A Realistic Dataset for the Smart Home Device Scheduling Problem for DCOPs

William Kluegel[1], Muhammad Aamir Iqbal[1], Ferdinando Fioretto[2], William Yeoh[1], and Enrico Pontelli[1]

[1] Department of Computer Science, New Mexico State University
{wkluegel,miqbal,wyeoh,epontell}@cs.nmsu.edu
[2] Department of Industrial and Operations Engineering, University of Michigan
{fioretto}@umich.edu

Abstract. The field of Distributed Constraint Optimization has gained momentum in recent years thanks to its ability to address various applications related to multi-agent cooperation. While techniques for solving Distributed Constraint Optimization Problems (DCOPs) are abundant and have matured substantially since the field inception, the number of DCOP realistic applications available to assess the performance of DCOP algorithms is lagging behind. To contrast this background we *(i)* introduce the Smart Home Device Scheduling (SHDS) problem, which describe the problem of coordinating smart devices schedules across multiple homes as a multi-agent system, *(ii)* detail the physical models adopted to simulate smart sensors, smart actuators, and homes' environments, and *(iii)* introduce a DCOP realistic benchmark for SHDS problems.

1 Introduction

Distributed Constraint Optimization Problems (DCOPs) [16,20,27] have emerged as one of the prominent agent models to govern the agents' autonomous behavior, where both algorithms and communication models are driven by the structure of the specific problem. Researchers have used DCOP algorithms to solve various multi-agent coordination and resource allocation problems, including meeting scheduling [13,29], power network management [12], and smart home appliances coordination [22].

Since the research field inception, a wide variety of algorithms has been proposed to solve DCOPs. DCOP algorithms are typically classified as either *complete* or *incomplete*, based on whether they can guarantee to find an optimal solution or they trade optimality for shorter execution times [6]. In addition, each of these classes can be categorized into several groups, depending on the degree of locality exploited by the algorithms (e.g., full decentralization or partial centralization) [11,14,21], the way local information is updated (e.g., synchronous [14,19,20] or asynchronous [5,10,16]), and the type of exploration process adopted (e.g., search-based [11,16,26,28], inference-based [5,20], or sampling-based [7,17,18]).

While techniques to solve DCOPs are abundant and have matured substantially since the field's inception, the number of realistic DCOP applications and benchmarks used to assess the performance of DCOP algorithms is lagging behind [9]. Typical DCOP algorithms are evaluated on artificial random problems, or simplified problems that are adapted to the often unrealistic assumptions made by DCOP algorithms

(e.g., that each agent controls exactly one variable, and that all problem constraints are binary). To evaluate the performance of DCOP algorithms, it is necessary to introduce realistic benchmarks of deployable applications.

Motivated by these issues, we recently introduced the *Smart Home Device Scheduling (SHDS)* problem [8], which formalizes the problem of coordinating smart devices (e.g., smart thermostats, circulator heating, washing machines) schedules across multiple smart homes as a multi-agent system (MAS). The SHDS problem is suitable to be modeled as a DCOP due to the presence of both complex individual agents' goals, describing homes' energy price consumption, as well as a collective agents' goal, capturing reduction in energy peaks.

In this document, we introduce a set of realistic synthetic benchmarks for the SHDS problem for DCOPs. We report the details of the physical models adopted to simulate smart home sensors and actuators, as well as home environments, and describe how the actuator's actions affect the environments of a home (e.g., home's temperature, cleanliness, humidity). The datasets, the models, and the source code used to generate the SHDS datasets are available at https://github.com/nandofioretto/SHDS_dataset.

1.1 DCOP

A *Distributed Constraint Optimization Problem (DCOP)* [16,27] is described by a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \mathcal{A}, \alpha \rangle$, where: $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of *variables*; $\mathcal{D} = \{D_1, \ldots, D_n\}$ is a set of finite *domains* (i.e., $x_i \in D_i$); $\mathcal{F} = \{f_1, \ldots, f_e\}$ is a set of *utility functions* (also called *constraints*), where $f_i : \times_{x_j \in \mathbf{x}^{f_i}} D_i \rightarrow \mathbb{R}_+ \cup \{-\infty\}$ and $\mathbf{x}^{f_i} \subseteq \mathcal{X}$ is the set of the variables (also called the *scope*) relevant to f_i; $\mathcal{A} = \{a_1, \ldots, a_p\}$ is a set of *agents*; and $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ is a function that maps each variable to one agent. f_i specifies the utility of each combination of values assigned to the variables in $\mathbf{x}^{f_i}$. A *partial assignment* σ is a value assignment to a set of variables $X_\sigma \subseteq \mathcal{X}$ that is consistent with the variables' domains. The utility $\mathcal{F}(\sigma) = \sum_{f \in \mathcal{F}, \mathbf{x}^f \subseteq X_\sigma} f(\sigma)$ is the sum of the utilities of all the applicable utility functions in σ. A *solution* is a partial assignment σ for all the variables of the problem, i.e., with $X_\sigma = \mathcal{X}$. We will denote with $\mathbf{x}$ a solution, while $\mathbf{x}_i$ is the value of x_i in $\mathbf{x}$. The goal is to find an optimal solution $\mathbf{x}^* = \text{argmax}_{\mathbf{x}} \mathcal{F}(\mathbf{x})$.

2 Scheduling Device in Smart Homes

A *Smart Home Device Scheduling (SHDS)* problem is defined by the tuple $\langle \mathbf{H}, \mathcal{Z}, \mathcal{L}, \mathbf{P}_H, \mathbf{P}_Z, H, \theta \rangle$, where: $\mathbf{H} = \{h_1, h_2, \ldots\}$ is a neighborhood of smart homes, capable of communicating with one another; $\mathcal{Z} = \cup_{h_i \in \mathbf{H}} \mathbf{Z}_i$ is a set of smart devices, where $\mathbf{Z}_i$ is the set of devices in the smart home h_i (e.g., vacuum cleaning robot, smart thermostat). $\mathcal{L} = \cup_{h_i \in \mathbf{H}} \mathbf{L}_i$ is a set of locations, where $\mathbf{L}_i$ is the set of locations in the smart home h_i (e.g., living room, kitchen); $\mathbf{P}_H$ is the set of state properties of the smart homes (e.g., cleanliness, temperature); $\mathbf{P}_Z$ is the set of devices state properties (e.g., battery charge for a vacuum robot); H is the planning horizon of the problem. We denote with $\mathbf{T} = \{1, \ldots, H\}$ the set of time points; $\theta : \mathbf{T} \rightarrow \mathbb{R}^+$ represents the real-time pricing schema adopted by the energy utility company, which expresses the

