A Multiagent System Approach to Scheduling Devices in Smart Homes

Ferdinando Fioretto
Department of Industrial & Operations Engineering
University of Michigan
fioretto@umich.edu

William Yeoh **Enrico Pontelli**
Department of Computer Science
New Mexico State University
{wyeoh,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 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.

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 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 (Voice et al. 2011; Logenthiran, Srinivasan, and Shun 2012). The former approach requires that homeowners own storage devices, in the form of batteries or electric vehicles; these are still rare resources in the current, and near future, smart grid scenarios. 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. However, uncoordinated scheduling may be detrimental to DSM performance without reducing peak load demands (Van Den Briel et al. 2013). 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.

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; **(ii)** We propose a mapping of the SHDS problem as a *distributed constraint optimization problem*, when the agents are assumed to be cooperative; **(iii)** We show how 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.

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. (2012) proposed a system to monitor and control electrical appliances in a home with the objective of reducing the energy consumption costs. Scott et al. (2013) 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. (2011) proposed a Mixed Integer Linear Program (MILP) to address smart appliance scheduling problem in single homes using a fine granularity for the technical specification of smart appliance (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 to adopt 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 (2012), through a MILP convex relaxation for the automatic load management of appliances in a smart home. Such an approach, however, pro-

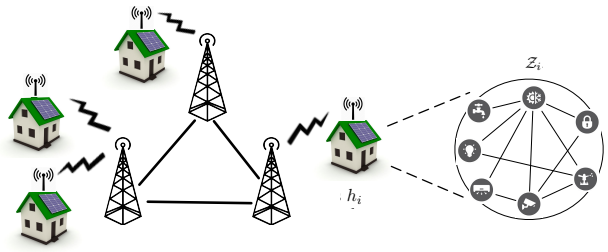

Figure 1: Illustration of a Neighborhood of Smart Homes

vides 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 (Lutati et al. 2014). Differently form such approaches, in our proposal we use a DCOP framework to solve a distributed scheduling problem for smart devices in smart homes. Similar to those approaches, we too propose to use the DCOP framework for our problem, but our underlying scheduling problem is different—we are scheduling smart devices in homes instead of electric vehicles in charging stations.

Scheduling of Devices in Smart Homes

A *Smart Home Devices 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 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 $\mathbf{T} = \{1, \ldots, H\}$ the set of time intervals.
- $\theta : \mathbf{T} \rightarrow \mathbb{R}^+$ 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.

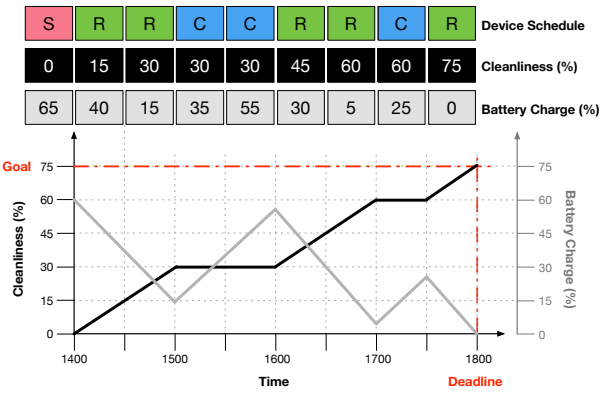

Figure 2: Smart Home Devices Scheduling Example

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, $\gamma_z^H : A_z \rightarrow 2^{\mathbf{P}_H}$ maps the actions of the device to the relevant state properties of the home, and $\gamma_z^Z : A_z \rightarrow 2^{\mathbf{P}_Z}$ 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 ℓ_{z_v} = living_room. The set of possible actions is $A_{z_v} = \{$run, charge, stop$\}$ and the mappings are:*

$\gamma_{z_v}^H$: run $\rightarrow \{cleanliness\}$; charge $\rightarrow \emptyset$; stop $\rightarrow \emptyset$

$\gamma_{z_v}^Z$: run $\rightarrow \{battery\_charge\}$; charge $\rightarrow \{battery\_charge\}$; stop $\rightarrow \emptyset$

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

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

For example, the top row of Figure 2 shows a possible schedule $\langle R, C, C, C, R, R, C, R \rangle$ 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.

