j.adhoc.2014.11.012.

H. Lei, T. Zhang, Y. Liu, Y. Zha, X. Zhu, SGEESS: Smart green energy-efficient scheduling strategy with dynamic electricity price for data center, Journal of Systems and Software 108 (2015) 23 – 38, doi:https://doi.org/10.1016/j.jss.2015.06.026.

A. Kassab, J.-M. Nicod, L. Philippe, V. Rehn-Sonigo, Assessing the Use of Genetic Algorithms to Schedule Independent Tasks Under Power Constraints,

in: 2018 International Conference on High Performance Computing & Simulation (HPCS), IEEE, 252–259, 2018.

X. Wang, Z. Du, Y. Chen, M. Yang, A green-aware virtual machine migration strategy for sustainable datacenter powered by renewable energy, Simulation Modelling Practice and Theory 58 (2015) 3–14.

T. Cioara, I. Anghel, M. Antal, S. Crisan, I. Salomie, Data center optimization methodology to maximize the usage of locally produced renewable energy, in: Sustainable Internet and ICT for Sustainability (SustainIT), 2015, IEEE, 1–8, 2015.

D. Paul, W.-D. Zhong, S. K. Bose, Demand response in data centers through energy-efficient scheduling and simple incentivization, IEEE Systems Journal 11 (2) (2017) 613–624.

G. Chonglin, L. Chunyan, Z. Jiangtao, H. Hejiao, X. Jia, Green scheduling for cloud data centers using renewable resources, in: Proc. IEEE INFOCOM 2015 Workshop on MCV, 2015.

C. Gu, C. Liu, J. Zhang, H. Huang, X. Jia, Green scheduling for cloud data centers using renewable resources, in: 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 354–359, doi:10.1109/INFCOMW.2015.7179410, 2015.

C.-M. Wu, R.-S. Chang, H.-Y. Chan, A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters, Future Generation Computer Systems 37 (2014) 141 – 147, ISSN 0167-739X, doi:http://dx.doi.org/10.1016/j.future.2013.06.009, URL http://www.sciencedirect.com/science/article/pii/S0167739X13001234, special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing Special Section: Semantics, Intelligent processing and services for big data Special Section: Advances in Data-Intensive Modelling and Simulation Special Section: Hybrid Intelligence for Growing Internet and its Applications.

S. Iturriaga, S. Nesmachnow, Scheduling energy efficient data centers using renewable energy, Electronics 5 (4) (2016) 71.

N. Beldiceanu, B. D. Feris, P. Gravey, S. Hasan, C. Jard, T. Ledoux, Y. Li, D. Lime, G. Madi-Wamba, J.-M. Menaud, P. Morel, M. Morvan, M.-L. Moulinard, A.-C. Orgerie, J.-L. Pazat, O. Roux, A. Sharaiha, Towards energy-proportional clouds partially powered by renewable energy, Computing 99 (1) (2017) 3–22, ISSN 1436-5057, doi:10.1007/s00607-016-0503-z, URL http://dx.doi.org/10.1007/s00607-016-0503-z.

Í. Goiri, W. Katsak, K. Le, T. D. Nguyen, R. Bianchini, Parasol and greenswitch: Managing datacenters powered by renewable energy, ACM SIGARCH Computer Architecture News 41 (1) (2013) 51–64.

I. Goiri, W. Katsak, K. Le, T. D. Nguyen, R. Bianchini, Designing and Managing Data centers Powered by Renewable Energy, IEEE Micro 34 (3) (2014) 8–16, ISSN 0272-1732, doi:10.1109/MM.2014.6, URL doi.ieeecomputersociety.org/10.1109/MM.2014.6.

I. D. Courchelle, T. Guérout, G. D. Costa, T. Monteil, Y. Labit, Green Energy efficient scheduling management, Simulation Modelling Practice and Theory ISSN 1569-190X, doi:https://doi.org/10.1016/j.simpat.2018.09.011, URL http://www.sciencedirect.com/science/article/pii/S1569190X18301382.

Y. Li, A. C. Orgerie, J. M. Menaud, Balancing the Use of Batteries and Opportunistic Scheduling Policies for Maximizing Renewable Energy Consumption in a Cloud Data Center, in: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 408–415, doi:10.1109/PDP.2017.24, 2017.

R. Roche, S. Caux, J. Lecuivre, J.-M. Pierson, D. Hissel, J.-M. Nicod, DATAZERO: Designing and Operating Datacenters Powered by Renewable Energy-Based Stand-Alone Microgrids, in: Journée Scientifique Nationale Micro-Réseaux, GDR SEEDS, 2017.