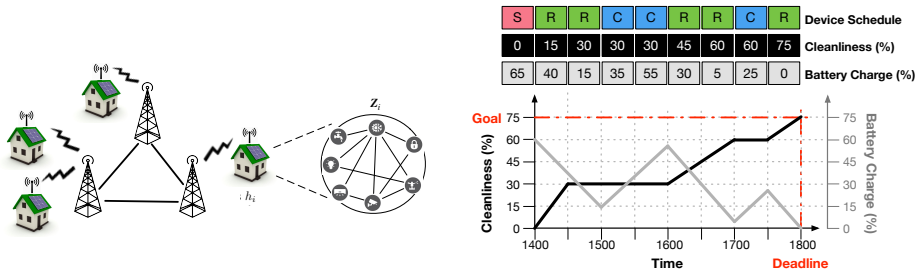

FIG. 1: Illustration of a Neighborhood of Smart Homes

cost per kWh of energy consumed by consumers. Finally, we use Ω_p to denote the set of all possible states for state property $p \in \mathbf{P}_H \cup \mathbf{P}_Z$ (e.g., all the different levels of cleanliness for the cleanliness property). Figure 1(right) shows an illustration of a neighborhood of smart homes with each home controlling a set of smart devices.

2.1 Smart Devices

For each home $h_i \in \mathbf{H}$, the set of smart devices $\mathbf{Z}_i$ is partitioned into a set of actuators $\mathbf{A}_i$ and a set of sensors $\mathbf{S}_i$. Actuators can affect the states of the home (e.g., heaters and ovens can affect the temperature in the home) and possibly their own states (e.g., vacuum cleaning robots drain their battery power when running). On the other hand, sensors monitor the states of the home. Each device $z \in \mathbf{Z}_i$ of a home h_i is defined by a tuple $\langle \ell_z, A_z, \gamma_z^H, \gamma_z^Z \rangle$, where $\ell_z \in \mathbf{L}_i$ denotes the relevant location in the home that it can act or sense, A_z is the set of actions that it can perform, $\gamma_z^H : A_z \to 2^{\mathbf{P}_H}$ maps the actions of the device to the relevant state properties of the home, and $\gamma_z^Z : A_z \to 2^{\mathbf{P}_Z}$ maps the actions of the device to its relevant state properties. We will use the following running example throughout this paper.

Example 1. Consider a vacuum cleaning robot z_v with location $\ell_{z_v} = \mathsf{living\_room}$. The set of possible actions is $A_{z_v} = \{\mathsf{run}, \mathsf{charge}, \mathsf{stop}\}$ and the mappings are:

$$\gamma_{z_v}^H\colon \mathsf{run} \to \{\mathsf{cleanliness}\};\ \mathsf{charge} \to \emptyset;\ \mathsf{stop} \to \emptyset$$
$$\gamma_{z_v}^Z\colon \mathsf{run} \to \{\mathsf{battery\_charge}\};\ \mathsf{charge} \to \{\mathsf{battery\_charge}\};\ \mathsf{stop} \to \emptyset$$

where $\emptyset$ represents a *null* state property.

2.2 Device Schedules

To control the energy profile of a smart home, we need to describe the behavior of the smart devices acting in the smart home during time. We formalize this concept with the notion of *device schedules*.

We use $\xi_z^t \in A_z$ to denote the action of device z at time step t, and $\xi_X^t = \{\xi_z^t \mid z \in X\}$ to denote the set of actions of the devices in $X \subseteq \mathcal{Z}$ at time step t.

Definition 1 (Schedule). *A schedule $\xi_X^{[t_a \to t_b]} = \langle \xi_X^{t_a}, \ldots, \xi_X^{t_b} \rangle$ is a sequence of actions for the devices in $X \subseteq \mathcal{Z}$ within the time interval from t_a to t_b.*

Consider the illustration of Figure 1(left). The top row of Figure 1(left) shows a possible schedule $\langle R, R, C, C, R, R, C, R\rangle$ for a vacuum cleaning robot starting at time 1400 hrs, where each time step is 30 minutes. The robot's actions at each time step are shown in the colored boxes with letters in them: red with 'S' for stop, green with 'R' for run, and blue with 'C' for charge.

At a high level, the goal of the SHDS problem is to find a schedule for each of the devices in every smart home that achieve some user-defined objectives (e.g., the home is at a particular temperature within a time window, the home is at a certain cleanliness level by some deadline) that may be personalized for each home. We refer to these objectives as *scheduling rules*.

2.3 Scheduling Rules

We define two types of scheduling rules: *Active scheduling rules (ASRs)* that define user-defined objectives on a desired state of the home (e.g., the living room is cleaned by 1800 hrs), and *Passive scheduling rules (PSRs)* that define implicit constraints on devices that must hold at all times (e.g., the battery charge on a vacuum cleaning robot is always between 0% and 100%). We provide a formal description for the grammar of scheduling rules in Section 3.4.

Example 2. The scheduling rule (1) describes an *ASR* defining a goal state where the living room floor is at least 75% clean (i.e., at least 75% of the floor is cleaned by a vacuum cleaning robot) by 1800 hrs:

$$\texttt{living\_room cleanliness} \geq 75 \texttt{ before } 1800 \tag{1}$$

$$z_v \texttt{ battery\_charge} \geq 0 \texttt{ always} \tag{2}$$

$$z_v \texttt{ battery\_charge} \leq 100 \texttt{ always} \tag{3}$$

and scheduling rules (2) and (3) describe *PSRs* stating that the battery charge of the vacuum robot z_v needs to be between 0 and 100 % of its full charge at all the times.

We denote with $R_p^{[t_a \to t_b]}$ a scheduling rule over a state property $p \in \mathbf{P}_H \cup \mathbf{P}_Z$, and time interval $[t_a, t_b]$. Each scheduling rule indicates a goal state at a location or on a device $\ell_{R_p} \in \mathbf{L}_i \cup \mathbf{Z}_i$ of a particular state property p that must hold over the time interval $[t_a, t_b] \subseteq \mathbf{T}$. The scheduling rule goal state is either a desired state of a home, if it is an ASR (e.g., the cleanliness level of the room floor) or a required state of a device or a home, if it is a PSR (e.g., the battery charge of the vacuum cleaning robot).