The high-level SHDS goal is to find schedules for all 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*.

Scheduling Rules

We define two types of scheduling rules: **(i)** *Active scheduling rules (ASRs)* that define user-defined objectives on a desired state of the home (e.g., living room is cleaned by 1800 hrs), and **(ii)** *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 introduce a simple syntax to express scheduling rules:

$\langle location \rangle\ \langle state\ \ property \rangle\ \langle relation \rangle\ \langle state \rangle\ \langle time \rangle$[1]

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:*

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

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:*

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

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

For a home h_i, 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 (either a desired state of a home if it is an ASR or a required state of a device or a home if it is a PSR) at a location $\ell_{R_p} \in \mathbf{L}_i \cup \mathbf{Z}_i$ of a particular state property p that must hold over the time interval $[t_a, t_b] \subseteq \mathbf{T}$. Each rule is associated with a set of actuators $\Phi_p \subseteq \mathbf{A}_i$ that can be used to reach the goal state. 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. More formally,

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

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

where the former defines ASR and the latter defines 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.

Feasibility of Schedules

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

[1]The type of relations that can be captured by our current implementation are: `at` t, `before` t, `after` t, `within` $[t_1, t_2]$, and `for` t time units, with $t, t_1, t_2 \in \mathbf{T}$.

Definition 2 (Predictive Model) *A predictive model Γ_p for a state property p (of either the home or a device) is a function $\Gamma_p : \Omega_p \times \bigtimes_{z \in \Phi_p} A_z \cup \{\perp\} \to \Omega_p \cup \{\perp\}$, 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 $\xi_{\Phi_p}^t$:

$$\Gamma_p^{t+1}(\omega_p, \xi_{\Phi_p}^t) = \omega_p + \Delta_p(\omega_p, \xi_{\Phi_p}^t) \quad (6)$$

where $\Delta_p(\omega_p, \xi_{\Phi_p}^t)$ is a function describing the effect of the actuators' joint action $\xi_{\Phi_p}^t$ on state property p.

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 $\xi_{z_v}^t$ at that time step. Thus:*

$$\Gamma_{battery\_charge}^{t+1}(65, \xi_{z_v}^t) = 65 + \Delta_{battery\_charge}(65, \xi_{z_v}^t) \quad (7)$$

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

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.

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 $\xi_{z_v}^t$ at that time step. Thus:*

$$\Gamma_{cleanliness}^{t+1}(0, \xi_{z_v}^t) = 0 + \Delta_{cleanliness}(0, \xi_{z_v}^t) \quad (9)$$

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

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.

Using the predictive model, one can recursively call it to predict the trajectory of a state property p for future time steps given a schedule of actions of relevant actuators Φ_p.

Definition 3 (Predicted State Trajectory) *Given a state property p, its current state ω_p at time step t_a, and 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:*

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

One can verify if a schedule satisfies a scheduling rule by checking if its predicted state trajectories are within the

set of feasible state trajectories of that rule. Note 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:

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

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.*

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 grey line shows the states for the robot's battery charge and the black line shows the states for the cleanliness of the room. The evaluated schedule is a feasible schedule, since the trajectories of both states over time satisfy both the *active scheduling rule* (1) and the *passive scheduling rules* (2) and (3).

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; we later define an *egalitarian* solution approach to this problem.

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:

$$E_i^t(\xi_{\mathbf{Z}_i}^{[0 \to H]}) = \sum_{z \in \mathbf{Z}_i} \rho_z(\xi_z^t) \tag{13}$$

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:

$$c_i(\xi_{\mathbf{Z}_i}^{[0 \to H]}) = \sum_{t \in \mathbf{T}} \left(\ell_i^t + E_i^t(\xi_{\mathbf{Z}_i}^{[0 \to H]})\right) \cdot \theta(t) \tag{14}$$

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.

