A Multiagent System Approach to Scheduling Devices in Smart Homes

Ferdinando Fioretto
University of Michigan
Ann Arbor, MI, USA
fioretto@umich.edu

William Yeoh
New Mexico State University
Las Cruces, NM, USA
wyeoh@cs.nmsu.edu

Enrico Pontelli
New Mexico State University
Las Cruces, NM, USA
epontell@cs.nmsu.edu

ABSTRACT

Demand-side management (DSM) in the smart grid allows customers to make autonomous decisions on their energy consumption, helping energy providers to reduce the energy peaks in load demand. The automated scheduling of smart devices in residential and commercial buildings plays a key role in DSM. Due to data privacy and user autonomy, such an approach is best implemented through distributed multi-agent systems. This paper makes the following contributions: **(i)** It introduces the *Smart Home Device Scheduling (SHDS)* problem, which formalizes the device scheduling and coordination problem across multiple smart homes as a multi-agent system; **(ii)** It describes a mapping of this problem to a *distributed constraint optimization problem*; **(iii)** It proposes a distributed algorithm for the SHDS problem; and **(iv)** It presents empirical results from a *physically distributed system* of Raspberry Pis, each capable of controlling smart devices through hardware interfaces, as well as larger scale synthetic experiments.

Keywords

DCOP; Smart Homes; Demand-Side Management

1. INTRODUCTION

Demand-side management (DSM) in the smart grid allows customers to make autonomous decisions on their energy consumption, helping the energy providers to reduce the peaks in energy load demand. Typical approaches for DSM focus on enforcing grid users' decisions to reduce consumptions by either *(i)* storing energy during off-peak hours and using the stored energy when the grid load demand is high, or *(ii)* scheduling shiftable loads in off-pick hours [26, 11, 3]. The former approach requires that homeowners own storage devices, in the form of batteries or electric vehicles. The availability of these resources, in the current and near future smart grid scenarios, is, however, limited. Thus, the latter approach is more appealing for the current smart grid scenario but requires producers to control a portion of the consumers' electrical appliances, which strongly affects privacy and users' autonomy.

On the other hand, residential and commercial buildings are progressively being partially automated through the introduction of smart devices (e.g., smart thermostats, circulator heating, washing machines). In addition, a variety of smart plugs, that allow users to intelligently control devices by remotely switching them on and off, are now commercially available. Device scheduling can, therefore, be executed by users, without the control of a centralized authority. This schema can, thus, be used for demand-side management. However, uncoordinated scheduling may be detrimental to DSM performance without reducing peak load demands [23]. For an effective DSM, a coordinated device scheduling within a neighborhood of buildings is necessary. Yet, privacy concerns arise when users share resources or cooperate to find suitable schedules.

Motivated by these issues, in this paper, we provide the following contributions: **(i)** We introduce the *Smart Home Device Scheduling (SHDS)* problem, which formalizes the problem of coordinating smart devices schedules across multiple smart homes as a multi-agent system (MAS); **(ii)** We propose a modeling of the SHDS problem as a *distributed constraint optimization problem*, where the agents are assumed to be cooperative; **(iii)** We propose a simple effective coordination scheme to solve this problem in a distributed manner; and **(iv)** We evaluate this distributed algorithm on a *physically distributed system* of Raspberry Pis, each capable of controlling a number of smart devices through hardware interfaces, such as Z-Wave dongles, as well as on larger scale synthetic experiments.

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

2. BACKGROUND AND RELATED WORK

The problem of scheduling devices in smart homes has recently attracted large interest within the AI and smart grid communities. Georgievski *et al.* [4] proposed a system to monitor and control electrical appliances in a home with the objective of reducing the energy consumption costs. Scott *et al.* [18] also studied a centralized online stochastic optimization approach for (single) home automation systems as a DSM mechanism, where future prices, occupant behavior, and environmental conditions are uncertain. Sou *et al.* [20] proposed a Mixed Integer Linear Program (MILP) to address smart appliances scheduling problem in single homes using a fine granularity for the technical specification of the smart appliances (e.g., they distinguish the different energy phases expressed by a dishwasher or a washing machine cycle). Due to the high complexity of the problem, the authors suggest adopting suboptimal solutions to reduce the overall resolution time. Another proposal to enhance the resolution time of a MILP formulation for scheduling smart devices has been presented by Tsui and Chan [21] through a convex re-

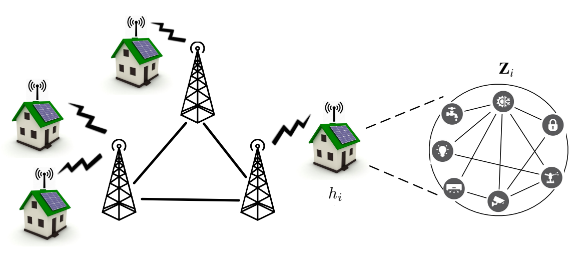

Figure 1: Illustration of a Neighborhood of Smart Homes

laxation for the automatic load management of appliances in a smart home. Such an approach, however, provides no guarantees on the solution quality with respect to the original problem. Unlike our approach, these proposals focus on single-home problems and/or are inherently centralized. In contrast, we focus on a distributed approach applied to multiple homes.

Researchers have also used distributed constraint optimization problem (DCOP) algorithms to solve resource allocation problems in the smart grid. For example, DCOP solutions to electric vehicle charging problems have been proposed in [10, 12], and Rust *et al.* [16] recently proposed a DCOP-based approach for coordinating the states of home appliances in a smart environment. Different from such approaches, in our proposal, we use a DCOP framework to schedule smart devices in smart homes as a proxy to coordinate the energy consumption of multiple smart homes with the goal of minimizing user costs and energy consumption peaks.