Each rule is associated with a set of actuators $\Phi_p \subseteq \mathbf{A}_i$ that can be used to reach the goal state. For instance, in our Example (2), Φ_p correspond to the vacuum cleaning robot z_v, which can operate on the living room floor. Additionally, a rule is associated with a sensor $s_p \in \mathbf{S}_i$ capable of sensing the state property p. Finally, in a PSRs the device can also sense its own internal states.

The ASR of Equation (1) is illustrated in Figure 1(left) by dotted red lines on the graph. The PSRs are not shown as they must hold for all time steps.

2.4 Feasibility of Schedules

To ensure that a goal state can be achieved across the desired time window the system uses a *predictive model* of the various state properties. This predictive model captures

the evolution of a state property over time and how such state property is affected by a given joint action of the relevant actuators. We describe the details of the physical predictive models used to generate our benchmark set in Section 3.3.

Definition 2 (Predictive Model). *A predictive model Γ_p for a state property p (of either the home or a device) is a function $\Gamma_p : \Omega_p \times \bigtimes_{z \in \Phi_p} A_z \cup \{\perp\} \rightarrow \Omega_p \cup \{\perp\}$, where $\perp$ denotes an infeasible state and $\perp + (\cdot) = \perp$.*

In other words, the model describes the transition of state property p from state $\omega_p \in \Omega_p$ at time step t to time step $t+1$ when it is affected by a set of actuators Φ_p running joint actions $\xi^t_{\Phi_p}$:

$$\Gamma_p^{t+1}(\omega_p, \xi^t_{\Phi_p}) = \omega_p + \Delta_p(\omega_p, \xi^t_{\Phi_p}) \tag{4}$$

where $\Delta_p(\omega_p, \xi^t_{\Phi_p})$ is a function describing the effect of the actuators' joint action $\xi^t_{\Phi_p}$ on state property p. We assume here, w.l.o.g. that the state of properties are numeric—when this is not the case, a mapping to the possible states to a numeric representation can be easily defined.

Notice that a recursive invocation of a predictive model allows us to predict the trajectory of a state property p for future time steps, given a schedule of actions of the relevant actuators Φ_p. Let us formally define this concept.

Definition 3 (Predicted State Trajectory). *Given a state property p, its current state ω_p at time step t_a, and a schedule $\xi^{[t_a \rightarrow t_b]}_{\Phi_p}$ of relevant actuators Φ_p, the predicted state trajectory $\pi_p(\omega_p, \xi^{[t_a \rightarrow t_b]}_{\Phi_p})$ of that state property is defined as:*

$$\pi_p(\omega_p, \xi^{[t_a \rightarrow t_b]}_{\Phi_p}) = \Gamma_p^{t_b}(\Gamma_p^{t_{b-1}}(\ldots(\Gamma_p^{t_a}(\omega_p, \xi^{t_a}_{\Phi_p}), \ldots), \xi^{t_{b-1}}_{\Phi_p}), \xi^{t_b}_{\Phi_p}) \tag{5}$$

Consider the device scheduling example in Figure 1(left). The predicted state trajectories of the *battery_charge* and *cleanliness* state properties are shown in the second and third rows of Figure 1(left). These trajectories are predicted given that the vacuum cleaning robot will take on the schedule shown in the first row of the figure. The predicted trajectories of these state properties are also illustrated in the graph, where the dark grey line shows the states for the robot's battery charge and the black line shows the states for the cleanliness of the room.

Notice that to verify if a schedule satisfies a scheduling rule, it is sufficient to check that the predicted state trajectories are within the set of feasible state trajectories of that rule. Additionally, notice that each active and passive scheduling rule defines a set of feasible state trajectories. For example, the active scheduling rule of Equation (1) allows all possible state trajectories as long as the state at time step 1800 is no smaller than 75. We use $R_p[t] \subseteq \Omega_p$ to denote the set of states that are feasible according to rule R_p of state property p at time step t. More formally, a schedule $\xi^{[t_a \rightarrow t_b]}_{\Phi_p}$ satisfies a scheduling rule $R_p^{[t_a \rightarrow t_b]}$ (written as $\xi^{[t_a \rightarrow t_b]}_{\Phi_p} \models R_p^{[t_a \rightarrow t_b]}$) iff:

$$\forall t \in [t_a, t_b] : \pi_p(\omega_p^{t_a}, \xi^{[t_a \rightarrow t]}_{\Phi_p}) \in R_p[t] \tag{6}$$

where $\omega_p^{t_a}$ is the state of state property p at time step t_a.

Definition 4 (Feasible Schedule). *A schedule is* feasible *if it satisfies* all *the passive and active scheduling rules of each home in the SHDS problem.*

In the example of Figure 1, the evaluated schedule is a feasible schedule since the trajectories of both the *battery_charge* and *cleanliness* states satisfy both the *active scheduling rule* (1) and the *passive scheduling rules* (2) and (3).

2.5 Optimization Objective

In addition to finding feasible schedules, the goal in the SHDS problem is to optimize for the aggregated total cost of energy consumed.

Each action $a \in A_z$ of device $z \in \mathbf{Z}_i$ in home $h_i \in \mathbf{H}$ has an associated energy consumption $\rho_z : A_z \to \mathbb{R}^+$, expressed in kWh. The aggregated energy $E_i^t(\xi_{\mathbf{Z}_i}^{[0\to H]})$ across all devices consumed by h_i at time step t under trajectory $\xi_{\mathbf{Z}_i}^{[0\to H]}$ is:

$$E_i^t(\xi_{\mathbf{Z}_i}^{[0\to H]}) = \sum_{z \in \mathbf{Z}_i} \rho_z(\xi_z^t) \tag{7}$$

where ξ_z^t is the action of device z at time t in the schedule $\xi_{\mathbf{Z}_i}^{[0\to H]}$. The cost $c_i(\xi_{\mathbf{Z}_i}^{[0\to H]})$ associated to schedule $\xi_{\mathbf{Z}_i}^{[0\to H]}$ in home h_i is:

$$c_i(\xi_{\mathbf{Z}_i}^{[0\to H]}) = \sum_{t \in \mathbf{T}} \left(\ell_i^t + E_i^t(\xi_{\mathbf{Z}_i}^{[0\to H]})\right) \cdot \theta(t) \tag{8}$$

where ℓ_i^t is the home background load produced at time t, which includes all non-schedulable devices (e.g., TV, refrigerator), and sensor devices, which are always active, and $\theta(t)$ is the real-time price of energy per kWh at time t.