Optimization Objective

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

$$\min_{\xi_{\mathbf{Z}_i}^{[0 \to H]}} \quad \alpha_c \cdot C^{\text{sum}} + \alpha_e \cdot E^{\text{peak}} \tag{15}$$

subject to:

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

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

$$C^{\text{sum}} = \sum_{h_i \in \mathbf{H}} c_i(\xi_{\mathbf{Z}_i}^{[0 \to H]})$$

is the aggregated monetary cost across all homes h_i; and

$$E^{\text{peak}} = \sum_{t \in \mathbf{T}} \sum_{h_i \in \mathcal{H}} \left(E_i^t(\xi_{\mathbf{Z}_i}^{[0 \to H]})\right)^2$$

is a quadratic penalty function on the aggregated energy consumption across all homes h_i. 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.

Solution Approach

We now describe our solution approach based on distributed constraint optimization problems (DCOP) (Modi et al. 2005; Petcu and Faltings 2005; Yeoh and Yokoo 2012; Fioretto, Pontelli, and Yeoh 2016). This is an egalitarian model, where agents are cooperative and seek to minimize the aggregated cost.

Distributed Constraint Optimization Problems

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.

A *solution* 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. A solution is said to be *complete* if $X_\sigma = \mathcal{X}$ is the value assignment for *all* variables. The goal is to find an optimal complete solution $\mathbf{x}^* = \text{argmin}_\mathbf{x} \mathbf{F}_\mathcal{P}(\mathbf{x})$.

To describe the SHDS problem as a DCOP, it is required that agents control multiple variables (i.e., the set of actuators in each house). We thus use the *Multiple-Variable Agents (MVA)* formulation introduced in (Fioretto, Yeoh, and Pontelli 2016). 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 (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).

Distributed Algorithm

SH-MGM is a distributed algorithm that operates in synchronous cycles. 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:

- FIRST CYCLE: Each agent a_i starts up and independently searches for a solution $\xi^{[0\to H]}_{\mathbf{Z}_i}$ to its local subproblem (i.e., schedule for all devices that satisfy all the rules of the home) that has the minimal cost $c_i(\xi^{[0\to H]}_{\mathbf{Z}_i})$. It then computes the energy consumption $E^t_i(\xi^{[0\to H]}_{\mathbf{Z}_i})$ of that schedule and broadcasts it to all the other agents in the problem.
- SECOND CYCLE: Each agent waits to receive the energy consumption of all the other agents. After receiving this information, it computes the peak energy consumption E^{peak} (second term in Equation (15)) of the problem with its current solution, and stores the weighted cost $\alpha_c \cdot c_i(\xi^{[0\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot E^{\text{peak}}$ of its current solution.
  Then, within a given time limit, it tries to find a new solution $\hat{\xi}^{[0\to H]}_{\mathbf{Z}_i}$ to its local subproblem that is no worse (i.e., whose weighted cost is no smaller) than its current solution. In other words,

$$\alpha_c \cdot c_i(\hat{\xi}^{[0\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot \hat{E}^{\text{peak}} \leq \alpha_c \cdot c_i(\xi^{[0\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot E^{\text{peak}}$$

  where $\hat{E}^{\text{peak}}$ is the new difference in aggregated energy consumption that takes into account the new schedule of the agent.[2] 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):

$$G_i = \left(\alpha_c \cdot c_i(\xi^{[0\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot E^{\text{peak}}\right) - \left(\alpha_c \cdot c_i(\hat{\xi}^{[0\to H]}_{\mathbf{Z}_i}) + \alpha_e \cdot \hat{E}^{\text{peak}}\right) \quad (17)$$

  between its current solution and new solution, and broadcasts this gain to all other agents in the problem. Upon receiving the gains of all agents, it checks if they are all 0, 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. 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. Finally, all the agents send the energy consumption of their respective schedules before starting the next cycle.
- The process repeats until convergence or a termination condition is satisfied (e.g., timeout, maximum number of cycles reached).

[2]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.

Figure 3: A Raspberry Pi with Z-Wave dongle (left); Example of Z-wave compatible smart devices (right).

Empirical Evaluations

Hardware and Physical Implementation: 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 Agent Development (JADE) framework,[3] 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. The algorithm takes as inputs a list of simulated smart devices to schedule as well as their associated scheduling rules and the real-time pricing schema adopted. In order to find a feasible local schedules, the agent uses a Constraint Programming solver[4] as subroutine. Finally, the agents communication is supported through JADE, and using a wired network connected through a router.