3. SCHEDULING 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:

- $\mathbf{H} = \{h_1, h_2, \dots\}$ 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 the state properties of the smart homes (e.g., cleanliness, temperature).
- $\mathbf{P}_Z$ is the set of the devices state properties (e.g., battery charge for a vacuum cleaning robot).
- H is the planning horizon of the problem. We denote with $\mathbf{T} = \{1, \dots, H\}$ the set of time points.
- $\theta : \mathbf{T} \to \mathbb{R}^+$ represents the real-time pricing schema adopted by the energy utility company, which expresses the 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 shows an illustration of a neighborhood of smart homes with each home controlling a set of smart devices.

3.1 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.

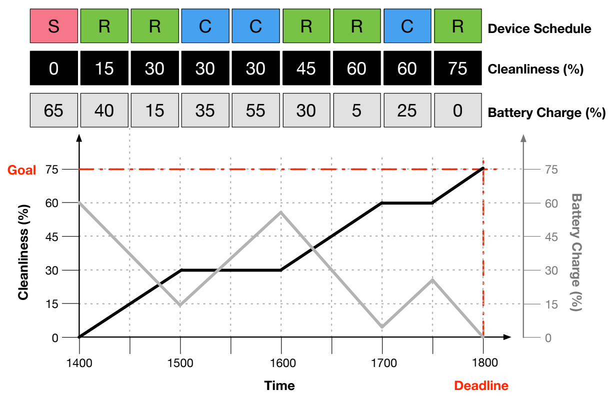

Figure 2: Smart Home Device Scheduling Example

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, and $\gamma_z^H : A_z \to 2^{\mathbf{P}_H}$ and $\gamma_z^Z : A_z \to 2^{\mathbf{P}_Z}$ map the actions of the device to the relevant state properties of the home and to those of the device, respectively.

We use the following example throughout this paper.

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

$$\begin{aligned} \gamma_{z_v}^H : \mathsf{run} &\to \{cleanliness\}; \\ \mathsf{charge} &\to \emptyset; \\ \mathsf{stop} &\to \emptyset. \\ \gamma_{z_v}^Z : \mathsf{run} &\to \{battery\_charge\}; \\ \mathsf{charge} &\to \{battery\_charge\}; \\ \mathsf{stop} &\to \emptyset, \end{aligned}$$

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

3.2 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 the time horizon. 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 $\xi_X^{[t_a \to t_b]} = \langle \xi_X^{t_a}, \dots, \xi_X^{t_b} \rangle$ 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 2. The top row of Figure 2 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.*

3.3 Scheduling Rules

We introduce 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).
- *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%).

The following syntax is used to express scheduling rules:[1]

$$\langle location \rangle \; \langle state \;\; property \rangle \; \langle relation \rangle \; \langle state \rangle \; \langle time \rangle.$$

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:*

$$\texttt{living\_room cleanliness} \geq \texttt{75 before 1800} \qquad (1)$$

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

$$z_v \;\; \texttt{battery\_charge} \;\geq\; \texttt{0 always} \qquad (2)$$

$$z_v \;\; \texttt{battery\_charge} \;\leq\; \texttt{100 always} \qquad (3)$$

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 home 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's goal state is either the 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 PSR, the device can also sense its own internal states. More formally,

$$\Phi_p = \{z \in \mathbf{A}_i \mid \ell_z = \ell_{R_p} \wedge \exists a \in A_z : p \in \gamma_z^H(a)\} \qquad (4)$$

$$\Phi_p = \{z \in \mathbf{A}_i \mid z = \ell_{R_p} \vee \ell_z = \ell_{R_p} \wedge \exists a \in A_z : p \in \gamma_z^H(a)\} \qquad (5)$$

where the former is associated to an ASR and the latter to a PSR. The ASR of Equation (1) is illustrated in Figure 2 by dotted red lines on the graph. The PSRs are not shown as they must hold for all time steps.

3.4 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 this state property is affected by a given joint action of the relevant actuators.

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

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 $\xi^t_{\Phi_p}$:

$$\Gamma_p^{t+1}(\omega_p, \xi^t_{\Phi_p}) = \omega_p + \Delta_p(\omega_p, \xi^t_{\Phi_p}) \qquad (6)$$

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, without loss of generality, that the state of properties is numeric—when this is not the case, a mapping of the possible states to a numeric representation can be easily defined.

Example 3. *Consider the* battery_charge *state property of the vacuum cleaning robot* z_v*. Assume it has* 65% *charge at time step* t *and its action is* $\xi^t_{z_v}$ *at that time step. Thus:*

$$\Gamma^{t+1}_{battery\_charge}(65, \xi^t_{z_v}) = 65 + \Delta_{battery\_charge}(65, \xi^t_{z_v}) \qquad (7)$$

$$\Delta_{battery\_charge}(\omega, \xi^t_{z_v}) = \begin{cases} \min(20, 100 - \omega) & \text{if } \xi^t_{z_v} = \textit{charge} \wedge \omega < 100 \\ -25 & \text{if } \xi^t_{z_v} = \textit{run} \wedge \omega > 25 \\ 0 & \text{if } \xi^t_{z_v} = \textit{stop} \\ \bot & \text{otherwise} \end{cases} \qquad (8)$$

In other words, at each time step, the charge of the battery will increase by 20% *if it is charging until it is fully charged, decrease by* 25% *if it is running until it has less than* 25% *charge, and no change if it is stopped.*

The predictive model of the example above models a device state property. Let us consider a further example where the predictive model models a home state property.