The objective of an SHDS problem is that of minimizing the following weighted bi-objective function:

$$\min_{\xi_{\mathbf{Z}_i}^{[0\to H]}} \quad \alpha_c \cdot C^{\text{sum}} + \alpha_e \cdot E^{\text{peak}} \tag{9}$$

$$\text{subject to:} \qquad \forall h_i \in \mathbf{H}, R_p^{[t_a \to t_b]} \in \mathbf{R}_i : \quad \xi_{\Phi_p}^{[t_a \to t_b]} \models R_p^{[t_a \to t_b]} \tag{10}$$

where $\alpha_c, \alpha_e \in \mathbb{R}$ are weights, $C^{\text{sum}} = \sum_{h_i \in \mathbf{H}} c_i(\xi_{\mathbf{Z}_i}^{[0\to H]})$ is the aggregated monetary cost across all homes h_i; and $E^{\text{peak}} = \sum_{t \in \mathbf{T}} \sum_{\mathbf{H}_j \in \mathcal{H}} \sum_{h_i \in \mathbf{H}_j} \left(E_i^t(\xi_{\mathbf{Z}_i}^{[0\to H]})\right)^2$ is a quadratic penalty function on the aggregated energy consumption across all homes h_i. Since the SHDS problem is designed for distributed multi-agent systems, in a cooperative approach, optimizing E^{peak} may require each home to share its energy profile with every other home. To take into account data privacy concerns and possible high network loads, we decompose the set of homes $\mathbf{H}$ into neighboring subsets of homes $\mathcal{H}$, so that E^{peak} can be optimized independently within each subset. One can use coalition formation algorithms [23,24,25] to form such coalitions/subsets of neighboring homes. These coalitions can be exploited by a distributed algorithm to (1) parallelize computations between multiple groups and (2) avoid data exposure over long distances or sensitive areas. Finally, Constraint (10) defines the valid trajectories for each scheduling rule $r \in \mathbf{R}_i$, where $\mathbf{R}_i$ is the set of all scheduling rules of home h_i.

2.6 DCOP Mapping

One can map the SHDS problem to a DCOP as follows:

- AGENTS: Each agent $a_i \in \mathcal{A}$ in the DCOP is mapped to a home $h_i \in \mathbf{H}$.
- VARIABLES and DOMAINS: Each agent a_i controls the following set of variables:
  - For each actuator $z \in \mathbf{A}_i$ and each time step $t \in \mathbf{T}$, a variable $x^t_{i,z}$ whose domain is the set of actions in A_z. The sensors in $\mathbf{S}_i$ are considered to be always active, and thus not directly controlled by the agent.
  - An auxiliary interface variable $\hat{x}^t_j$ whose domain is the set $\{0, \ldots, \sum_{z \in \mathbf{Z}_i} \rho(\text{argmax}_{a \in A_z} \rho_z(a))\}$, which represents the aggregated energy consumed by all the devices in the home at each time step t.
- CONSTRAINTS: There are three types of constraints:
  - *Local* soft constraints (i.e., constraints that involve only variables controlled by the agent) whose costs correspond to the weighted summation of monetary costs, as defined in Equation (8).
  - *Local* hard constraints that enforce Constraint (10). Feasible schedules incur a cost of 0 while infeasible schedules incur a cost of ∞.
  - *Global* soft constraints (i.e., constraints that involve variables controlled by different agents) whose costs correspond to the peak energy consumption, as defined in the second term in Equation (9).

3 Model Parameters and Realistic Data Set Generation

This section describes the parameters and models adopted in our SHDS datasets generation. We first describe the house structural parameters, which are used in turn to calculate the house predictive models. Next, we report a detailed list of the smart devices adopted in our datasets, discussing their power consumptions and effects on the house environments. We then describe the predictive models adopted to capture changes in the house's environments and devices' states. Finally, we report the BNF for the scheduling rules introduced in Section 2.3, and the pricing scheme adopted in our experiments.

3.1 House Structural Parameters

We consider three house sizes (small, medium, and large). The floor plans for the three house structures are shown in Figure 2.

Our house structural model simplifies the floor plans shown in Figure 2 by ignoring internal walls. This abstraction is sufficient to capture the richness of the predictive models introduced in Section 2.4. Table 1 reports the parameters of the houses adopted in our SHDS dataset. The house sizes are expressed in meters ($L \times W$). The wall's height is assumed to be 2.4m and the window area denotes the area of the walls covered by windows. The overall heat transfer coefficient (also referred to as U-value) describes how well a building element conducts heat. It is defined as the rate of heat transfer (in watts) through one unit area (m^2) of a structure divided by the difference in temperature across the structure [15].

FIG. 2: Floor plans for a small (left), medium (center), and large (right) house.

Structural Parameters	small	medium	large	Structural Parameters	small	medium	large
house size (m)	6×8	8×12	12×15	U_{roof} (W/(m^2 °C))	1.1	1.1	1.1
walls area (m^2)	67.2	96	129.6	lights energy density (W/m^3)	9.69	9.69	9.69
window area (m^2)	7.2	10	16	background load (kW)	0.166	0.166	0.166
U_{walls} (W/(m^2 °C))	3.9	3.9	3.9	background heat gain (W)	50	50	50
U_{windows} (W/(m^2 °C))	2.8	2.8	2.8	people heat gain (Btu/h)	400	400	400

TABLE 1: House structural parameters.

The material of the walls is considered to be a 150mm poured concrete (1280 kg/m^3) with a heat-transfer coefficient (U_{walls}) of 3.9 $\frac{\text{W}}{\text{m}^2 \cdot \text{°C}}$. We consider vertical double glazed windows, with distance between glasses $30 - 60$mm whose heat-transfer coefficient (U_{windows}) is 2.8 $\frac{\text{W}}{\text{m}^2 \cdot \text{°C}}$. Additionally, we consider a 2.54cm wood roof with 2.54cm insulation, with heat-transfer coefficient (U_{roof}) of 1.1 $\frac{\text{W}}{\text{m}^2 \cdot \text{°C}}$. Finally, we consider a 5.08cm wood door, with heat-transfer coefficient of 2.6 $\frac{\text{W}}{\text{m}^2 \cdot \text{°C}}$. These are commonly adopted materials in the US house construction industry [15]. We assume a background load consumption which accounts of a medium-size refrigerator (120W), a wireless router (6W), and a set of light bulbs (collectively 40W) [15]. The heat gain from the background house appliances is computed according to [15] (Table 9.8). We consider the heat gain generated by two people staying in the house, and computed as in [15] (Table 9.7), assuming the metabolic rate of *light office work*.

