# Multi-objective cooperative scheduling: An application on smart grids

Khouloud Salameh, Richard Chbeir, Haritza Camblong

# Multi-objective Cooperative Scheduling: An Application on Smart Grids

Khouloud Salameh[1,2], Richard Chbeir[1], Haritza Camblong[2]

[1] University of Pau and Adour Countries, 64600 Anglet, France
email: {khouloud.salameh, richard.chbeir}@univ-pau.fr

[2] University of Basque Country, Donostia 20018, Spain
email: aritza.camblong@ehu.eus

# Multi-objective Cooperative Scheduling: An Application on Smart Grids

No Author Given

No Institute Given

**Abstract.** As the size of the Smart Grids ($SG$) grows, the economic significance of power generation, consumption and storage scheduling becomes more and more apparent. A proper scheduling for electricity generation, consumption and storage will also ensure the reliability of the $SG$ and extend the operational lives of its constituent units. Besides, it can achieve economical and ecological benefits for the $SG$. In this work, we propose a multi-objective cooperative scheduling consisting of two main modules: 1) the Preference-based Compromise Builder and 2) the Multi-objective Scheduler. The Preference-based Compromise Builder aims at generating the best balance or what we call 'the compromise' between the preferences or associations of the sellers and the buyers that must exchange power simultaneously. Once done, the Multi-objective Scheduler aims at proposing a power schedule for the associations, in order to achieve three-dimensional benefits: economical by reducing the electricity costs, ecological by minimizing the toxic gas emissions, and operational by reducing the peak load of the $SG$ and its components, and by increasing their comfort. Conducted experiments showed that the proposed algorithms provide convincing results.

**Keywords:** Multi-objective optimization, Smart Grid, Scheduling

## 1 Introduction

The Information and Communication technologies (ICT) represent unprecedented opportunities to move the power systems into a new era of reliability and efficiency that will contribute to operational, economical and ecological improvements. During this transition period, it is important to implement adequate techniques allowing to ensure that the benefits envisioned from the $SG$ become a reality. Commonly considered as a key mechanism towards a more efficient and cost effective $SG$, the Demand-Side Management or DSM [6, 14] refers to the planning and implementation of the utility companies' programs[1] designed to directly or indirectly influence the consumer consumption in the aim of reducing the system peak load and electricity costs. DSM techniques can be mainly gathered in two main categories: the load shifting [11] and the energy efficiency and conservation [2] programs.

[1] A utility company is a company that engages in the generation and the distribution of electricity for sale generally in a regulated market.

In our study, we focus on the load shifting, and more specifically on the power scheduling, since it has been observed that it is easier to motivate users to reschedule their needs rather than asking them to reduce their consumption [11]. Several approaches have been provided in the literature to address the power scheduling problem [10, 17, 13, 1, 5, 15]. However, and to the best of our knowledge, none of the them fully addresses the following challenges:

- **Operational Challenges**: Several limitations can be mentioned regarding the operational aspect:
  1. **The consumer discomfort**: while the consumers are enjoying their reduced electricity bills when shifting their consumption from on-peak to off-peak periods, they might risk discomfort related to the delay time of receiving their desired power.
  2. **The local peaks**: while trying to reduce the whole $SG$ peak load, it is essential to consider the individual $SG$ components' peak loads as well. This would conduct to increase the reliability of the components and decrease the local failure risks.
  3. **The consumption wise**: the current shifting programs [10, 17, 13, 5] provide consumption scheduling without considering the production nor the storage scheduling (with the exception of few approaches [1, 15]) which negatively impact their efficiency in shaping the peak load, reducing the electricity bills and minimizing the gas emissions effects.
- **Economical Challenges**: Knowing that the electricity price relies on the demand and supply over a specific period [9], an adequate scheduling is expected to shift loads during periods of high market prices (peak hours) and consequently minimize the electricity costs.
- **Ecological Challenges**: A significant power production from pollutant energy sources leads to a significant toxic gas emissions. Hence, it is essential to provide a power production scheduling allowing to reduce the bad emissions and effects on the environment by reducing the simultaneous toxic power production.

To address the aforementioned challenges, we introduce here $MOCSF$, a 'Multi-Objective Cooperative Scheduling Framework' designed for the power scheduling in the $SG$. $MOCSF$ aims at scheduling a set of couples, each consisting of a seller and buyer, working together in a mutual spirit so to ensure a better reduction of the economical, operational and ecological costs and impacts within the $SG$. In addition, our approach presents several advantages over existing approaches, namely:

1. It provides a scheduling coverage able to consider all of the power consumption, production and storage entities of the $SG$,
2. It considers multiple energy sources (unlike existing approaches [10, 17, 13, 1, 5, 15] that studied the interaction of the consumers with only one energy source),
3. It takes into account the $SG$ components' preferences unlike current approaches [17, 1, 15] that consider them partially.

The rest of this paper is organized as follows. Section 2 provides details about existing power scheduling techniques and their drawbacks regarding aforementioned challenges. Section 3 details the 'MOCSF' modules. An illustrative example is provided after each step to ease the understanding of each module. In Section 4, the experiments conducted to validate our approach and the main results obtained are presented. Section 5 concludes the paper.

## 2 Related work

Many approaches have been proposed in the literature to solve the power scheduling problem. Current approaches can be categorized into two main groups [7]: semi-automatic schedulers [10, 17, 5] and automatic schedulers [15, 13, 1] schedulers. In the semi-automatic schedulers, the consumers inject their desired preferences (e.g., desired temperature, appliances start time preferences, etc.) during the scheduling, contrary to the automatic schedulers where there is no human intervention. In our work, we will be focusing on six scheduling approaches, that vary in their scheduler, optimization problem type, appliances types and objectives.

### 2.1 Semi-automatic Schedulers

In [10], the authors developed a distributed power consumption scheduling algorithm aiming at reducing the electricity bills and balancing the total power demand when multiple consumers share a single energy source. To do so, the authors formulated a game-theory technique, where the players are the consumers and the strategies are their corresponding power consumption schedules (represented as vectors). The objective function of each consumer $n$ when choosing the strategy $x_n$ is defined as follows:

$$Min \sum_{h=1}^{H} C_h(\sum_{n \in N} \sum_{a \in A_n} x_{n,a}^h) \tag{1}$$

Where $C_h$ is the cost function, assumed to be strictly convex for each $h \in H$, andd $H = 24$. $x_{n,a}^h$ is the schedule of the appliance $a$, owned by the player $n$, at hour $h$. The pseudo-code of the distributed algorithm proposed is provided in cf. Algorithm 1.

For each player $n \in N$, the power consumption scheduling is generated randomly. The intuition behind this choice is that the authors considered that, at the beginning, a player $n$ has no prior information about others players. Then, a loop is executed until the algorithm converges. Within the loop, the objective function is resolved using an IPM algorithm [3], resulting a new schedule for each player. The same process is repeated until there is no new announced schedule for all the players. Simulations results showed that the proposed distributed algorithm can reduce the electricity bills and the peak of average ratio.

**Algorithm 1:** Executed by each consumer $n \in N$

```
Randomly initialize x_n and x_-n
repeat
    At random time instances Do
    Solve the objective function using IPM
    if x_n changes compared to current schedule then
        Update x_n according to the new schedule
        Broadcast a control message to annouce l_n to the other consumers
    if a control message is received then
        Update l_n accordingly
until no new schedule is announced
```

In [17], the authors developed a meta-heuristic scheduling algorithm, aiming at reducing the dissatisfaction and the energy cost of a set of homes in a district, and the variance of the grid. To do so, the authors divided the home appliances into two categories: power-shiftable and time-shiftable appliances. The objective function is formulated as follows:

$$\begin{aligned} Min \sum_{t=1}^{T}\sum_{j=1}^{S}[I_{ij}(t) * U_{ij}(t) + \alpha * (\gamma(t) * \sum_{j=1}^{S} P_{ij}(t)) \\ +\beta * (\sum_{t=1}^{T}\sum_{j=1}^{S} * P_{ij}(t) - 1/|T| * \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{S} P_{ij}(t))^2] \end{aligned} \tag{2}$$

Where $T$ is the set of time interval, $N$ is the set of households, $S$ is the set of electric appliances, $I_{ij}(t)$ is a binary variable denoting the working status of the appliance $j$ in the household $i$ at time $t$, $U_{ij}(t)$ is the dissatisfaction caused by operating the appliance $i$ in the household $i$ at time $t$, $\gamma(t)$ is the electricity sale price at time $t$, and $P_{ij}(t)$ is the working power of the appliance $j$ in the household $i$ at time $t$.