Example 4. *Consider the* cleanliness *state property of a room, where the only actuator that can affect that state is a vacuum cleaning robot* z_v *(i.e.,* $\Phi_{cleanliness} = \{z_v\}$*). Assume the room is* 0% *clean at time step* t *and the action of robot* z_v *is* $\xi^t_{z_v}$ *at that time step. Thus:*

$$\Gamma^{t+1}_{cleanliness}(0, \xi^t_{z_v}) = 0 + \Delta_{cleanliness}(0, \xi^t_{z_v}) \qquad (9)$$

$$\Delta_{cleanliness}(\omega, \xi^t_{z_v}) = \begin{cases} \min(15, 100 - \omega) & \text{if } \xi^t_{z_v} = \textit{run} \\ 0 & \text{otherwise} \end{cases} \qquad (10)$$

In other words, at each time step, the cleanliness of the room will increase by 15% *if the robot is running until it is fully cleaned and no change otherwise.*

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*

[1] A complete and formal description of the grammar for the scheduling rules is provided in [9].

a schedule $\xi_{\Phi_p}^{[t_a \to t_b]}$ *of relevant actuators* Φ_p*, the predicted state trajectory* $\pi_p(\omega_p, \xi_{\Phi_p}^{[t_a \to t_b]})$ *of that state property is defined as:*

$$\pi_p(\omega_p, \xi_{\Phi_p}^{[t_a \to t_b]}) = \Gamma_p^{t_b}(\Gamma_p^{t_b-1}(\ldots(\Gamma_p^{t_a}(\omega_p, \xi_{\Phi_p}^{t_a}), \ldots), \xi_{\Phi_p}^{t_b-1}), \xi_{\Phi_p}^{t_b}) \quad (11)$$

Consider the device scheduling example in Figure 2. The predicted state trajectories of the *battery_charge* and *cleanliness* state properties following Equations (7) and (9) are shown in the second and third rows of Figure 2. 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 gray 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 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 $\xi_{\Phi_p}^{[t_a \to t_b]}$ satisfies a scheduling rule $R_p^{[t_a \to t_b]}$ (written as $\xi_{\Phi_p}^{[t_a \to t_b]} \models R_p^{[t_a \to t_b]}$) iff:

$$\forall t \in [t_a, t_b]: \quad \pi_p(\omega_p^{t_a}, \xi_{\Phi_p}^{[t_a \to t]}) \in R_p[t] \quad (12)$$

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 Figure 2, the evaluated schedule is feasible since the trajectories of both the *battery_charge* and *cleanliness* states satisfy both the *ASR* (1) and the *PSRs* (2) and (3).

3.5 Cost of Schedules

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 $\xi \in A_z$ 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 $E_i^t(\xi_{\mathbf{Z}_i}^{[1 \to H]})$ across all devices consumed by h_i at time step t under trajectory $\xi_{\mathbf{Z}_i}^{[1 \to H]}$ is:

$$E_i^t(\xi_{\mathbf{Z}_i}^{[1 \to H]}) = \ell_i^t + \sum_{z \in \mathbf{Z}_i} \rho_z(\xi_z^t) \quad (13)$$

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 ξ_z^t is the action of device z at time t in the schedule $\xi_{\mathbf{Z}_i}^{[1 \to H]}$. The cost $c_i(\xi_{\mathbf{Z}_i}^{[1 \to H]})$ associated to schedule $\xi_{\mathbf{Z}_i}^{[1 \to H]}$ in home h_i is:

$$c_i(\xi_{\mathbf{Z}_i}^{[1 \to H]}) = \sum_{t \in \mathbf{T}} \left(E_i^t(\xi_{\mathbf{Z}_i}^{[1 \to H]})\right) \cdot \theta(t) \quad (14)$$

where $\theta(t)$ is the real-time energy price per kWh at time t.

3.6 Optimization Objective

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

$$\min_{\xi_{\mathbf{Z}_i}^{[1 \to H]}} \quad \alpha_c \cdot C^{\text{sum}} + \alpha_e \cdot E^{\text{peak}} \quad (15)$$

subject to:

$$\forall h_i \in \mathbf{H}, R_p^{[t_a \to t_b]} \in \mathbf{R}_i: \quad \xi_{\Phi_p}^{[t_a \to t_b]} \models R_p^{[t_a \to t_b]} \quad (16)$$

where $\alpha_c, \alpha_e \in \mathbb{R}$ are weights,

$$C^{\text{sum}} = \sum_{h_i \in \mathbf{H}} c_i(\xi_{\mathbf{Z}_i}^{[1 \to H]})$$

is the aggregated monetary cost across all homes h_i; and

$$E^{\text{peak}} = \sum_{t \in \mathbf{T}} \sum_{\mathbf{H}_j \in \mathcal{H}} \sum_{h_i \in \mathbf{H}_j} \left(E_i^t(\xi_{\mathbf{Z}_i}^{[1 \to H]})\right)^2$$

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 each 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 [19, 17, 22, 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 (16) 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.

4. SOLUTION APPROACH

We now describe an egalitarian solution approach which relies on the distributed constraint optimization problem (DCOP) [14, 15, 27] model, where agents cooperatively seek to minimize their aggregated costs.