3.2 Smart Devices

In this section, we report the complete list of smart devices (sensors and actuators) adopted by the smart homes in our SHDS datasets.

Sensors. Table 2 reports the sensors adopted in our SHDS problem. For each sensor, we report an identifier (ID), the state property (see Section 2.1) it senses and its location in the house. All sensors are considered to be constantly active, sensing a single state property at a location (e.g., an air temperature sensor is located in a room of the house, a charge sensor is located on a device).

Actuators. Table 3 reports the list of the actuators. It tabulates the type of actuator and its model, its possible actions, the power consumption (in kWh), the state properties affected by each of its action, and the effects (Δ) on the associated predictive models in the small, medium, and large house sizes. The latter represents the incremental

ID	State property	Location	ID	State property	Location
01	air temperature	house room	08	dish cleanliness	appliance
02	floor cleanliness (dust)	house room	09	air humidity	house room
03	temperature	appliance	10	luminosity	house room
04	battery charge	appliance	11	occupancy	house room
05	bake	appliance	12	movement	house room
06	laundry wash	appliance	13	smoke detector	house room
07	laundry dry	appliance			

TABLE 2: List of sensors.

Actuator	Model	Actions	Consumption (kWh)	State properties (ID)	Effects Small(Δ)	Effects Medium(Δ)	Effects Large(Δ)
Heater	Dyson AM09	off	0	{01}	$-\frac{L_b}{148.48 \cdot T_A}$	$-\frac{L_b}{296.86 \cdot T_A}$	$-\frac{L_b}{593.75 \cdot T_A}$
		fan	0.008	{01}	$-\frac{L_b}{148.48 \cdot T_A}$	$-\frac{L_b}{296.86 \cdot T_A}$	$-\frac{L_b}{593.75 \cdot T_A}$
		heat	0.025	{01}	$\frac{L_b}{148.48 \cdot \lvert T_Z - T_A \rvert}$	$\frac{L_b}{296.86 \cdot \lvert T_Z - T_A \rvert}$	$\frac{L_b}{593.75 \cdot \lvert T_Z - T_A \rvert}$
AC	Bryant 697CN030B	off	0	{01}	$\frac{L_b}{148.48 \cdot T_A}$	$\frac{L_b}{296.86 \cdot T_A}$	$\frac{L_b}{593.75 \cdot T_A}$
		fan	0.012	{01}	$\frac{L_b}{148.48 \cdot T_A}$	$\frac{L_b}{296.86 \cdot T_A}$	$\frac{L_b}{593.75 \cdot T_A}$
		cool	0.037	{01}	$\frac{L_b}{148.48 \cdot \lvert T_A - T_Z \rvert}$	$\frac{L_b}{296.86 \cdot \lvert T_A - T_Z \rvert}$	$\frac{L_b}{593.75 \cdot \lvert T_A - T_Z \rvert}$
Waterheater	Tempra 36	off	0	{03}	{0}	{0}	{0}
		on	0.060	{03}	$\{9.90°C\}$	$\{8.94°C\}$	$\{6.83°C\}$
Vacuum Bot	iRobot Roomba 880	off	0	{02, 04}	{0.0%, 0.0%}	{0.0%, 0.0%}	{0.0%, 0.0%}
		vacuum	0	{02, 04}	{0.676%, −0.21%}	{0.338%, −0.21%}	{0.168%, −0.21%}
		charge	0.004	{04}	{0.33%}	{0.33%}	{0.33%}
Electric Vehicle	Tesla Model S	off	0	{04}	{0}	{0}	{0}
		48 amp wall charger	0.192	{04}	{0.226%}	{0.226%}	{0.226%}
		72 amp wall charger	0.333	{04}	{0.333%}	{0.333%}	{0.333%}
		Super charger	120	{04}	{2.326%}	{2.326%}	{2.326%}
Clothes Washer	GE WSM2420D3WW	off	0	{06}	{0}	{0}	{0}
		wash (Regular)	0.007	{06}	{1}	{1}	{1}
		spin (Regular)	0.008	{06}	{1}	{1}	{1}
		rinse (Regular)	0.008	{06}	{1}	{1}	{1}
		wash (Perm-Press)	0.007	{06}	{1}	{1}	{1}
		spin (Perm-Press)	0.007	{06}	{1}	{1}	{1}
		rinse (Perm-Press)	0.008	{06}	{1}	{1}	{1}
		wash (Delicates)	0.007	{06}	{1}	{1}	{1}
		spin (Delicates)	0.007	{06}	{1}	{1}	{1}
		rinse (Delicates)	0.008	{06}	{1}	{1}	{1}
Clothes Dryer	GE WSM2420D3WW	off	0	{07}	{0}	{0}	{0}
		on (Regular)	0.027	{07}	{1}	{1}	{1}
		on (Perm-Press)	0.024	{07}	{1}	{1}	{1}
		on (Timed)	0.028	{07}	{1}	{1}	{1}
Oven	Kenmore 790.91312013	off	0	{05}	{0}	{0}	{0}
		bake	0.037	{05, 01}	$\{1, 0.017°C\}$	$\{1, 0.009°C\}$	$\{1, 0.004°C\}$
		broil	0.042	{05, 01}	$\{1.25, 0.02°C\}$	$\{1.25, 0.01°C\}$	$\{1.25, 0.005°C\}$
Dishwasher	Kenmore 665.13242K900	off	0	{08}	{0}	{0}	{0}
		wash	0.006	{08}	{1}	{1}	{1}
		rinse	0.009	{08}	{1}	{1}	{1}
		dry	0.006	{08}	{1}	{1}	{1}

TABLE 3: List of actuators.

quantity that affects the physical system, given the action of the actuator, as defined in Equation (4). We detail the calculation of the house and devices physical models below.

3.3 Physical Models

In this section, we describe the physical models used to compute the effects Δ of the actuators' actions on a predictive model (see Table 3). These values, in turn, are adopted within the SHDS predictive models as described in Equation (4).