The dissatisfaction function $U_{ij}(t)$ represents the difference between the desired temperature and the actual indoor temperature for the space heater at time $t$, and the difference between the desired hot water temperature and the actual hot water temperature for the water heater at time $t$.

The authors used the Cooperative Particle Swarm Optimization (CPSO) (cf. Figure 1) to find the optimal scheduling of the appliances. Experimental results showed the positive impact of the households coordination in decreasing the peak loads and reducing the power costs.

In [5], the authors developed a power consumption scheduling aiming at reducing the electricity bills of the consumers with a minimum impact on their consumption preferences. The authors considered that the scheduler needs to determine the consumption vector $X_i = [x_{i,1}, x_{i,1}, ..., x_{i,H}]$ for each unit $i$ in the determined zone horizon $H$, where $H$ consists of $M$ segments comprised of $m$ time intervals, i.e., $H = M * m$. Then, a shrinking horizon optimization problem [4] has been defined as follows:

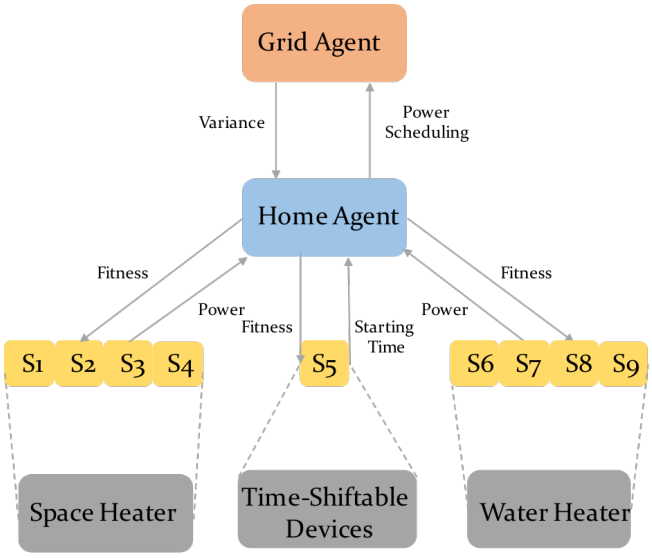


**Fig. 1.** CPSO Configuration and Operation

$$\mathcal{S}^{(j)}(H) = \sum_{h=jm-m+1}^{jm} \mathcal{S}(t_h) + \sum_{h=jm+1}^{jm} \hat{\mathcal{S}}(t_h) \tag{3}$$

Where $\mathcal{S}^{(j)}(H)$ is the total electricity cost in the $j^{th}$ optimization step, $\sum_{h=jm-m+1}^{jm} \mathcal{S}(t_h)$ is the energy cost for m intervals in the $j$th time segment based on actual electricity prices, and $\sum_{h=jm+1}^{jm} \hat{\mathcal{S}}(t_h)$ is the estimated energy cost based on the forecasted electricity prices for subsequent time intervals. Note that the user preferences are considered by including the time intervals where energy scheduling is performed for unit $i$. Without giving details about the obtained results, the authors assume that the proposed model can minimize the electricity consumption costs while including the consumers' preferences.

### 2.2 Automatic Schedulers

In [13], the authors formulated an optimization model for households power scheduling, aiming at reducing the electricity costs and the peak load of the grid. To do so, the authors integrated the incentive and inconvenience concepts. The incentive is offered to the users during peak times to encourage them to reduce their consumption, while the inconvenience seeks to reduce the difference between the baseline and the optimal appliances schedule. The objective function is defined as follows:

$$Min \sum_{t=1}^{T} \sum_{j=1}^{I} [P_i(\gamma_t * U_{i,t}^{opt} - \beta_t * \delta(U_{i,t}^{bl} - U_{i,t}^{opt})) * \Delta.t + (U_{i,t}^{bl} - U_{i,t}^{opt})^2] \quad (4)$$

Where $P_i$ is the rated power of the appliance $i$, $U_{i,t}^{opt}$ is the new on/off status of the appliance $i$ at time $t$, $U_{i,t}^{bl}$ is the baseline on/off status of the appliance $i$ at time $t$, $I = 10$, $T = 144$, $\delta(U_{i,t}^{bl} - U_{i,t}^{opt}) = 1$ if $(U_{i,t}^{bl} - U_{i,t}^{opt}) > 0$ and $\delta(U_{i,t}^{bl} - U_{i,t}^{opt}) = 0$ if $(U_{i,t}^{bl} - U_{i,t}^{opt}) < 0$. The formulated model is solved using the MINLP algorithm, which utilizes the Mixed Integer Programming (MIP) [16] and the Non-Linear Programming (NLP) [3]. Simulations results showed that using this model, the consumers realized 25% of electricity cost reduction. Noting that this percentage is affected by several factors, such as the number of shiftable appliances and the prices of the on-peak and off-peak times.

In [15], the authors developed a power storage scheduling algorithm aiming at managing the storage in the grid in a way of saving energy and reducing the reliance on the non-renewable energy sources. To do so, the authors formulated a game-theorist technique, where the players are the consumers and the strategies are their storage schedule vectors. The objective function of each player $i$ when choosing the strategy $s_i$ is defined as follows:

$$P_i(s_i, s_{-i}) \sum_{h=1}^{H} (s_i^h + l_i^h) \quad (5)$$

Where $s_i$ is the storage schedule vector of all the players expect $i$, $P_i(s_i, s_{-i})$ is the power price determined using a continuous and supply curve, and $l_i^h$ is the amount of power required by the player $i$ at time $h$. The Nash equilibrium of the game correspond to the storage schedule $s_i$ that minimizes the global generator costs given by $\sum_{H}^{h=1} \int_0^{q_h} b_h(x)dx$, where $b_h()$ is the supply curve and $q_h$ is the the total amount of power traded by all the players at time $h$. Simulation results showed that it is possible to realize an electricity bill saving of 13% per consumer with a storage capacity of $4KW$.

Similar to [13], the authors in [1] proposed an energy storage and loads scheduling algorithms aiming at reducing the electricity costs and the peak load hours. In this study, the electricity load analysis is done by grouping the day periods into three time zones each representing a cluster. Each cluster represents the loads expected to be launched during a given period. The cost required to satisfy the power needs of a given cluster j, consisting of K appliances is given by:

$$\mathcal{C}_j = \sum_{m=1}^{K} \sum_{h \in T_j} \{(E_{h,m} + B_{h,m}^c - B_{h,m}^d) * r_h\} \quad (6)$$

Where $E_{h,m}$ is the power purchased from the utility grid by a consumer m to meet its electrical appliances' power needs at period h, $B_{h,m}^c$ and $B_{h,m}^d$ are the

charging and discharging power profiles of the consumer m for the same period h, and $r_h$ is the market power price at a period h. An optimal load and storage scheduling should satisfy the consumers' requirements at the lowest cost in each period without harming the grid stability. To do so, the objective function has been defined as follows:

$$Min \sum_{j=1}^{3} (\mathcal{C}_j) \tag{7}$$

Here, linear programming was applied in resolving the optimization problem. Simulation results showed a 20% of peak load reduction and a 17% of costs savings.

### 2.3 Discussion

Table 1) shows a comparison between the existing DSM approaches highlighting their strengths and drawbacks with respect to the aforementioned challenges. On can observe the following:

**Table 1.** Comparing existing DSM approaches

| | **Scheduling Coverage** | **Satisfaction** | **Multiple Energy Sources** | **Restricted Goal** |
|---|---|---|---|---|
| Rad et al. [10] | Partial | - | - | Partial |
| Koukam et al. [17] | Partial | Partial | - | Partial |
| Ditiro et al. [13] | Partial | + | - | Partial |
| Peruknishnen et al. [15] | Partial | Partial | - | Partial |
| Christopher et al. [1] | Partial | - | - | Partial |
| Amin et al. [5] | Partial | + | - | Partial |
| **Our Approach** | + | + | + | + |

- **Scheduling coverage**: All the existing DSM approaches [10, 17, 13, 1, 5, 15] focused only on the power consumption scheduling, with the exception of [1, 15] that addressed the storage scheduling as well. However, none of them covers the power production scheduling.
- **Consumer satisfaction**: Few approaches [1, 15, 17] took into account the consumers' satisfaction. In [17], the consumers' comfort is ensured by reducing the gap between the desired and the actual hot water, and between the desired and the actual indoor temperature. However, in [1], [15], the satisfaction is measured by the delay time between the desired start time and the real operation of its household appliances. Contrariwise to [10], [13, 5], where this aspect was completely absent.
- **Multiple energy sources**: To the best of our knowledge, all the DSM approaches [10], [17, 13, 1, 5, 15] target the interaction of the consumers while assuming having only one utility grid and consequently one energy source.