4.1 DCOP Model

A *distributed constraint optimization problem (DCOP)* is a tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \mathcal{A}, \alpha \rangle$, 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., D_i is the domain of x_i); $\mathcal{F} = \{f_1, \ldots, f_e\}$ is a set of *constraints* (also called *cost tables* in this work), where $f_i : \times_{x_j \in \mathbf{x}^{f_i}} D_i \to \mathbb{R}_0^+ \cup \{\infty\}$ maps each combination of value assignments of the variables $\mathbf{x}^{f_i} \subseteq \mathcal{X}$ in the *scope* of the function to a non-negative cost; $\mathcal{A} = \{a_1, \ldots, a_p\}$ is a set of *agents*; and $\alpha : \mathcal{X} \to \mathcal{A}$ is a function that maps each variable to one agent. We use $N_{a_i} = \{a_j \in \mathcal{A} \mid \exists f \in \mathcal{F}, a_i \in \mathbf{x}^f \wedge x_j \in \mathbf{x}^f\}$ to denote the neighbors of agent a_i.

An *assignment* σ is a value assignment to a set of variables $X_\sigma \subseteq \mathcal{X}$ that is consistent with the variables' domains. The cost function $\mathbf{F}_{\mathcal{P}}(\sigma) = \sum_{f \in \mathcal{F}, \mathbf{x}^f \subseteq X_\sigma} f(\sigma)$ is the sum of the costs of all the applicable constraints in σ. An assignment is said to be a *solution* if $X_\sigma = \mathcal{X}$ is the value assignment for *all* variables, and no constraint in $\mathcal{P}$ is *violated* (i.e., the cost of the assignment $\mathbf{F}_{\mathcal{P}}(\sigma) < \infty$). The goal in a DCOP is to find an optimal solution $\mathbf{x}^* = \text{argmin}_{\mathbf{x}} \mathbf{F}_{\mathcal{P}}(\mathbf{x})$.

DCOP Mapping of the SHDS problem: 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 interval $[0, \sum_{z\in \mathbf{Z}_i} \rho(\text{argmax}_{a\in A_z} \rho_z(a))]$, 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 one agent) whose costs correspond to the weighted summation of monetary costs, as defined in Equation (14).
  - *Local* hard constraints that enforce Constraint (16). 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 (15).

The neighbors N_{a_i} of agent a_i are defined as all the agents in the coalition $\mathcal{H}$ that contains h_i.

4.2 Distributed Algorithm

The SHDS problem requires several homes to solve a complex scheduling subproblem, which involves multiple variables, and to optimize the resulting energy profiles among the neighborhood of homes. The problem can be thought of as each agent solving a local complex subproblem, whose optimization is influenced by the energy profiles of the agent's neighbors, and by a coordination algorithm that allows agents to exchange their newly computed energy profiles. We thus adopt the multiple-variable agent (MVA) decomposition [2] to delegate the resolution of the agent's local problem to a centralized solver while managing inter-agent coordination through the message-passing procedure described next.

Motivated by the complexity of our problem and by successful applications of DCOP local search algorithms in real-world applications [7, 1, 28], we adapt a local search DCOP algorithms, called *Maximum Gain Message (MGM)* [13], to solve the SHDS problem. The resulting algorithm is called *Smart Home MGM (SH-MGM)*.

SH-MGM is a distributed algorithm that operates in synchronous cycles. Algorithm 1 illustrates its pseudocode. The algorithm first finds a feasible DCOP solution and then iteratively improves it, at each cycle, until convergence or time out. SH-MGM operates as follows:

- INITIALIZATION: Each agent a_i starts up and independently searches for a solution $\xi^{[1\to H]}_{\mathbf{Z}_i}$ to its local subproblem (i.e., a schedule for all devices that satisfies all the rules of the home) (line 1) that has the minimal cost $c_i(\xi^{[1\to H]}_{\mathbf{Z}_i})$ (line 2).
- SH-MGM CYCLE: Each agent a_i then computes the energy consumption $\vec{E}_i(\xi^{[1\to H]}_{\mathbf{Z}_i}) = \langle E^1_i(\xi^{[1\to H]}_{\mathbf{Z}_i}), \ldots, E^H_i(\xi^{[1\to H]}_{\mathbf{Z}_i})\rangle$ associated to its schedule and broadcasts it to all the other agents in its neighboring coalition through ENERGY messages (lines 3–4). Next, the agent waits to receive the energy consumption of all the other agents in its coalition(s) (lines 5–6). After receiving this information, it computes the peak energy consumption E^{peak}_i (second term in Equation (15))[2] of the coalition with its current solution (line 7), and stores the weighted cost $\alpha_c \cdot c_i(\xi^{[1\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot E^{\text{peak}}_i$ of its current solution.

  Then, within a given time limit, it tries to find a new solution $\hat{\xi}^{[1\to H]}_{\mathbf{Z}_i}$ to its local subproblem that is no worse (i.e., whose weighted cost is no larger) than its current solution (lines 8–10). In other words,

$$\alpha_c \cdot c_i(\hat{\xi}^{[1\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot \hat{E}^{\text{peak}}_i \leq \alpha_c \cdot c_i(\xi^{[1\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot E^{\text{peak}}_i$$

  where $\hat{E}^{\text{peak}}_i$ is the new difference in aggregated energy consumption that takes into account the new schedule of the agent.[3] If no time limit is imposed, it will find an optimal solution to its local subproblem. It then computes its gain G_i (improvement in cost):

$$G_i = \left(\alpha_c \cdot c_i(\xi^{[1\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot E^{\text{peak}}\right) - \left(\alpha_c \cdot c_i(\hat{\xi}^{[1\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot \hat{E}^{\text{peak}}\right) \quad (17)$$