Battery (Dis)charge Model. The battery charge/discharge model adopted in our work for the battery-powered devices is as follows. For a given battery b with capacity Q_b (expressed in kWh), voltage V_b, and electric charge $E_b = \frac{V_b}{Q_b}$ (expressed in ampere-hour (Ah)), and assuming a 100% charging/discharging efficiency, the battery charge time b_α^+ and discharge time b_α^- are computed, respectively, as:

$$b_\alpha^+ = \frac{E_b}{C^+}; \qquad b_\alpha^- = \frac{E_b}{C^-}, \tag{11}$$

	Tesla Model S			iRobot Roomba 880
	Slow Charge	Regular Charge	Super Charger	
V_b	240	240	240	120
E_b	354 Ah	354 Ah	354 Ah	3 Ah
C^+	48 A	72 A	500 A	1.25 A
C^-	60 A	60 A	60 A	0.75 A
b_α^+	7 hr 22 min	5 hr	43 min	2 hr 24 min
b_α^-	6 hr	6 hr	6 hr	4 hr

TABLE 4: Electric vehicles [3] and robotic vacuum cleaner[1] batteries physical model.

and expressed in hours. C^+ and C^- are, respectively, the charging amperage and the in-use amperage. In Table 4, we report the battery model parameters associated to our electric vehicle and to our robotic vacuum cleaner. These parameters are derived following the products' manuals [1] and [3], respectively. The devices' action effects Δ associated to the charging time and discharging time are computed by dividing the total charging time and discharging time by $|\mathbf{T}|$.

Air Temperature Model. The air temperature predictive model is computed following the standard principle of heating and ventilation [15], and described as follows. Let G be the ventilation conductance: $G = \dot{V} \cdot \rho_a \cdot \bar{h}$, where $\dot{V}$ is the volume flow rate, whose value is set to 100, ρ_a is the density of the air, set to 0.75, and $\bar{h}$ is the specific heat of the air, set to 0.24, following [15]. The house heat loss coefficient h_{loss} is expressed as:

$$h_{\text{loss}} = U_{\text{walls}} \cdot A_{\text{walls}} + U_{\text{roof}} \cdot A_{\text{roof}} + U_{\text{windows}} \cdot A_{\text{windows}} + G \tag{12}$$

where U_{walls}, U_{roof}, and U_{windows} describe the heat transfer coefficients for the walls, roof, and windows of the house, respectively, and A_{walls}, A_{roof}, and A_{windows} describe the areas of the walls, roof, and windows, respectively. Their values are provided in Table 1. Let T_A by the current temperature and T_Z be a target temperature; the heating load $\dot{L}_h$ is given by:

$$\dot{L}_h = h_{\text{loss}} |T_Z - T_A| \tag{13}$$

The heating load defines the quantity of heat per unit time (in BTU) that must be supplied in a building to reach a target temperature T_Z, from the given temperature T_A. Given the heating load $\dot{L}_h$ and the heater capacity C of a heater/cooler, the time required for a device to operate so to reach the desired temperature is given by: $\frac{L_h}{C}$.

The heating/cooling load is also effected by the outdoor and indoor temperature difference. Consider the example where T_A = 12°C and T_Z = 22°C, and the outdoor temperature changes from T_A to T_N = 8°C. We can calculate the new load due to change in temperature using the following:

$$\dot{L}_n = \dot{L}_h \cdot \frac{|T_Z - T_N|}{|T_Z - T_A|}. \tag{14}$$

The above expression shows that an outdoor temperature drops of 4°C, causes the heating load to increase by a factor of 1.4 (w.r.t. the previous heating load T_A). In our model we need to compute the change in temperature per time step (Δ). This can be done using the heat-loss relationship:

$$\Delta = \frac{h_{\text{loss}}}{m \cdot c_p}, \tag{15}$$

time start	0:00	8:00	12:00	14:00	18:00	22:00
time end	7:59	11:59	13:59	17:59	21:59	23:59
price ($)	0.198	0.225	0.249	0.849	0.225	0.198

TABLE 5: Pacific Gas & Electric Co. pricing schema

where m is the mass of the air and c_p is the specific heat of air. In our model, m depends on volume flow rate of an air in the house, and c_p = 1kJ/kg·K.

Water Temperature Model. The rise in the water temperature per unit of time (Δ value) is dependent on the difference in the water temperature flowing into the water heater and the amount of water flowing out of the water heater, as well as water usage. We considered an on-demand electric water heater (tankless). The water usage depends on household size and the activities of multiple users. In our model, to compute the rise and drop in water temperature, we adopted the highest potential peak in households water usage following [2,4], and corresponding to 26.50 liters/min (small house), 29.34 liters/min (medium house), and 38.38 liters/min (large house). The rise in temperature is $18.33°C$ for 14.31 liters/minute of water usage [2]. Thus the rise in temperature for our small, medium, and large house, are, respectively, $9.90°C$, $8.94°C$, and $6.83°C$.

Cleanliness Model. Our floor cleanliness model is computed using the following equation: $T = \frac{A}{0.313}$, where A represents the area of the room (in m^2) and T is the amount of time (in minutes) required by a robotic vacuum cleaner to vacuum the entire room. A robotic vacuum cleaner *iRobot Roomba 880* is estimated to cover a 17.84 m^2 room in 57 minutes [1] (which is approximately 0.313 m^2/min). In our proposed dataset we use three different areas: $A_{small} = 48m^2$, $A_{medium} = 96m^2$, and $A_{large} = 180m^2$. Thus the estimated times to cover a 100% floor for the small, medium, and large houses are, respectively: $T = 153.35$, 306.71, and 575.08 minutes. The corresponding Δ value of Table 3 (which is a percentage) is computed as: $\Delta = \frac{100\%}{T}$

All other predictive models (e.g., laundry wash and dry, bake, dish cleanliness, etc.) simply capture the time needed for a device to achieve the required goals by checking that accumulated device effects achieves the desired property. The specifics for such values are provided in the dataset generation, in Section 4.