– **Restricted goal**: Another limitation of all the existing approaches is that they do not cope with the three objectives (operational, ecological, and economical) of a successful DSM. In almost all the approaches [10], [17, 13, 1, 5, 15], the goal was mainly to reduce the electricity costs (economical aspect). In [10, 17], the peak load reduction (operational aspect) is addressed aiming at reducing the peak hours in the power grid. However, none of the approaches considers the gas emission reduction (ecological aspect).

All these limitations lead us to develop a new DSM cooperative model, allowing the scheduling of the power production, consumption and storage while considering the three-objective aspect of the DSM and the components' preferences.

## 3 Multi-objective Cooperative Scheduling

In this section, we detail our 'Multi-Objective Cooperative Scheduling Framework' or $MOCSF$ aiming at reducing electricity bills, peak loads and environmental bad effects, while enhancing the comfort of the $SG$ components. In order to conceive a cooperative environment, $MOCSF$ takes as input a set of couples, or what we call : seller-to-buyer associations, each consisting of a seller and buyer having mutual benefits in working together, with their desired schedules reflecting their operational preferences in terms of: start time, end time and power quantity (to sell or to buy). The main reason behind this choice relies on the fact that we do not want to schedule the sellers and the buyers randomly but we rather want to maintain the power exchange between the sellers and the buyers having the biggest interest in working together (the interest can be expressed via an objective function that takes into account the ecological, economical and operational parameters). Several approaches have been proposed in the literature to provide appropriate associations [12].

For the rest of our paper, we will be using the following example: Let us consider an $SG$ consisting of 9 components having the power generation *(g)*, demand *(d)* and storage *(s)* as shown in Table A.1. After classifying the $SG$ components, they will be put into three main categories: the sellers willing to sell their power surplus ($nR_1 \rightarrow nR_1^+$ , $nR_4 \rightarrow nR_2^+$, $nR_6 \rightarrow nR_3^+$, $nR_8 \rightarrow nR_4^+$), the buyers willing to buy their power needs ($nR_2 \rightarrow nR_1^-$, $nR_3 \rightarrow nR_2^-$, $nR_5 \rightarrow nR_3^-$, $nR_9 \rightarrow nR_4^-$), and the self-satisfied components ($nR_5 \rightarrow nR_1^0$) (c.f. Table A.2.).

After applying a clustering algorithm aiming at gathering the $SG$ components having mutual interests in working together based on the minimization of an objective function that takes into account the ecological, economical and operational costs, the result is a set of couples or seller-to-buyer associations as follows:

$$\begin{cases} Association\ 1 : (nR_1^+, nR_3^-) \text{ - } nR_1^+ \text{ should sell } nR_3^- \text{ a quantity of 14 } kW \\ Association\ 2 : (nR_1^+, nR_1^-) \text{ - } nR_1^+ \text{ should sell } nR_1^- \text{ a quantity of 3 } kW \\ Association\ 3 : (nR_4^+, nR_1^-) \text{ - } nR_4^+ \text{ should sell } nR_1^- \text{ a quantity of 1 } kW \\ Association\ 4 : (nR_4^+, nR_2^-) \text{ - } nR_4^+ \text{ should sell } nR_2^- \text{ a quantity of 2 } kW \end{cases}$$

| Component | Generation (g) | Demand (d) | Storage (s) |
|---|---|---|---|
| $nR_1$ | 17 | 0 | 0 |
| $nR_2$ | 2 | 35 | 1 |
| $nR_3$ | 4 | 10 | 1 |
| $nR_4$ | 20 | 0 | 5 |
| $nR_5$ | 5 | 5 | 0 |
| $nR_6$ | 0 | 0 | 3 |
| $nR_7$ | 6 | 20 | 0 |
| $nR_8$ | 5 | 3 | 1 |
| $nR_9$ | 10 | 20 | 5 |

Table A.1. Example of 9 components

| Component | (g) | (d) | (s) | Gap(G) |
|---|---|---|---|---|
| $nR_1$ ($nR_1^+$) | 17 | 0 | 0 | +17 |
| $nR_2$ ($nR_1^-$) | 2 | 35 | 1 | -32 |
| $nR_3$ ($nR_2^-$) | 4 | 10 | 1 | -5 |
| $nR_4$ ($nR_2^+$) | 20 | 0 | 5 | +25 |
| $nR_5$ ($nR_1^0$) | 5 | 5 | 0 | 0 |
| $nR_6$ ($nR_3^+$) | 0 | 0 | 3 | +3 |
| $nR_7$ ($nR_3^-$) | 6 | 20 | 0 | -14 |
| $nR_8$ ($nR_4^+$) | 5 | 3 | 1 | +3 |
| $nR_9$ ($nR_4^-$) | 10 | 20 | 5 | -5 |

Table A.2. Classification module result

As shown in Figure 2, *MOCSF* consists of two main modules: **Preference-**

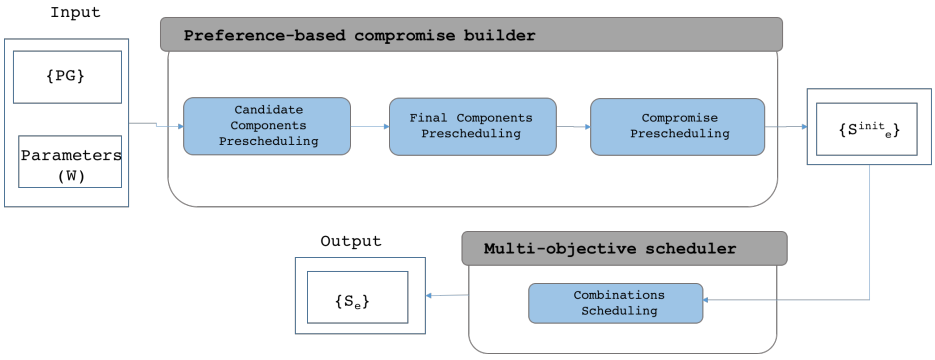


**Fig. 2.** Multi-objective Cooperative Scheduling Framework

**based compromise builder** and **Multi-objective Scheduler** detailed in what follows.

### 3.1 Preference-based compromise builder

As mentioned before, the input of this module is a set of seller-to-buyer associations, each composed of a seller and a buyer. Note that, each seller or buyer might belong to one or several associations. While sellers and buyers of the same association have to exchange power, each one of them has its own preferences to be respected so to establish a successful cooperative *SG*. Hence, the first step towards each association scheduling is to find the best balance or what we call the compromise, between the preferences of the related seller and buyer. Let us consider the first association ($nR_1^+$, $nR_3^-$) of the illustrative example. $nR_1^+$ and $nR_3^-$ should be scheduled together, however $nR_1^+$ may have several preferences that are different from $nR_3^-$'s: for instance, this latter prefers to buy power at 7:00 am, while $nR_1^+$ prefers to sell its surplus at 8:00 am. Hence, our goal is

to find the trade-off between the sellers and the buyers preferences. The problem becomes more and more complicated when each seller and buyer exchanges power with several components (since each can belong to several associations). For instance, $nR_1^+$ belongs to another association as well, ($nR_1^+$, $nR_1^-$), where $nR_1^-$ prefers also to buy power at 7:00 am. Hence, our module should propose an optimal distribution of the sellers' available power at each time t, in that it can meet its preferences and the buyers preferences, as well. Note that, for privacy reasons, a component has no prior information with whom he will exchange power, he can only precise the quantity he needs to sell or buy at each time t.

Before detailing the process, we present first some definitions used in our study. Each component $nR^2$, coud be a seller $nR^+$ or a buyer $nR^-$ having respectively power surplus and power need.