Algorithm 1: SH-MGM

```
   /* Initialization */
1  $\xi^{[1\to H]}_{\mathbf{Z}_i} \leftarrow$ computeLocalSchedule( )
2  $C_i \leftarrow c_i(\xi^{[1\to H]}_{\mathbf{Z}_i})$
   /* SH-MGM Cycle */
3  foreach $a_j \in N_{a_i}$ do
4      send ENERGY$_{a_i}\left(\vec{E}_i(\xi^{[1\to H]}_{\mathbf{Z}_i})\right)$ to $a_j$
5  foreach $a_j \in N_{a_i}$ do
6      receive ENERGY$_{a_j}\left(\vec{E}_j(\xi^{[1\to H]}_{\mathbf{Z}_j})\right)$ from $a_j$
7  $E^{\text{peak}}_i \leftarrow \sum_{t\in\mathbf{T}} \sum_{a_j\in N_{a_i}\cup\{a_i\}} (E^t_j(\xi^{[1\to H]}_{\mathbf{Z}_j}))^2$
8  $\hat{\xi}^{[1\to H]}_{\mathbf{Z}_i} \leftarrow$ computeLocalSchedule($C_i$, $E^{\text{peak}}_i$)
9  $\hat{C}_i \leftarrow c_i(\hat{\xi}^{[1\to H]}_{\mathbf{Z}_i})$
10 $\hat{E}^{\text{peak}}_i \leftarrow \sum_{t\in\mathbf{T}} \sum_{a_j\in N_{a_i}} (E^t_j(\xi^{[1\to H]}_{\mathbf{Z}_j}))^2 + \sum_{t\in\mathbf{T}} (E^t_i(\hat{\xi}^{[1\to H]}_{\mathbf{Z}_i}))^2$
11 $G_i \leftarrow (\alpha_c \cdot C_i + \alpha_e \cdot E^{\text{peak}}_i) - (\alpha_c \cdot \hat{C}_i + \alpha_e \cdot \hat{E}^{\text{peak}}_i)$
12 foreach $a_j \in N_{a_i}$ do
13     send GAIN$_{a_i}(G_i)$ to $a_j$
14 foreach $a_j \in N_{a_i}$ do
15     receive GAIN$_{a_j}(G_j)$ from $a_j$
16 if all received gains are 0 then terminate
17 if $G_i > \max_{j\in\{G_k | a_k\in N_{a_i}\}} G_j$ then
18     $\xi^{[1\to H]}_{\mathbf{Z}_i} \leftarrow \hat{\xi}^{[1\to H]}_{\mathbf{Z}_i}$
19     $C_i \leftarrow \hat{C}_i$
20 Repeat SH-MGM Cycle until termination condition occurs
```

[2]We use E^{peak}_i to denote the energy consumption associated to the coalition(s) containing agent a_i.
[3]It is the sum of the energy consumption of the new agent's schedule with that of the other agents' schedules received at the start of the cycle.

between its current solution and new solution (line 11), and broadcasts this gain to all other agents in its coalition(s) using a GAIN message (lines 12–13). Upon receiving the gains of all agents within its coalition(s) (lines 14–15), it checks if they are all 0 (line 16), in which case the algorithm has converged to a local optima and no agent can unilaterally improve its schedule to improve the global solution (joint schedule of all agents). Otherwise, if the agent has the largest gain, then it will change its schedule to the new schedule. If it does not have the largest gain, it keeps its old schedule (lines 17–19). Ties are broken using an order based on the agent IDs. This mechanism ensures that at most one agent will change its schedule at each time step and that the new global solution will continuously improve until convergence.

- The process repeats until convergence or a termination condition is satisfied (e.g., timeout, a maximum number of cycles reached) (line 20).

Differences with MGM: There are several key differences between SH-MGM and (vanilla) MGM, which necessitates the new extension:

- MGM assumes that *all* the constraint costs are known a priori. In contrast, the costs of our global soft constraint E^{peak} are not known a priori. Each agent is unaware of the scheduling rules of the other agents and, thus, their possible schedules. As such, it is unaware of the set of possible energy consumption profiles (i.e., the domain of $\hat{x}_j^t$) of neighboring agents and, thus, it does not know about the possible peak energy consumption a priori. The computation of this cost can only be done lazily on demand when the energy consumptions of all agents in the coalition are broadcasted.
- MGM assumes that the domains of the variables are discrete. In contrast, the domain of the auxiliary interface variables $\hat{x}_j^t$ is continuous; they are the possible aggregated energy consumed by all devices in a home at time step t.
- Finally, each MGM agent can find a value assignment for its variable with the smallest local solution cost in linear time, with respect to the size of the variable's domain. In contrast, at each SH-MGM cycle, each agent must solve an NP-hard subproblem to find a schedule that satisfies all the constraints associated with the users' scheduling rules, and that minimizes the local solution cost. Notice that the agent' subproblem size is controlled by the number of home's devices and by the horizon time-step discretization (i.e., the number of local decision variables controlled by that agent).

5. EMPIRICAL EVALUATIONS

Our empirical evaluations compare the SH-MGM algorithm against an uncoordinated greedy approach, where each agent computes a feasible schedule for its home devices without optimizing over its cost and the aggregated peak energy incurred. We ran two types of experiments: (1) Small-scale experiments on an actual distributed system of Raspberry Pis, and (2) Large-scale simulations on synthetic microgrid instances. The fist set of experiments allows us to test the feasibility of this approach on actual physical hardware, while the second set allows us to systematically evaluate the algorithm across different problem parameters.