[3]`http://jade.tilab.com/`

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

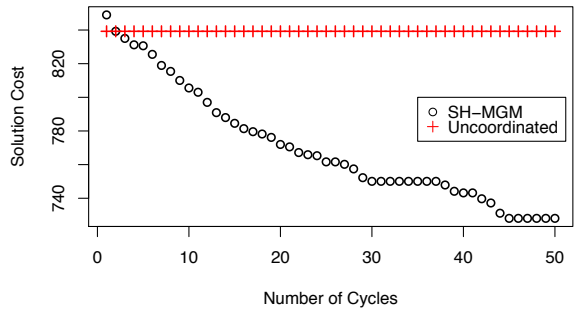

Figure 4: Average solution cost of SH-MGM vs. an uncoordinated over 100 instances.

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

Experimental Setup: We set up our experiments with 7 PIs, each controlling 9 smart actuators—Tesla electric vehicle, Kenmore oven and dishwasher, GE clothes washer and dryer, iRobot vacuum cleaner, LG air conditioner, Bryant heat pump, and American water heater—to schedule, and 5 sensors. We selected these devices as they are available in a typical home and they have published statistics (e.g., energy consumption profiles). 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. 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.

We set $H = 12$ and adopted a pricing schema 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. Finally, we choose the weights $\alpha_c = \frac{0.5}{\text{max cost}}$ and $\alpha_e = \frac{0.5}{\text{max difference}}$ (see Equation (15)) so that both the components have equal normalized weights.

Experimental Results: To evaluate the impact of SH-MGM, we compared it against a baseline, where the agents schedule their devices in an uncoordinated way: Each agent finds a schedule that minimizes its local monetary cost and disregards the aggregated peak energy incurred. Figure 4 shows the results, where we imposed a timeout of 10 seconds for the CP solver. As expected, it shows that the SH-MGM solution improves with increasing number of cycles, providing economical advantage for the uses 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. The SH-MGM finds locally optimal DCOP solutions.

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

Conclusions and Future Work

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 describes a mapping of this problem to a *distributed constraint optimization problem*; This model is suitable in problems where agents are cooperative (e.g., buildings in a campus), while the latter is suitable in a setting where the agents are self-interested (e.g., homes in a neighborhood). We introduced a distributed local search algorithm to find locally optimal solutions, when formulated as a DCOP. This algorithm is implemented on an actual physical distributed system of Raspberry Pis, each capable of controlling and scheduling smart devices through hardware interfaces. Our experimental results shows that such approach outperforms a simple uncoordinated solutions. Therefore, in this paper, we make the key first steps toward the formal modeling of the SHDS problem as well as deployment of distributed algorithms on physical systems.

In the future, we plan to tackle a number of significant challenges on multiple fronts including: (1) Investigating the use of more sophisticated algorithms that exploit the structure of the network to better propagate the problem constraints (e.g., as done in (Fioretto et al. 2014)), and conduct more comprehensive experiments; (2) Developing user-friendly user interfaces that will enable human users to interact with the system as well as provide scheduling rules; (3) Using machine learning techniques to automatically learn and predict such scheduling rules to improve the convenience factor of the users; and (4) Extending the SHDS model to allow conditional scheduling rules and to take into account uncertainty from background loads. 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 the National Science Foundation under grants 1345232 and 1550662. 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

Fioretto, F.; Le, T.; Yeoh, W.; Pontelli, E.; and Son, T. C. 2014. Improving DPOP with branch consistency for solving distributed constraint optimization problems. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 307–323.

Fioretto, F.; Pontelli, E.; and Yeoh, W. 2016. Distributed constraint optimization problems and applications: A survey. *CoRR* abs/1602.06347.

Fioretto, F.; Yeoh, W.; and Pontelli, E. 2016. Multi-variable agent decomposition for DCOPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

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

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

Lutati, B.; Levit, V.; Grinshpoun, T.; and Meisels, A. 2014. Congestion games for v2g-enabled ev charging. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 1421–1427.

Modi, P.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2005. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161(1–2):149–180.

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

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

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

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

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

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

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