**Definition 1 (Schedule *[S]*)** *A schedule S consists of the power exchanged vector* $s_{\mathcal{R}} = [s_{\mathcal{R}}^1, s_{\mathcal{R}}^2, ..., s_{\mathcal{R}}^T]$*, where* $s_{\mathcal{R}}^t$ *denotes the corresponding power quantity (in KW) that an entity R is willing to exchange, at a time t over a period T* ⧫

**Definition 2** ***[PurchaseGraph** [PG]] A PurchaseGraph PG is an oriented graph (V, E, S, EV) consisting of representing power scheduling of vertices* $v_i$ *and associations* $e_i^j$ *where each vertice* $v_i \in V = \{nR^+\} \cup \{nR^-\}$ *represents a component, each edge* $e_i^j$ *connects a seller* $v_i \in \{nR^+\}$ *to a buyer* $v_j \in \{nR^-\}$ *with the total power quantity in EV exchanged between them, and each vertice* $v_i$ *or edge* $e_i^j$ *is associated to one desired schedule, denoted* $s^{init} \in S$*, and one operational schedule* $s^{op} \in S$*. The desired schedule designates the component operational preferences expressing its willing power quantity to exchange at each time t within a period T. The operational schedule designates the proposed schedule (provided by our algorithm). Note that,* $\forall\ nR \in \{PG_k\} \Rightarrow nR \notin \{PG_{\neq k}\}$*. To simplify in what follows,*

- $e.nR^+$ *designates the edge seller,*
- $e.nR^-$ *designates the edge buyer,*
- $e.EV$ *designates the edge total power quantity,*
- $s_{nR}^{init}$ *designates the component desired schedule,*
- $s_{nR}^{op}$ *designates the component operational schedule,*
- $s_e^{init}$ *designates the edge desired schedule, and*
- $s_e^{op}$ *designates the edge operational schedule.*

⧫

**Definition 3** ***[Satisfaction** $[S(e, W)]$]. The satisfaction of an edge e is defined according to the operational, economical, and ecological satisfactions of its vertices (its connected seller and buyer). It considers the sellers and buyers' comfort (operational), the power peak load (operational), the electricity bills (economical) and the environmental impacts (ecological). Although it can be defined using different aggregation functions (e.g., maximum, average, etc.), we adopted the*

[2] Self-satisfied components are not included here

*weighted sum function to combine the different objective aspects costs, allowing the user to tune the weight of each criterion. Formally:*

$$S(e, W) = W.w_{op} \times S_{op}(e) + W.w_{eco} \times S_{eco}(e) + W.w_{ecolo} \times S_{ecolo}(e) \quad (8)$$

*where:*

- $S_{op}(e)$ *represents the operational satisfaction of* $e$*,*
- $S_{eco}(e)$ *represents the economic satisfaction of* $e$*,*
- $S_{ecolo}(e)$ *represents the ecological satisfaction of* $e$*, and*
- $W$ *is a set of three weights, denoted as* $:\prec w_{op}, w_{eco}, w_{ecolo} \succ, w_{op} + w_{eco} + w_{ecolo} = 1$ *and* $(w_{op}, w_{eco}, w_{ecolo}) \geq 0$

♦

Thus, the satisfaction of a $PG$ consisting of $M$ edges is defined as follows:

$$S(PG, W) = \sum_{i=0}^{M} S(e_i, W) \quad (9)$$

Similarly, the satisfaction of an $SG$ consisting of $N$ number of $PG$ is defined as follows:

$$S(SG, W) = \sum_{i=0}^{N} S(PG_i, W) \quad (10)$$

Note that, in our study, we are aiming to minimize the operational, ecological and economical dissatisfactions ($Dis$) as follows:

$$\begin{aligned} Dis(e, W) &= \frac{1}{1 + S(e, W)} \in [0, 1] \\ Dis(PG, W) &= \frac{1}{1 + S(PG, W)} \in [0, 1] \\ Dis(SG, W) &= \frac{1}{1 + S(SG, W)} \in [0, 1] \end{aligned} \quad (11)$$

where, the lower is the dissatisfaction (tends to 0), the higher is the satisfaction.

**Definition 4 (Operational Satisfaction** *[$S_{op}$]***)** *The operational satisfaction of an edge* $e$*, denoted* $S_{op}(e, W)$*, is defined as:*

$$S_{op}(e, W) = W.w_{\alpha} \times Comfort(e) + W.w_{\beta} \times Variance(e) + W.w_{\gamma} \times Variance(PG) \quad (12)$$

*where* $W$ *is a set of three weights, denoted as* $:\prec w_{\alpha}, w_{\beta}, w_{\gamma} \succ, w_{\alpha} + w_{\beta} + w_{\gamma} = 1$ *and* $(w_{\alpha}, w_{\beta}, w_{\gamma}) \geq 0$*, and* $PG$ *is the PurchaseGraph to which* $e$ *belongs.* ♦

Thus, the operational satisfaction of a $PG$ consisting of $M$ edges is defined as follows:

$$S_{op}(PG, W) = \sum_{i=0}^{M} S_{op}(e_i, W) \quad (13)$$

Similarly, the operational satisfaction of an $MG$ consisting of $N$ number of $PG$ is defined as follows:

$$S_{op}(SG, W) = \sum_{i=0}^{N} S_{op}(PG_i, W) \tag{14}$$

Note that, the operational dissatisfactions ($Dis_{op}$) is defined as follows:

$$\begin{aligned} Dis_{op}(e, W) &= \frac{1}{1 + S_{op}(e, W)} \in [0, 1] \\ Dis_{op}(PG, W) &= \frac{1}{1 + S_{op}(PG, W)} \in [0, 1] \\ Dis_{op}(SG, W) &= \frac{1}{1 + S_{op}(SG, W)} \in [0, 1] \end{aligned} \tag{15}$$

where, the lower is the operational dissatisfaction (tends to 0), the higher is the operational satisfaction.

**Definition 5 (Comfort** *[$Comfort(e)$]***)** *The comfort of an edge e, is the waiting time penalization of its vertices, defined as:*

$$\begin{aligned} Comfort(e) = \sum_{t=1}^{T} Avg(&e.nR^{+}.Op.Penalty \times |s_e^{op}[t] - s_{e.nR^{+}}^{init}[t]| \\ &+ e.nR^{-}.Op.Penalty \times |s_e^{op}[t] - s_{e.nR^{-}}^{init}[t]|) \end{aligned} \tag{16}$$

*where $Penalty$ is the waiting time penalty of the seller $nR^{+}$ and the buyer $nR^{-}$, $|s_e^{op}[t] - s_{e.nR^{+}}^{init}[t]|$ is the difference between the initial desired schedule and the real operation of the seller $nR^{+}$, and $|s_e^{op}[t] - s_{e.nR^{-}}^{init}[t]|$ is the difference between the initial desired schedule and the real operation of the buyer $nR^{-}$.* ♦

Note that, the penalty is a positive weighting factor, which represents the waiting time flexibility of the component. If the penalty is zero, this means that the component does not penalize the delay between its desired and operational schedule. The highest is the penalty, the most the component is delay time constraining.

**Definition 6 (Variance e** *[$Variance(e)$]***)** *The variance of an edge e, denoted $Variance(e)$ is the peak load ratio of its vertices, defined as:*

$$Variance(e) = \sum_{t=1}^{T} \left( s_e^{op}[t] - \frac{\sum_{t=1}^{T} s_e^{op}[t]}{|T|} \right)^2 \tag{17}$$

♦

Note that, the variance is a positive value reflecting the power load dispersion all along T. The highest is the variance, the higher are the peak loads probabilities.