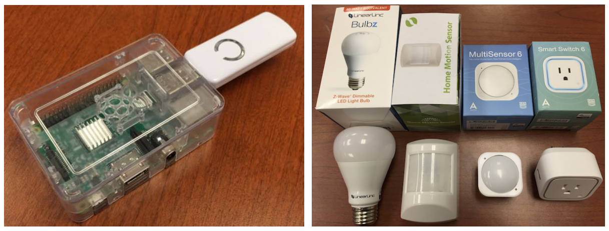

Figure 3: A Raspberry Pi with Z-Wave Dongle (left); Example of Z-wave Compatible Smart Devices (right)

We now describe the experimental setup that is common to both sets of experiments. In both experiments, each agent controls 9 smart actuators to schedule—Kenmore oven and dishwasher, GE clothes washer and dryer, iRobot vacuum cleaner, Tesla electric vehicle, LG air conditioner, Bryant heat pump, and American water heater—and 5 sensors. We selected these devices as they can be available in a typical (smart) home and they have published statistics (e.g., energy consumption profiles). The algorithms take as inputs a list of simulated smart devices to schedule as well as their associated scheduling rules and the real-time pricing scheme adopted. Each device has an associated active scheduling rule that is randomly generated for each agent and a number of passive rules that must always hold. In order to find local schedules at each SH-MGM iteration, each agent uses a Constraint Programming solver[4] as a subroutine. The effect Δ_p of each device p (see Equation (6)) on the different possible properties (e.g., how much a room can be cooled by an air-conditioner) is collated from the literature. An extended description of the smart device properties, the structural parameters (i.e., size, material, heat loss) of the homes, and the predictive models for the homes and devices state properties is reported in [9]. Finally, we set $H = 12$ and adopted a pricing scheme used by the Pacific Gas & Electric Co. for its customers in parts of California,[5] which accounts for 7 tiers ranging from \$0.198 per kWh to \$0.849 per kWh. The other parameters differ for the two experiments and are described below.

5.1 Physical MAS Experiments

In order to evaluate SH-MGM in as realistic a setting as possible, we implemented the algorithm on an actual distributed system of Raspberry Pis. A Raspberry Pi (called "PI" for short) is a bare-bones single-board computer with limited computation and storage capabilities. We used Raspberry Pi 2 Model Bs with quadcore 900MHz CPUs and 1GB of RAM. We implemented the SH-MGM algorithm using the Java Agenopment (JADE) framework,[6] which provides agent abstractions and peer-to-peer agent communication based on the asynchronous message passing paradigm. Each PI implements the logic for one agent and the agent's communication is supported through JADE, and using a wired network connected through a router.

Figure 3(left) shows an illustration of a PI with a Z-Wave dongle that can be used to issue commands to smart devices and receive information from them. Figure 3(right) shows a sample of smart devices that can be controlled. While our implementations can be used to schedule actual physical

[4]We adopt the JaCoP solver (http://www.jacop.eu/)
[5]https://goo.gl/v0eNqj/
[6]http://jade.tilab.com/

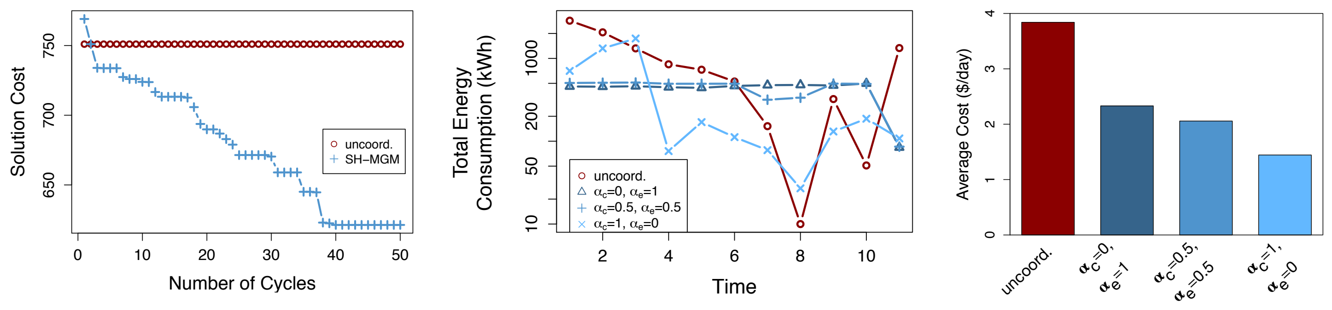

Figure 4: Physical Experimental Result with PIs (left); Synthetic Experimental Results Varying α_c and α_e (middle and right)

devices, we decided to use simulated devices as procuring a large number of commercially-available smart devices is too costly for the current project.

We set up our experiments with 7 PIs, each controlling the 9 smart actuators and 5 sensors described above. All agents belong to the same coalition. We set the equal weights α_c and α_e of the objective (see Equation (15)) to 0.5. Figure 4(left) illustrates the results of this experiment, where we imposed a 60 seconds timeout for the CP solver. As expected, the SH-MGM solution improves with increasing number of cycles, providing an economical advantage for the users as well as peak energy reduction, when compared to the uncoordinated schema. These results, thus, show the feasibility of using a local search-based schema implemented on hardware with limited storage and processing power to solve a complex problem.

5.2 Large-Scale Synthetic MAS Experiments

In our second set of experiments, 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×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. In generating our SHDS topologies, we verified that the resulting topology is connected. 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.