3.4 Scheduling Rules

We now report the complete Backus-Naur Form (BNF) for the *scheduling rules* for a smart home $h_i \in \mathbf{H}$, introduced in Section 2.2

$$
\begin{aligned}
\langle \textit{rules} \rangle &::= \langle \textit{simple rule} \rangle \mid \langle \textit{simple rule} \rangle \wedge \langle \textit{rules} \rangle \\
\langle \textit{simple rule} \rangle &::= \langle \textit{active rule} \rangle \mid \langle \textit{passive rule} \rangle \\
\langle \textit{active rule} \rangle &::= \langle \textit{location} \rangle \langle \textit{state property} \rangle \langle \textit{relation} \rangle \langle \textit{goal state} \rangle \langle \textit{time} \rangle \\
\langle \textit{passive rule} \rangle &::= \langle \textit{location} \rangle \langle \textit{state property} \rangle \langle \textit{relation} \rangle \langle \textit{goal state} \rangle \\
\langle \textit{location} \rangle &::= \ell \in \mathbf{L}_i \\
\langle \textit{state property} \rangle &::= s \in \mathbf{P}_H \mid s \in \mathbf{P}_Z \\
\langle \textit{relation} \rangle &::= \leq \mid < \mid = \mid \neq \mid > \mid \geq \\
\langle \textit{goal state} \rangle &::= \textit{sensor state} \mid \textit{actuator state}
\end{aligned}
$$

⟨*location*⟩	⟨*state property*⟩	⟨*relation*⟩	⟨*goal state*⟩	⟨*time*⟩
Room	*air temperature*	$r \in \{>, \geq\}$	$g_1 \in [17, 24]$	⟨*time*⟩
Room	*floor cleanliness*	$r \in \{>, \geq\}$	$g_2 \in [50, 80]$	⟨*time*⟩
Electric Vehicle	*charge*	$r \in \{>, \geq\}$	$g_3 \in [50, 65]$	⟨*time*⟩
Water heater	*temperature*	$r \in \{>, \geq\}$	$g_4 \in [15, 40]$	⟨*time*⟩
Clothes Washer	*laundry wash*	$r \in \{\geq\}$	$g_5 \in \{45, 60\}$	⟨*time*⟩
Clothes Dryer	*laundry dry*	$r \in \{\geq\}$	$g_6 \in \{45, 60\}$	⟨*time*⟩
Oven	*bake*	$r \in \{=\}$	$g_7 \in \{60, 75, 120, 150\}$	⟨*time*⟩
Dishwasher	*dish cleanliness*	$r \in \{\geq\}$	$g_8 \in \{45, 60\}$	⟨*time*⟩

TABLE 6: Scheduling (active) rules

⟨*location*⟩	⟨*state property*⟩	⟨*relation*⟩	⟨*goal state*⟩	⟨*location*⟩	⟨*state property*⟩	⟨*relation*⟩	⟨*goal state*⟩
Room	*air temperature*	$\geq$	0	*EV*	*charge*	$\leq$	100
Room	*air temperature*	$\leq$	33	*Water heater*	*temperature*	$\geq$	10
Room	*floor cleanliness*	$\geq$	0	*Water heater*	*temperature*	$\leq$	55
Room	*floor cleanliness*	$\leq$	100	*Oven*	*bake*	$\leq$	g_7
Roomba	*charge*	$\geq$	0	*Clothes Washer*	*laundry wash*	$\leq$	g_6
Roomba	*charge*	$\leq$	100	*Clothes Dryer*	*laundry dry*	$\leq$	g_7
EV	*charge*	$\geq$	0	*Dishwasher*	*dish cleanliness*	$\leq$	g_8

TABLE 7: Scheduling (passive) rules

⟨*time*⟩ ::= at ⟨*T*⟩ | before ⟨*T*⟩ | after ⟨*T*⟩ | within [⟨*T*⟩, ⟨*T*⟩] | for ⟨*T*⟩ time units

⟨*T*⟩ ::= $t \in \mathbf{T}$

In our dataset the device states are mapped to numeric values, i.e., $\Omega_p = \mathbb{N}$, for all $p \in \mathbf{P}_H \cup \mathbf{P}_Z$.

3.5 Pricing Schema

For the evaluation of our SHDS datasets we adopted a pricing schema used by the Pacific Gas & Electric Co. for its customers in parts of California,[3] which accounts for 7 tiers ranging from \$0.198 per kWh to \$0.849 per kWh, reported in Table 5.

4 SHDS Dataset

We now introduce a dataset for the SHDS problem for DCOPs. We generate synthetic microgrid instances sampling neighborhoods in three cities in the United States (Des Moines, IA; Boston, MA; and San Francisco, CA) and estimate the density of houses in each city. The average density (in houses per square kilometers) is 718 in Des Moines, 1357 in Boston, and 3766 in San Francisco. For each city, we created a 200m$\times$200m grid, where the distance between intersections is 20m, and randomly placed houses in this grid until the density is the same as the sampled density. We then divided the city into k (=$|\mathcal{H}|$) coalitions, where each home can communicate with all homes in its coalition. Finally, we ensure that there are no disjoint coalitions; this is analogous to the fact that microgrids are all connected to each other via the main power grid.

We generate a total of 624 problem instances, where, for each city, we vary the number of agents—up to 7532 for the largest instances—the number of coalitions from

[3] https://goo.gl/vOeNqj

Physical model	Parameter	Value (small house)	Value (medium house)	Value (large house)
Air Temperature	$\dot{V}$	100	200	400
	m	148.48	296.86	593.75
	c_p	1.0	1.0	1.0
	ρ_a	0.75	0.75	0.75
	$\bar{h}$	0.24	0.24	0.24
	h_loss	352.24	544	764.75
	T_Z	22	22	22
	T_A	10	10	10
	$\dot{L}_n$	4226.88	6528	9177
Floor Cleanliness	A	48 m^2	96 m^2	180 m^2
	T	153.35 min	306.71 min	575.08 min
	Δ	0.652%	0.326%	0.174%
Water Temperature	household size	2	3	4
	liters/min usage	26.50	29.34	38.38
	Δ	27.9°C	25.2°C	19.2°C

TABLE 8: Physical models: Values and assumptions

1 to 1024, and the number of devices controlled by each house agent (from 4 to 20). The SHDS datasets is available at https://github.com/nandofioretto/SHDS_dataset.

Each home device has an associated active scheduling rule that is randomly generated and a number of passive rules that must always hold. The parameters used to generate active rules and passive rules are reported, respectively, in Tables 6 and 7. The time predicates associated with these rules are generated at random within the given horizon. Additionally, the relations r and goals states g_i are randomly generated by sampling from the sets corresponding, respectively, to the columns $\langle$*relation*$\rangle$ and $\langle$*goal state*$\rangle$ of Table 6.