**Definition 7 (Variance PG** *[VariancePG)]*) *The variance of a PG, denoted* $Variance(PG)$ *is the peak load ratio of its edges, defined as:*

$$Variance(PG) = \sum_{t=0}^{T} \left( \sum_{i=0}^{M} S_e^{op}[t] - \frac{\sum_{t=0}^{T} \sum_{i=0}^{M} S_e^{op}[t]}{|T|} \right)^2 \tag{18}$$

*where* $M$ *is the number of* $e$ *in* $PG$ ♦

**Definition 8 (Variance SG** *[*$Variance(SG)$*]*) *The variance of an MG, denoted* $Variance(SG)$ *is the peak load ratio of the PGs forming the SG, defined as:*

$$Variance(SG) = \sum_{t=1}^{T} \sum_{j=1}^{N} \left( \sum_{i=1}^{M} PG_j.s_{e_i}^{op}[t] - \frac{\sum_{t=1}^{T} \sum_{i=1}^{M} PG_j.s_{e_i}^{op}[t]}{|T|} \right)^2 \tag{19}$$

*where* $N$ *is the number of* $PG$ *in* $SG$ *and* $M$*is the number of* $e$ *in* $PG$ ♦

**Definition 9 (Economical Satisfaction** *[*$S_{eco}$*]*) *The economical satisfaction of an edge* $e$*, denoted* $S_{eco}(e)$*, is defined as:*

$$\begin{aligned} S_{eco}(e) = \sum_{t=1}^{T} Avg(s_e^{op}[t] \times MG.Op.PwrCost[t] \\ + e.nR^{+}.Op.LaunchCount \times (e.nR^{+}.Eco.SUCost + e.nR^{+}.Eco.SDCost) \\ + e.nR^{-}.Op.LaunchCount \times (e.nR^{-}.Eco.SUCost + e.nR^{+}.Eco.SDCost)) \end{aligned} \tag{20}$$

*where* $PwrCost[t]$ *is the electricity price at a time* $t$ *and* $LaunchCount$ *is the number of launches of the sellers and buyers belonging to* $e$*, during* $T$*. Note that, all these parameters are represented in our OntoMG.* ♦

Similarly,

$$\begin{aligned} Dis_{eco}(e) &= \frac{1}{1 + S_{eco}(e)} \in [0,1] \\ Dis_{eco}(PG) &= \frac{1}{1 + S_{eco}(PG)} \in [0,1] \\ Dis_{eco}(SG) &= \frac{1}{1 + S_{eco}(SG)} \in [0,1] \end{aligned} \tag{21}$$

where, the lower is the economical dissatisfaction (tends to 0), the higher is the economical satisfaction.

**Definition 10 (Ecological Satisfaction** *[*$S_{ecolo}$*]*) *The ecological satisfaction of an edge* $e$*, denoted* $S_{ecolo}(e)$*, is defined as:*

$$S_{ecolo}(e) = \sum_{t=1}^{T} s_e^{op}[t] \times e.nR^{+}.Ecolo.GasEss \times SG.Op.GasEssCost \tag{22}$$

*The ecological satisfaction depends on the toxic gas emissions GasEss emitted during the power production, and the cost GasEssCost per unit of gas emission. Note that, all these parameters are modeled in our OntoMG ontology.* ♦

Thus, the ecological satisfaction of a $PG$ consisting of $M$ edges is defined as follows:

$$S_{ecolo}(PG) = \sum_{i=0}^{M} S_{ecolo}(PG) \tag{23}$$

Similarly, the ecological satisfaction of an $MG$ consisting of $N$ number of $PG$ is defined as follows:

$$S_{ecolo}(SG) = \sum_{i=0}^{N} S_{ecolo} S_{ecolo}(SG) \tag{24}$$

Note that, the ecological dissatisfactions ($Dis_{ecolo}$) is defined as follows:

$$\begin{aligned} Dis_{ecolo}(e) &= \frac{1}{1 + S_{ecolo}(e)} \in [0,1] \\ Dis_{ecolo}(PG) &= \frac{1}{1 + S_{ecolo}(PG)} \in [0,1] \\ Dis_{ecolo}(SG) &= \frac{1}{1 + S_{ecolo}(SG)} \in [0,1] \end{aligned} \tag{25}$$

where, the lower is the ecological dissatisfaction (tends to 0), the higher is the ecological satisfaction.

An overview of the Preferences-based Compromise Builder module is shown in Figure 3 consisting of three main components: 1) Candidate components' prescheduling, 2) Final components' prescheduling, and 3) Compromise prescheduling. They are detailed below.

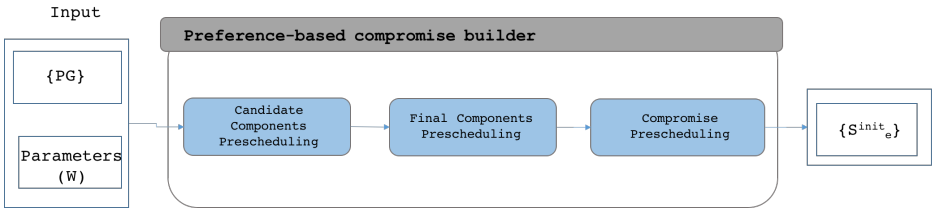


**Fig. 3.** Preferences-based Compromise Builder Framework

**3.1.1 Candidate components' prescheduling** The aim of this module is to dissociate the desired schedule of each seller/buyer, so as to distribute the power

quantity at each time t (its capacity of selling/buying) between the components with which, it must exchange, without exceeding nor being inferior to its desired capacity at time t. The pseudo-code of the candidate components' prescheduling is provided in Algorithm 2. Briefly, for each seller/buyer, we retrieve the list of edges to which the seller/buyer belongs. Then, we generate the list of all the possible permutations of the retrieved edges (Lines 7-16). For each possible permutation list of edges at a time t, we verify if the seller/buyer has enough power to sell/to buy to the buyer/from the seller of the same edge (Line 17). If there is enough power (Lines 18-20), we fill the schedule with the quantity to buy/to sell and recall the process by the next seller/buyer. If not, we fill the schedule with the quantity to buy/sell, reduce the quantity to sell/buy, and verify the quantity to sell/buy to the next buyer/from the next seller of the next edge (Lines 21-25).

**Algorithm 2:** Candidate Components' Prescheduling

```
   Input: PG[]                                                          // Set of PG forming the SG
   Output: PG[]                                  // Set of SG components updated with their candidate preschedules
 1 S.CS = new int [][]                           // Initialize a candidate Solution S having a candidate Schedule CS
 2 S.e = new Edge []                                 // Initialize a candidate Solution S having a list of edges e
 3 int RPL
 4 for int i = 0; i < | SG.PG[] |; i++                                  // For each PurchaseGraph in the Smart Grid
 5  do
 6     E[] = GLE(SG.PG[i].e.nR)          // Retrieve the list of edges to which the seller/buyer of the edge belongs
 7     PE[][] = Permutate(E[])                          // Retrieve the possible permutation of the list of edges
 8     for each e[] ∈ PE[][]                                    // For each list of permutated lists of edges
 9      do
10        S.CS = new int [| e[] |][T]                  // Initialize a candidate schedule CS for a solution S
11        S.Ce = new Couple [| e[] |]                      // Initialize a set of edges Ce for a solution S
12        RP[] = ∑_{t=1}^{T} S^{init}_{SG.PG[i].e.nR}[t]   // Initialize the remaining production to sell/buy to the desired
            selling/buying vector
13        for int j = 0, j < | e[] | , j++                          // For each edge in the pemutated list of edges
14         do
15           RPL = e[j].EV     // Initialize the remaining production to buy/sell (of the linked component) with
               the valued exchanged of the couple
16           for int k = 0; k < T; k++                                                    // For each time k
17            do
18              if RP[k] >= RPL                                  // If there is sufficient power to sell
19              /buy then
20                 S.CS[j][k] = RPL                  // Fill the schedule with the quantity to buy/sell
21                 RPL = 0                                  // No more power need to buy/sell
22              else
23                 S.CS[j][k] = RP[k]                // Fill the schedule with the quantity to buy/sell
24                 RPL− = RP[k]                             // Reduce the quantity to buy/sell
25              RP[k]− = S.CS[j][k]                         // Reduce the quantity to sell/buy
26        SG.PG[i].e.nR.S.Add(S)                  // Add S as a candidate solution of the seller/buyer
```

**3.1.2 Final components' prescheduling** The aim of this module is to select the candidate components' schedules that guarantee that each edge is provided with its exchanged value ($EV$) at each time t (e.g., at the end of the day, where t = 24h). In other words, for each edge, the sum of the power quantity exchanged between its sellers and the buyers at T, should be equal to their exchanged value ($EV$) in the $PG$. So, the sellers sell all their power surplus and the buyers satisfy all their needs. The pseudo-code of the final components' schedules is provided in Algorithm 3. Briefly, for each candidate schedule of each seller (Lines 2-9)

and for each time t of the day, we calculate the sum of the energy exchanged of the edges to which the seller/buyer belongs. The schedule is accepted if the sum is equal to the exchanged value of the edge (Lines 12-13). If the equality is verified for all the edges, we add the candidate schedule to the final components' schedules (Lines 13-16).