In these experiments, we imposed a 10 seconds timeout for the agents' CP solver. We first evaluate the effects of the weights α_c and α_e (see Equation (15)) on the quality of solutions found by the algorithms. We evaluate three configurations: $(\alpha_c = 1.0, \alpha_e = 0.0), (\alpha_c = 0.5, \alpha_e = 0.5)$, and $(\alpha_c = 0.0, \alpha_e = 1.0)$, using randomly generated microgrids with the density of Des Moines, IA. Figure 4(middle) shows the total energy consumption per hour (in kWh) of the day under each configuration. Figure 4(right) illustrates the average daily cost paid by one household under the different objective weights. The uncoordinated greedy approach achieves the worst performance, with the highest energy peak at 2852 kWh and the most expensive daily cost ($3.84). The configuration that disregards the local cost reduction ($\alpha_c = 0$) reports the best energy peak reduction (peak at 461 kWh) but highest daily cost among the coordinated configurations ($2.31). On the other extreme, the configuration that disregards the global peak reduction ($\alpha_p = 0$) reports the worst energy peak reduction (peak at 1738 kWh) but the lowest daily cost among the coordinated configurations ($1.44). Finally, the configuration with $\alpha_c = \alpha_e = 0.5$ reports intermediate results, with the highest peak at 539 kWh and a daily cost of $2.18.

Next, we investigate the impact of the number of coalitions k as well as the size of the problem on the quality of solutions found using the configuration with $\alpha_c = \alpha_e = 0.5$. Table 1 tabulates the results, where we varied $k \in \{1, 5, 10\}$ on microgrids with the densities of the three cities described above. The first row of the sub-table associated with each city reports the results for the uncoordinated approach. The table columns describe the number of coalitions k, the solving time of the algorithm (at convergence), the average time an agent spent solving its local subproblem (at each cycle of the algorithm), the network load (total number of messages exchanged by the agents), the average daily cost per household, and the maximum energy peak evaluated on the converged solutions. We make the following observations:

- The uncoordinated algorithm is the fastest approach, while the solving time of the other SH-MGM algorithms increases as the number of coalitions decreases. The reason is the number of neighbors increases as the number of coalitions decreases. As a result, the number of agents that can simultaneously improve their schedules decreases, which increases the convergence time. Additionally, the local solving time is not affected by the number of coalitions.
- Since the uncoordinated approach employs no coordination, its agents do not communicate. For the other algorithms, the network load increases as the number of coalitions decreases. The reason is the same as above, where the number of neighbors increases with decreasing coalition size, thereby increasing the number of messages sent in each broadcast.
- In terms of average daily cost, the uncoordinated approach reports the highest costs, and the cost reduces with increasing number of coalitions. This effect is due to the fact that, within a smaller coalition, agents can optimize more their cost reduction objective, since the aggregated energy peaks are smaller and thus affect less the overall objective compared to a problem with larger coalitions.

City	k	convergence time (sec)	avg. l.s. time (sec)	network load	avg. cost (\$/day)	max peak (kWh)
Des Moines	-	7.8	0.72	0	3.84	2852
	1	1044	9.62	9.8e+5	2.18	508
	5	304	9.44	4.8e+4	1.89	579
	10	218	9.37	1.2e+4	1.71	607
Boston	-	13.9	1.59	0	3.79	6034
	1	2821	9.91	1.2e+7	2.22	985
	5	866	9.91	6.7e+5	2.05	1058
	10	527	9.89	1.8e+5	1.88	1111
San Francisco	-	26.6	4.51	0	3.81	11944
	1	4238	10.4	1.7e+8	2.36	1870
	5	940	10.4	1.6e+6	2.06	2120
	10	679	10.7	1.1e+6	2.01	2310

Table 1: Synthetic Experimental Results Varying the Number of Coalitions k

- The opposite effect is observed for the maximum peak, which increases with increasing number of coalitions. This is because with smaller coalitions there is less global coordination and, hence, less opportunity to minimize the global energy peaks.

6. DISCUSSIONS AND 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 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. Furthermore, we described a mapping of this problem to a DCOP and introduced a distributed local search algorithm to find locally optimal DCOP solutions. This algorithm is implemented on an actual distributed system of Raspberry Pis, each capable of controlling and scheduling smart devices through hardware interfaces. Our experimental results show that this approach outperforms a simple uncoordinated solution on realistic small-scale experiments as well as large-scale synthetic experiments. Thus, in this paper, we make the key first steps toward the formal modeling of the SHDS problem and deployment of distributed algorithms on physical systems.

While we study the SHDS problem in a cooperative multi-agent context, our SHDS problem formulation is robust to malevolent agents and can cope with several degrees of cooperation and malevolence: On one extreme, with one partition containing all homes, the model is completely cooperative. On the other extreme, with **H** partitions, each containing one home, agents will act selfishly optimizing exclusively their own private objective.

In the future, we plan to tackle a number of significant challenges on multiple fronts: (1) We will investigate the use of Asymmetric DCOPs [5, 6] to cope with agents' actions enduring a personal cost for a specific operation that affects other agents; (2) In addition to an *egalitarian* solution, we will study an *opportunistic* solution through a Graphical Game model [8, 24] for the SHDS problem; (3) We will explore the use of more sophisticated algorithms that exploit the structure of the network to provide stronger theoretical quality guarantees; (4) We will develop user-friendly user interfaces that will enable human users to interact with the system as well as provide scheduling rules; and (5) We will use reinforcement learning to automatically learn and predict scheduling rules to improve the convenience factor of the users. These efforts, together with a number of others, are needed for actual deployment of such systems in the future.

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.

APPENDIX

A. LIST OF KEY SYMBOLS

Table 2 summarizes the most commonly used notation in the Smart Home Device Scheduling problem description.