Table 9 reports the results of the SHDS experiments for a subset of the Des Moines, Boston, and San Francisco instances, where we vary the number of agents (n)—up to 474 for the largest instances—while retaining the number of coalitions $k = 1$. To solve these instances we use an *uncoordinated* approach, where agents solve their private scheduling subproblem without coordinating their actions with those of other agents, and thus, disregarding the energy peak minimization objective. Each agent reports the best schedule found with a local Constraint Programming solver[4] as subroutine within a 10 seconds timeout. The row *obj* of Table 9 reports the upper bounds for the SHDS objective function, while the rows *avg price*, *avg power*, and *largest peak*, report, respectively, the average cost of the schedule, in US dollars, the average energy consumption, in kWh, and the largest peak (in kWh) produced during the day. For our experiments, we set $H = 12$, and report a summary of the parameters' settings adopted in our smart homes physical models, in Table 8. In these experiments, we notice that a large portion of the houses power consumption is caused by charging electric vehicles' batteries.

5 Conclusions

With the proliferation of smart devices, the automation of smart home scheduling can be a powerful tool for demand-side management within the smart grid vision. In this

[4] We adopt the JaCoP solver (http://www.jacop.eu/)

instance	n	obj	avg price ($)	avg energy (kWh)	largest peak (kWh)
dm_7	7	29227.05	3.31	16.04	299.3
dm_21	21	81841.35	3.31	15.77	885.8
dm_35	35	136696.19	3.28	15.76	1479.5
dm_71	71	287989.80	3.32	15.96	3015.8
dm_251	251	1006807.18	3.32	15.92	10622.5
bo_13	13	50493.74	3.33	15.89	534.6
bo_40	40	163246.01	3.34	16.15	1722.5
bo_67	67	272651.41	3.33	16.03	2844.1
bo_135	135	534692.07	3.31	15.90	5694.7
bo_474	474	1890711.09	3.31	15.92	19969.5
sf_37	37	149964.95	3.33	16.01	1563.4
sf_112	112	450723.92	3.32	15.97	4778.3
sf_188	188	750741.31	3.31	15.89	7904.1
sf_376	376	1486321.71	3.30	15.84	15669.0

TABLE 9: Des Moines, Boston, and San Francisco

paper we proposed the *Smart Home Device Scheduling (SHDS)* problem, which formalizes the device scheduling and coordination problem across multiple smart homes as a multi-agent system, and its mapping to a DCOP. Furthermore, we described in great detail the physical models adopted to model the smart home's sensors and actuators, as well as the physical model regulating the effect of the devices actions on the house environments properties (e.g., temperature, cleanliness). Finally, we reported a realistic dataset for the SHDS problem for DCOPs which includes 624 instances of increasing difficulty. We hope that the DCOP community will find this dataset useful in their empirical evaluations.

Acknowledgments

This research is partially supported by NSF grants 1345232 and 0947465. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References

1. Roomba 880 specs. http://www.consumerreports.org/products/robotic-vacuum/roomba-880-290102/specs/. [Online; accessed 18-February-2017].
2. Sizing a new water heater. https://www.energy.gov/energysaver/sizing-new-water-heater. [Online; accessed 18-February-2017].
3. Tesla model s specifics. https://www.tesla.com/models.
4. Typical water used in normal home activities. http://www.pittsfield-mi.gov/DocumentCenter/View/285. [Online; accessed 18-February-2017].
5. A. Farinelli, A. Rogers, A. Petcu, and N. Jennings. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *AAMAS*, pages 639–646, 2008.

6. F. Fioretto, E. Pontelli, and W. Yeoh. Distributed constraint optimization problems and applications: A survey. *CoRR*, abs/1602.06347, 2016.
7. F. Fioretto, W. Yeoh, and E. Pontelli. A dynamic programming-based MCMC framework for solving DCOPs with GPUs. In *CP*, pages 813–831, 2016.
8. F. Fioretto, W. Yeoh, and E. Pontelli. A multiagent system approach to scheduling devices in smart homes. In *AAMAS*, page (to appear), 2017.
9. E. C. Freuder and B. O'Sullivan. Grand challenges for constraint programming. *Constraints*, 19(2):150–162, 2014.
10. A. Gershman, A. Meisels, and R. Zivan. Asynchronous Forward-Bounding for distributed COPs. *Journal of Artificial Intelligence Research*, 34:61–88, 2009.
11. K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *CP*, pages 222–236, 1997.
12. A. Kumar, B. Faltings, and A. Petcu. Distributed Constraint Optimization with Structured Resource Constraints. In *AAMAS*, pages 923–930, 2009.
13. R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Event Scheduling. In *AAMAS*, pages 310–317, 2004.
14. R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS*, pages 438–445, 2004.
15. J. W. Mitchell and J. E. Braun. *Principles of Heating, Ventilation and Air Conditioning in Buildings*. Wiley, 2012.
16. P. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1–2):149–180, 2005.
17. D. T. Nguyen, W. Yeoh, and H. C. Lau. Distributed Gibbs: A memory-bounded sampling-based DCOP algorithm. In *AAMAS*, pages 167–174, 2013.
18. B. Ottens, C. Dimitrakakis, and B. Faltings. DUCT: An upper confidence bound approach to distributed constraint optimization problems. In *AAAI*, pages 528–534, 2012.
19. J. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *IJCAI*, pages 1446–1451, 2007.
20. A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 1413–1420, 2005.
21. A. Petcu, B. Faltings, and R. Mailler. PC-DPOP: A new partial centralization algorithm for distributed optimization. In *IJCAI*, pages 167–172, 2007.
22. P. Rust, G. Picard, and F. Ramparany. Using message-passing DCOP algorithms to solve energy-efficient smart environment configuration problems. In *IJCAI*, pages 468–474, 2016.
23. T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1):209–238, 1999.
24. O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.
25. T. Voice, M. Polukarov, and N. Jennings. Coalition structure generation over graphs. *Journal of Artificial Intelligence Research*, 45:165–196, 2012.
26. W. Yeoh, A. Felner, and S. Koenig. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. *Journal of Artificial Intelligence Research*, 38:85–133, 2010.
27. W. Yeoh and M. Yokoo. Distributed problem solving. *AI Magazine*, 33(3):53–65, 2012.
28. W. Zhang, G. Wang, Z. Xing, and L. Wittenberg. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1–2):55–87, 2005.
29. R. Zivan, S. Okamoto, and H. Peled. Explorative anytime local search for distributed constraint optimization. *Artificial Intelligence*, 212:1–26, 2014.