**Algorithm 3:** Final Components' Prescheduling

```
   Input: PG[]                                                    // Set of PG forming the SG
   Output: PG[]                                  // Set of SG components updated with their final preschedules
 1 for int i = 0; i < | SG.PG[] |; i++                            // For each PurchaseGraph in the Smart Grid
 2  do
 3     for each s ∈ SG.PG[i].e.nR.S                               // For each possible solution of the seller/buyer
 4      do
 5         bool isAcceptedSolution = true
 6         for int j = 0; j < | s.CS[0][] | ; j++                 // For each candidate schedule of the seller/buyer
 7          do
 8             int sev = 0                                       // Initialize the sum of the exchange value with zero
 9             for int k = 0; k < T ; k++                                          // For each time k
10              do
11                 sev+ = s.CS[j][k]          // Calculate the sum of the energy exchanged during T of the edge
12             isAcceptedSolution = sev.Equals(s.Cs[j].EV)  // The solution is accepted if the sum is equal
                 to the value exchanged of the edge
13         if isAcceptedSolution                                  // If the equality is verified for all the edges
14          then
15             | SG.PG[i].e.nR.S.Add(S)                           // Add S as an accepted solution of the seller/buyer
```

**3.1.3 Compromise prescheduling** The aim of this module is to generate every seller-to-buyer association (edge) desired schedule. It consists of selecting the best combination between the final preschedules of the sellers and buyers. This can be done by selecting the combination that ensures the minimum gap between the desired schedules of the sellers and buyers and the proposed compromise desired schedule. The pseudo-code of the final components' schedules is provided in Algorithm 4. First, we generate the combinations between the final preschedules of the sellers and the buyers (Lines 1-3). Then, for each seller/buyer of each combination, we calculate the power quantity for each edge in each candidate schedule for all combinations at a time t and fill it into a new vector (*FinalQuantity*) (Lines 4-17). After that, a similarity computation of the resulting vector and the initial desired schedule (vector) of each seller/buyer is done using the cosine similarity measure (Line 18). In fact, we adopted the commonly adopted cosine measure to calculate the distance between the proposed and the desired schedule vectors (instead of many others such the Euclidean Distance, the Pearson Correlation Coefficient, etc.) since it provides better results when there are many values in common between the two schedules to compare. Finally, the combination vector having the biggest similarity or what we call it here 'minimum delay' will be retrieved (Lines 19-27).

**3.1.4 Preference-based compromise builder illustration** In our previous illustration, all the buyers and sellers are connected, forming one purchase graph (cf. Figure 4).

**Algorithm 4:** Compromise Prescheduling

```
Input: PG[]                                                          // Set of PG forming the SG
Output: S_e^init[]                                                   // Edges desired schedule
for int i = 0; i < | SG.PG[] |; i++                                  // For each PurchaseGraph in the Smart Grid
do
    Comb[] = Combination(SG.PG[i].e.nR+.S[], SG.PG[i].e.nR−.S[])     // Retrieve the possible
      combinations of the final sellers and buyers preschedules
for int i = 0; i < | Comb[] |; i++                                   // For each combination
do
    for int j = 0; j < | SG.PG[] |; j++                              // For each PurchaseGraph in the Smart Grid
    do
        for int k = 0; k < | Comb[i].Ce |; k++                       // For each set of edges of the selected combination
        do
            if Comb[i].Ce[k].nR == SG.PG[j].nR                       // Check if we are verifying the schedules of the same
              seller/buyer
            then
                for int l = 0; l < | Comb[i].CS |; k++               // For each set of candidate schedules of the
                  selected combination
                do
                    Comb[i].FinalQuantityPerHour[j][l]+ = Comb[i].CS[k][l]   // Sum the power
                      quantity for each edge in each candidate schedule of each combination
                for int x = 0; x < | Comb[i].CS |; x++               // For each set of candidate schedules of the selected
                  combination
                do
                    FinalQuantity[x] = Comb[i].FinalQuantityPerHour[j][x]    // Calculate the
                      combination's power quantity for each seller/buyer
                Comb[i].TotalDelay+ = 1 − Cosinus(S^init_SG.PG[j].nR, FinalQuantity)   // Calculate the
                  similarity between the desired schedule of the seller/buyer and the combination's schedule
minDelay = Comb[0].TotalDelay
for int i = 0; i < | Comb[] |; i++                                   // Retrieve the minimum delay
do
    if Comb[i].TotalDelay < minDelay then
        minDelay = Comb[i].TotalDelat
for int i = 0; i < | Comb[] |; i++                                   // Retrieve the combination having the minimum delay
do
    if Comb[i].TotalDelay == minDelay then
        S_e^init[] = Comb[i]
```

As an input, each seller and buyer proposes its desired schedule. Here, we will consider that T= 4.

$$\begin{cases} S^{init}_{nR_1^+} = [3, 14, 0, 0] \\ S^{init}_{nR_4^+} = [0, 0, 1, 2] \\ S^{init}_{nR_1^-} = [3, 0, 1, 0] \\ S^{init}_{nR_2^-} = [0, 0, 0, 2] \\ S^{init}_{nR_3^-} = [0, 14, 0, 0] \end{cases}$$

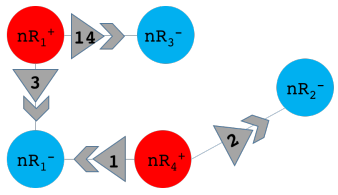


**Fig. 4.** Purchase Graph Illustration

The aim of applying the Preference-based comprimise builder is to find the desired schedule of the resulting linked couples ($nR_1^+$,$nR_3^-$), ($nR_1^+$,$nR_1^-$), ($nR_4^+$,$nR_1^-$) and ($nR_4^+$,$nR_2^-$): $S^{init}_{nR_1^+,nR_3^-}$, $S^{init}_{nR_1^+,nR_1^-}$, $S^{init}_{nR_4^+,nR_1^-}$ and $S^{init}_{nR_4^+,nR_2^-}$, respectively.

– Candidate components' prescheduling:
The output of the candidate components' prescheduling is as follows:

Candidate $nR_1^+$ prescheduling:
There are two possible solutions:

$$Solution1 \begin{cases} S^{init}_{nR_1^+,nR_3^-} = [3,11,0,0] \\ S^{init}_{nR_1^+,nR_1^-} = [0,3,0,0] \end{cases}$$

$$Solution2 \begin{cases} S^{init}_{nR_1^+,nR_3^-} = [3,0,0,0] \\ S^{init}_{nR_1^+,nR_1^-} = [0,14,0,0] \end{cases}$$

Those solutions were selected since the sum of the selling power at each time t is equal to the desired power quantity given as an input (3 kw at t=1 and 14 kw at t=2)

Candidate $nR_4^+$ prescheduling:
There are two possible solutions:

$$Solution1 \begin{cases} S^{init}_{nR_4^+,nR_1^-} = [0,0,1,0] \\ S^{init}_{nR_4^+,nR_2^-} = [0,0,0,2] \end{cases}$$

$$Solution2 \begin{cases} S^{init}_{nR_4^+,nR_1^-} = [0,0,0,1] \\ S^{init}_{nR_4^+,nR_2^-} = [0,0,1,1] \end{cases}$$

Those solutions were selected since the sum of the selling power at each time t is equal to the desired power quantity given as an input (1 kw at t=3 and 2 kw at t=4)

Candidate $nR_1^-$ prescheduling:
There are two possible solutions:

$$Solution1 \begin{cases} S^{init}_{nR_1^+,nR_1^-} = [3,0,0,0] \\ S^{init}_{nR_4^+,nR_1^-} = [0,0,1,0] \end{cases}$$

$$Solution2 \begin{cases} S^{init}_{nR_1^+,nR_1^-} = [1,0,0,0] \\ S^{init}_{nR_4^+,nR_1^-} = [2,0,1,0] \end{cases}$$

Those solutions were selected since the sum of the buying power at each time t is equal to the desired power quantity given as an input (3 kw at t=1 and 1 kw at t=3)

Candidate $nR_2^-$ prescheduling:
There is one possible solution:

$$Solution \begin{cases} S^{init}_{nR_4^+,nR_2^-} = [0,0,0,2] \end{cases}$$

Those solutions were selected since the sum of the buying power at each time t is equal to the desired power quantity given as an input (2 kw at t=4)

Candidate $nR_4^-$ prescheduling:
There is one possible solution:

$$Solution \begin{cases} S^{init}_{nR_1^+,nR_4^-} = [0,14,0,0] \end{cases}$$

Those solutions were selected since the sum of the buying power at each time t is equal to the desired power quantity given as an input (14 kw at t=2)

– Final components' prescheduling:
In our case, the output of the final components' prescheduling is the same output generated in the candidate components' prescheduling. Those solutions ensure that the sum of the power exchanged between a linkedcouple is equal to the exchanged value of this couple.