$\mathbf{H}$	Neighborhood of smart homes
h_i	A smart home in $\mathbf{H}$
$\mathbf{Z}_i$	Set of devices in smart home h_i
$\mathbf{A}_i$	Set of actuators in smart home h_i
$\mathbf{S}_i$	Set of sensros actuators in smart home h_i
$\mathbf{L}_i$	Set of locations in smart home h_i
$\mathbf{P}_H$	Set of *state properties* of the smart homes
$\mathbf{P}_Z$	Set of *state properties* of the smart appliances
H	Problem's planning horizon
$\mathbf{T}$	Set of time points in the problem
θ	Energy pricing function
Ω_p	Set of all possible values for state property p
ℓ_z	Home's locations potentially affected by device's z actions
A_z	Set of actions for device z
ξ_z^t	The action executed by device z at time t
ξ_X^t	Set of actions executed by all devices in X at time t
$\gamma_z^H(\xi)$	Set of home' state properties affected by action ξ of device z
$\gamma_z^Z(\xi)$	Set of device' state properties affected by action ξ of device z
$\xi_X^{[t_a \to t_b]}$	A schedule for the devices in X during time interval $[t_a, t_b]$
$R_p^{[t_a \to t_b]}$	A scheduling rule over state property p and time interval $[t_a, t_b]$
ℓ_{R_p}	Location associated to the scheduling rule R_p
Φ_p	Set of actuators whose actions can affect state property p
Γ_p	Predictive model for state property p
$\Delta_p(\cdot)$	Describes the effect of the actuators' actions on state property p
$\pi_p(\cdot)$	Predicted state trajectory for property p
$\rho_z(\xi)$	Energy consumption associated to action ξ of device z
$E_i^t(\cdot)$	Aggregated energy consumption of home h_i at time t
$c_i(\cdot)$	Energy cost associated to a schedule in home h_i
C^{sum}	Objective function 1: aggregated monetary cost
E^{peak}	Objective function 2: aggregated energy consumption

Table 2: List of Key Symbols

REFERENCES

[1] A. Farinelli, A. Rogers, and N. Jennings. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Autonomous Agents and Multi-Agent Systems*, 28(3):337–380, 2014.

[2] F. Fioretto, W. Yeoh, and E. Pontelli. Multi-variable agents decomposition for DCOPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 2480–2486, 2016.

[3] F. Fioretto, W. Yeoh, E. Pontelli, Y. Ma, and S. Ranade. A DCOP approach to the economic dispatch with demand response. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2017.

[4] I. Georgievski, V. Degeler, G. A. Pagani, T. A. Nguyen, A. Lazovik, and M. Aiello. Optimizing energy costs for offices connected to the smart grid. *IEEE Transactions on Smart Grid*, 3(4):2273–2285, 2012.

[5] T. Grinshpoun, A. Grubshtein, R. Zivan, A. Netzer, and A. Meisels. Asymmetric distributed constraint optimization problems. *Journal of Artificial Intelligence Research*, 47:613–647, 2013.

[6] A. Grubshtein, R. Zivan, T. Grinshpoun, and A. Meisels. Local search for distributed asymmetric optimization. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1015–1022, 2010.

[7] M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 181–186, 2009.

[8] M. J. Kearns, M. L. Littman, and S. P. Singh. Graphical models for game theory. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 253–260, 2001.

[9] W. Kluegel, M. A. Iqbal, F. Fioretto, W. Yeoh, and E. Pontelli. A realistic dataset for the smart home device scheduling problem for DCOPs. *CoRR*, abs/1702.06970, 2017.

[10] V. Levit, T. Grinshpoun, and A. Meisels. Boolean games for charging electric vehicles. In *Proceedings of the International Conference on Intelligent Agent Technology (IAT)*, pages 86–93, 2013.

[11] T. Logenthiran, D. Srinivasan, and T. Shun. Demand side management in smart grid using heuristic optimization. *IEEE Transactions on Smart Grid*, 3(3):1244–1252, 2012.

[12] B. Lutati, V. Levit, T. Grinshpoun, and A. Meisels. Congestion games for V2G-enabled EV charging. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1421–1427, 2014.

[13] R. Maheswaran, J. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical game-based approach. In *Proceedings of the International Conference on Parallel and Distributed Computing Systems (PDCS)*, pages 432–439, 2004.

[14] 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.

[15] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1413–1420, 2005.

[16] P. Rust, G. Picard, and F. Ramparany. Using message-passing DCOP algorithms to solve energy-efficient smart environment configuration problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 468–474, 2016.

[17] 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.

[18] P. Scott, S. Thiébaux, M. van den Briel, and P. van Hentenryck. Residential demand response under uncertainty. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, pages 645–660, 2013.

[19] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.

[20] K. C. Sou, J. Weimer, H. Sandberg, and K. H. Johansson. Scheduling smart home appliances using mixed integer linear programming. In *Proceedings of the IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 5144–5149, 2011.

[21] K. M. Tsui and S.-C. Chan. Demand response optimization for smart home scheduling under real-time pricing. *IEEE Transactions on Smart Grid*, 3(4):1812–1821, 2012.

[22] S. Ueda, A. Iwasaki, and M. Yokoo. Coalition structure generation based on distributed constraint optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 197–203, 2010.

[23] M. Van Den Briel, P. Scott, S. Thiébaux, et al. Randomized load control: A simple distributed approach for scheduling smart appliances. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[24] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 345–351, 2002.

[25] T. Voice, M. Polukarov, and N. Jennings. Coalition structure generation over graphs. *Journal of Artificial Intelligence Research*, 45:165–196, 2012.

[26] T. Voice, P. Vytelingum, S. Ramchurn, A. Rogers, and N. Jennings. Decentralised control of micro-storage in the smart grid. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1421–1427, 2011.

[27] W. Yeoh and M. Yokoo. Distributed problem solving. *AI Magazine*, 33(3):53–65, 2012.

[28] R. Zivan, S. Okamoto, and H. Peled. Explorative anytime local search for distributed constraint optimization. *Artificial Intelligence*, 212:1–26, 2014.