S. Caux, P. Renaud-Goud, G. Rostirolla, P. Stolf, IT Optimization for Datacenters Under Renewable Power Constraint, in: Euro-Par 2018: Parallel Processing - 24th International Conference on Parallel and Distributed Computing, Turin, Italy, August 27-31, 2018, Proceedings, 339–351, doi:10.1007/978-3-319-96983-1_24, URL https://doi.org/10.1007/978-3-319-96983-1_24, 2018.

M. Haddad, M.-C. Pera, C. Varnier, Stand-Alone Renewable Power System Scheduling for a Green Data-Center using Integer Linear Programming Version 1, Research Report, FEMTO-ST, URL https://hal.archives-ouvertes.fr/hal-02081951, 2019.

I. n. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, R. Bianchini, GreenSlot: Scheduling Energy Consumption in Green Datacenters, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11, ACM, New York, NY, USA, ISBN 978-1-4503-0771-0, 20:1–20:11, doi:10.1145/2063384.2063411, URL http://doi.acm.org/10.1145/2063384.2063411, 2011.

B. Aksanli, J. Venkatesh, L. Zhang, T. Rosing, Utilizing Green Energy Prediction to Schedule Mixed Batch and Service Jobs in Data Centers, in: Proceedings of the 4th Workshop on Power-Aware Computing and Systems, HotPower '11, ACM, New York, NY, USA, ISBN 978-1-4503-0981-3, 5:1–5:5, doi:10.1145/2039252.2039257, URL http://doi.acm.org/10.1145/2039252.2039257, 2011.

L. Grange, G. Da Costa, P. Stolf, Green IT scheduling for data center powered with renewable energy, Future Generation Computer Systems 86 (2018) 99–120.

A. Sayah, D. Hissel, P. Stolf, S. Caux, J.-M. Pierson, G. DaCosta, B. Celik, J. Nicod, Interactions between system modules and messages format, de-

liverable D3.1 Available from https://www.irit.fr/datazero/images_specific/Deliverables/M27-D3.1.pdf [accessed 1 Mar 2019], 2017.

D. A. Van Veldhuizen, G. B. Lamont, Evolutionary computation and convergence to a pareto front, in: Late breaking papers at the genetic programming 1998 conference, 221–228, 1998.

D. R. Krause, R. Terpend, K. J. Petersen, Bargaining stances and outcomes in buyer–seller negotiations: experimental results, Journal of Supply Chain Management 42 (3) (2006) 4–15.

R. Terpend, D. R. Krause, Competition or cooperation? Promoting supplier performance with incentives under varying conditions of dependence, Journal of Supply Chain Management 51 (4) (2015) 29–53.

L. C. Giunipero, Motivating and monitoring JIT supplier performance, Journal of Purchasing & Materials Management 26 (3) (1990) 19–25.

S. B. Modi, V. A. Mabert, Supplier development: Improving supplier performance through knowledge transfer, Journal of operations management 25 (1) (2007) 42–64.

D. G. Pruitt, Bargaining behavior, New York: Ac .

J. L. Graham, The problem-solving approach to negotiations in industrial marketing, Journal of Business Research 14 (6) (1986) 549–566.

R. J. Lewicki, D. M. Saunders, J. W. Minton, J. Roy, N. Lewicki, Essentials of negotiation, McGraw-Hill/Irwin Boston, MA, 2011.

R. J. Lewicki, Bargaining and negotiation, Exchange: The Organizational Behavior Teaching Journal 6 (2) (1981) 33–42.

K. Binmore, Modeling Rational Players: Part I: Ken Binmore, Economics and Philosophy .

M. J. Osborne, A. Rubinstein, A course in game theory, MIT press, 1994.

K. Kurowski, A. Oleksiak, W. Piatek, T. Piontek, A. Przybyszewski, J. Weglarz, DCworms – A tool for simulation of energy efficiency in distributed computing infrastructures, Simulation Modelling Practice and Theory 39 (2013) 135–151.

J.-M. Pierson, G. Baudic, S. Caux, B. Celik, G. Da Costa, L. Grange, M. Haddad, J. Lecuivre, J.-M. Nicod, L. Philippe, et al., DATAZERO: DATAcenter with Zero Emission and RObust management using renewable energy, IEEE Access .

M. Haddad, J.-M. Nicod, M.-C. Pera, Hydrogen infrastructure: data-center supply-refueling station synergy, in: 2017 IEEE Vehicle Power and Propulsion Conference (VPPC), IEEE, 1–6, 2017.