Final $nR_1^+$ prescheduling:
There are two possible solutions:

$$Solution1 \begin{cases} S^{init}_{nR_1^+,nR_3^-} = [3,11,0,0] : (nR_1^+, nR_3^-).EV = 14 = 11+3+0+0 \\ S^{init}_{nR_1^+,nR_1^-} = [0,3,0,0] : (nR_1^+, nR_1^-).EV = 3 = 0+3+0+0 \end{cases}$$

$$Solution2 \begin{cases} S^{init}_{nR_1^+,nR_1^-} = [3,0,0,0] : (nR_1^+, nR_1^-).EV = 3 = 3+0+0+0 \\ S^{init}_{nR_1^+,nR_3^-} = [0,14,0,0] : (nR_1^+, nR_3^-).EV = 14 = 0+14+0+0 \end{cases}$$

Final $nR_4^+$ prescheduling:
There are two possible solutions:

$$Solution1 \begin{cases} S^{init}_{nR_4^+,nR_1^-} = [0,0,1,0] : (nR_4^+, nR_1^-).EV = 1 = 0+0+1+0 \\ S^{init}_{nR_4^+,nR_2^-} = [0,0,0,2] : (nR_4^+, nR_2^-).EV = 2 = 0+0+0+2 \end{cases}$$

$$Solution2 \begin{cases} S^{init}_{nR_4^+,nR_1^-} = [0,0,0,1] : (nR_4^+, nR_1^-).EV = 1 = 0+0+0+1 \\ S^{init}_{nR_4^+,nR_2^-} = [0,0,1,1] : (nR_4^+, nR_2^-).EV = 2 = 0+0+1+1 \end{cases}$$

Final $nR_1^-$ prescheduling:
There are two possible solutions:

$$Solution1 \begin{cases} S^{init}_{nR_1^+,nR_1^-} = [3,0,0,0] : (nR_1^+, nR_1^-).EV = 3 = 3+0+0+0 \\ S^{init}_{nR_4^+,nR_1^-} = [0,0,1,0] : (nR_4^+, nR_1^-).EV = 1 = 0+0+1+0 \end{cases}$$

$$Solution2 \begin{cases} S^{init}_{nR_4^+,nR_1^-} = [1,0,0,0] : (nR_4^+, nR_1^-).EV = 1 = 1+0+0+0 \\ S^{init}_{nR_1^+,nR_1^-} = [2,0,1,0] : (nR_1^+, nR_1^-).EV = 3 = 2+0+1+0 \end{cases}$$

Final $nR_2^-$ prescheduling:
There is one possible solution:

$$Solution \begin{cases} S^{init}_{nR_4^+,nR_2^-} = [0,0,0,2] : (nR_4^+, nR_2^-).EV = 2 = 0+0+0+2 \end{cases}$$

Final $nR_4^-$ prescheduling:
There is one possible solution:

$$Solution \begin{cases} S^{init}_{nR_1^+,nR_4^-} = [0,14,0,0] : (nR_1^+, nR_4^-).EV = 14 = 0+14+0+0 \end{cases}$$

– Compromise prescheduling:
The output of the compromise prescheduling is as follows:

$$\begin{cases} S^{init}_{nR_1^+,nR_3^-} = [3,0,0,0] \\ S^{init}_{nR_1^+,nR_1^-} = [0,14,0,0] \\ S^{init}_{nR_4^+,nR_1^-} = [0,0,1,0] \\ S^{init}_{nR_4^+,nR_2^-} = [0,0,0,2] \end{cases}$$

This solution is the combination of the final seller and buyer preschedules that reduces the gap with the initial desired schedules of the sellers and buyers. Here, the Gap = 0 (the ideal solution).

### 3.2 Multi-objective Scheduler

Once done with the preferences-based combination generator that aims at extracting the desired schedules of the seller-to-buyer associations based on the sellers and buyers desired schedules given as input, it is time to schedule the resulting associations in a way to minimize the operational, economical and ecological aspects. As defined in Equation 3, our objective function takes into account: 1)- the operational aspect by considering the comfort of the sellers and buyers measured by the delay time between the desired schedule and the real operation, the peak load reduction of the $SG$ and the components calculated using the variance of the power at a time t 2)- the economical aspect is considered by measuring the electricity price at a time t, and 3)- the ecological aspect is treated by calculating the toxic gas emissions produced at a time t in the $SG$.

In our work, we adopted Particle Swarm Optimization (PSO), to search for the near-optimal scheduling for each seller-to-buyer association, because of its straightforward implementation and demonstrated ability of optimization. In essence, PSO is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality using an objective function. It is considered as a powerful tool to solve complex non-linear and non-convex optimization problems. Moreover, it has several other advantages, such as fewer parameters to adjust, and easier to escape from local optimal solutions.

Briefly, the problem is solved by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formula over the particle's position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best-known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

**Algorithm 5:** Particle Swarm Optimization Algorithm

**Input:** $x[], v[]$ // Set of particles' positions and velocities
**Output:** $x[], v[]$ // Set of updated particles' positions and velocities

**for** *each particle* $i = 0; i < S; i++$ **do**
  Initialize the particle's position with a uniformly distributed random vector: $x_i$ $U(b_{lo}, b_{up})$
  Initialize the particle's best known position to its initial position: $p_i = x_i$
  **if** $Dis(p_i) < Dis(g)$ **then**
    Update the swarm's best known position: $g = p_i$
  Initialize the particle's velocity: $v_i = U(\mid b_{up} - b_{lo} \mid, \mid b_{up} - b_{lo} \mid)$
  **while** *a termination criterion is not met* **do**
    **for** *each particle* $i = 0; i < S; i++$ **do**
      **for** *each dimension* $d = 0; i < n; i++$ **do**
        $Pickrandomnumbers : r_p, r_g$ $U(0, 1)$
        Update the particle's velocity:
        $v_{i,d} = \times v_{i,d} + p \times r_p(pi, d - xi, d) + g \times r_g(g_d - x_{i,d})$
      Update the particle's position: $x_i = x_i + v_i$
      **if** $Dis(x_i) < Dis(p_i)$ **then**
        Update the particle's best known position: $p_i = x_i$
        **if** $Dis(p_i) < Dis(g)$ **then**
          Update the swarm's best known position: $g = p_i$

The pseudo-code of the adapted PSO is provided in Algorithm 5. The goal is to find a solution a for which $Dis(a) < Dis(b)$ for all $b$, which would mean that $a$ is the global minimum. Let $S$ be the number of particles in the swarm, each having a position $x_i$ and a velocity $v_i$. Let $p_i$ be the best known position of particle $i$ and let $g$ be the best known position of the entire swarm. The values $b_{lo}$ and $b_{up}$ are respectively the lower and upper boundaries. The termination criterion can be number of iterations performed, or a solution with adequate objective function value is found. In our method, we set the parameters as in [17] to calibrate the PSO problem, used for mathematical models of Smart Homes.

## 4 Experiments

A set of experiments have been conducted to highlight the efficiency of our approach as explained below.

### 4.1 Experimental Context

A prototype has been implemented using Java to conduct the test on a PC with an Intel Core i7-3630 QM CPU, 2.40 GHz processor with 8 GB RAM. Since the $SG$ is relatively a recent concept in the power systems area, there is a lack of a current Benchmark to be based on. Hence, we carried out our experimental scenario inspired by the one provided in [12] but adapted to fit better the scope of our study. Here, we set up an $SG$ within an area of 10 km $\times$ 10 km with: 1) the main grid located at the onshore , and 2) the $SG$ components randomly located within this area. The prototype includes the following functionalities: 1) the Preference-based compromise builder and the 2) Multi-objective scheduler.

### 4.2 Experimental Metrics and Results

The main criteria used to evaluate the effectiveness of our approach are: i) the preference-based compromise builder effectiveness, ii) the time needed to gener-

ate the compromises, as well as, iii) the multi-objective scheduler impact on the electricity cost reduction, the peak loads and the gas emissions.

**4.2.1 Preferences-based compromise builder effectiveness** The efficiency of the generated desired compromise schedule is measured by its similarity with the desired schedules of the sellers and the buyers given as an input. The similarity measure used in our module is the 'Cosine Similarity Measure', which results a similarity between 0 and 1 (from an absence of similarity '0' to the biggest similarity '1').

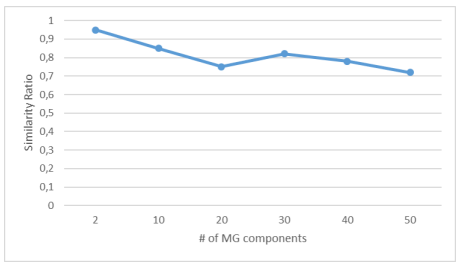


**Fig. 5.** Compromise Similarity w.r.t the number of *SG* Components

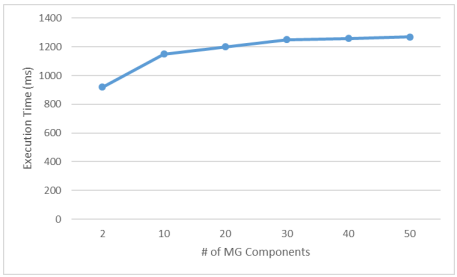


**Fig. 6.** Time performance w.r.t the number of *SG* Components

Figure 5 shows that the worst similarity ratio obtained is 0.72 and the best one is 0.95. This result reflects that our module ensures nice results providing an adequate compromise between the seller and the buyer preferences.

**4.2.2 Preferences-based compromise builder performance** In addition to testing the effectiveness of our module in reducing the gap between the proposed compromise desired schedule and the desired sellers and buyers, we also evaluated its time performance. This test consisted of measuring the necessary time to build the compromise from the sellers and buyers associations (cf. Figure 6), and showed a linear complexity of our algorithm.

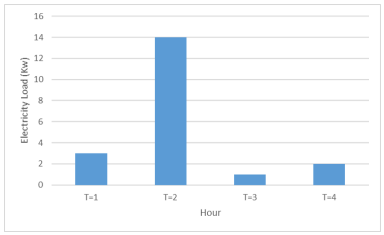


**Fig. 7.** Non-cooperative electricity load result

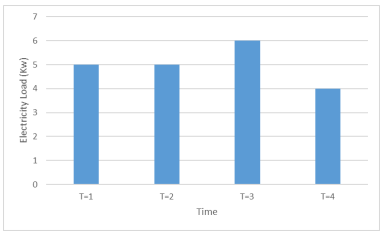


**Fig. 8.** MOCSF electricity load result

**4.2.3 Multi-objective scheduler impact on the *SG*** The cooperation and the multi-objective aspects of the $SG$ are the key features of our scheduling. Hence, we measured the following resulting costs: the total electricity prices, the total toxic gas emissions, the components comfort, and the peak loads.

In this test, two different scenarios were considered: 1) a non-cooperative scheduling, where each association is selfish in that it only considers its desired schedule, and 3) a cooperative scheduling based on our proposed multi-objective scheduling. To remain coherent, we will consider the scheduling of the seller-to-buyer associations of our same previous illustration. Note that, the output of the preference-based compromise builder will be used here to calculate the comfort of the components by calculating the gap between the resulting schedule and the compromise desired schedule.

Figure 7 shows the electricity load resulting from the non-cooperative case. At T=2, a peak load (Electricity load = 14 Kw) appeared having several bad effects on the economical, ecological and the operational costs. From the economical perspective, this peak load leads to a total electricity cost of 163 c. From the ecological perspective, and having at T=2, a conventional power generator (emitting 0.26 Kg Co2/Kwh), the non-cooperation scheduling caused a simultaneous gas emissions of 3.64 KgCO2. The only advantage of this scheduling is that it answers exactly the desired preferences of the components, which gives a similarity of 1 (the highest), between the proposed and the desired schedules.

Figure 8 shows the electricity load resulting from our multi-objective scheduler. It shows how the peak loads are shaved (Highest electricity load = 6 Kw). The result is a trade-off between the economical, ecological and operational aspects. From the economical perspective, the total electricity cost is reduced to 136 c. From the ecological perspective, the highest simultaneous gas emissions is reduced to 1.57 KgCO2. The only feature affected negatively is the similarity between the desired and the proposed schedule, reduced to 0.75. Despite this reduction, the value remains a very good result.

## 5 Conclusion

In this paper, we proposed $MOCSF$, a Multi-Objective Cooperative Scheduling Framework providing a multi-type scheduling for the power generation, storage and consumption, while taking into account the ecological, economical and operational costs in a power system. $MOCSF$ consists of two main modules: the Preference-based Compromise Builder, providing the best balance between the desired schedulers of the sellers and the buyers given as an input, and the Multi-objective Scheduler, providing seller-to-buyer associations scheduling aiming at ensuring the economical, ecological and operational satisfactions. Experiments results showed the potential of our modules in providing efficient preference-based compromises able to reduce the gap with the initial components preferences and in minimizing the three-dimensional costs. Currently, we are working on implementing a privacy-by-design [8] grid control allowing to protect the components privacy whilst preserving the advanced control and monitoring func-

tionalities of the power systems. Further, it is interesting to apply strategy-proof techniques, in order to avoid cheating in the desired schedules,

## References


1. Christopher O Adika and Lingfeng Wang. Smart charging and appliance scheduling approaches to demand side management. *International Journal of Electrical Power & Energy Systems*, 57:232–240, 2014.
2. Hunt Allcott. Social norms and energy conservation. *Journal of Public Economics*, 95(9):1082–1095, 2011.
3. Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.
4. Moritz Diehl, H Georg Bock, Johannes P Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.
5. Amin Fakhrazari, Hamid Vakilzadian, and F Fred Choobineh. Optimal energy scheduling for a smart entity. *IEEE Transactions on Smart Grid*, 5(6):2919–2928, 2014.
6. Clark W Gellings and JH Chamberlin. Demand-side management. *Energy Efficiency and Renewable Energy Handbook, Second Edition edited by D. Yogi Goswami, Frank Kreith (Chapter 15)*, pages 289–310, 1988.
7. Ijaz Hussain, Sajjad Mohsin, Abdul Basit, Zahoor Ali Khan, Umar Qasim, and Nadeem Javaid. A review on demand response: Pricing, optimization, and appliance scheduling. *Procedia Computer Science*, 52:843–850, 2015.
8. Marc Langheinrich. Privacy by designprinciples of privacy-aware ubiquitous systems. In *International conference on Ubiquitous Computing*, pages 273–291. Springer, 2001.
9. Julio J Lucia and Eduardo S Schwartz. Electricity prices and power derivatives: Evidence from the nordic power exchange. *Review of derivatives research*, 5(1):5–50, 2002.
10. Amir-Hamed Mohsenian-Rad and el. al. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *Smart Grid, IEEE Transactions on*, 1(3):320–331, 2010.
11. José Monteiro, Srinivas Devadas, Pranav Ashar, and Ashutosh Mauskar. Scheduling techniques to enable power management. In *Proceedings of the 33rd annual Design Automation Conference*, pages 349–352. ACM, 1996.
12. Walid Saad, Zhu Han, and H Vincent Poor. Coalitional game theory for cooperative micro-grid distribution networks. In *Communications Workshops (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.
13. Ditiro Setlhaolo, Xiaohua Xia, and Jiangfeng Zhang. Optimal scheduling of household appliances for demand response. *Electric Power Systems Research*, 116:24–28, 2014.
14. Goran Strbac. Demand side management: Benefits and challenges. *Energy policy*, 36(12):4419–4426, 2008.
15. Perukrishnen Vytelingum, Thomas D Voice, Sarvapali D Ramchurn, Alex Rogers, and Nicholas R Jennings. Agent-based micro-storage management for the smart grid. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1-Volume 1*, pages 39–46. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

16. Laurence A Wolsey. Mixed integer programming. *Wiley Encyclopedia of Computer Science and Engineering*, 2008.
17. Jiawei Zhu, Fabrice Lauri, Abderrafiaa Koukam, and Vincent Hilaire. Scheduling optimization of smart homes based on demand response. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 223–236. Springer, 2015.