Building THINC: User Incentivization and Meeting Rescheduling for Energy Savings

Jun-young Kwak, Debarun Kar, William Haskell, Pradeep Varakantham*, Milind Tambe
University of Southern California, Los Angeles, CA, 90089
*Singapore Management University, Singapore, 178902
{junyounk,dkar,tambe}@usc.edu, wbhaskell@gmail.com, *pradeepv@smu.edu.sg

ABSTRACT

This paper presents THINC, an agent developed for saving energy in real-world commercial buildings. While previous work has presented techniques for computing energy-efficient schedules, it fails to address two issues, centered on human users, that are essential in real-world agent deployments: (i) incentivizing users for their energy saving activities and (ii) interacting with users to reschedule key "energy-consuming" meetings in a timely fashion, while handling the uncertainty in such interactions. THINC addresses these shortcomings by providing four new major contributions. First, THINC computes fair division of credits from energy savings. For this fair division, THINC provides novel algorithmic advances for efficient computation of Shapley value. Second, THINC includes a novel robust algorithm to optimally reschedule identified key meetings addressing user interaction uncertainty. Third, THINC provides an end-to-end integration within a single agent of energy efficient scheduling, rescheduling and credit allocation. Finally, we deploy THINC in the real-world as a pilot project at one of the main libraries at the University of Southern California and present results illustrating the benefits in saving energy.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence

General Terms

Algorithms, Experimentation, Human Factors

Keywords

Innovative Applications; Energy Conservation; Fair Division; Shapley Value

1. INTRODUCTION

Researchers in the multiagent community have begun investigating energy conservation, specifically sustainable production, delivery and use of energy [18, 21, 24, 25]. In the present innovative application paper, we study the problem of scheduling meetings from user requests in a commercial building — a very significant source of energy consumption [8, 15, 18] — with the aim of conserving energy. We contribute a new agent for this problem, along with a suite of supporting algorithms.

Our agent, THINC (*agent **T**ool for **H**uman **IN**centivization and **C**ooperation*), leverages user flexibility to gain greater energy savings while maintaining comfort. Note that majority of research in energy savings has focused on residential buildings [21, 25]; and while comparatively much less has focused on THINC's domain of commercial buildings [15, 16, 18], THINC provides a significant advance over this work by addressing three major new challenges. First, we want to motivate users to participate in the energy conservation process by allocating energy credits among the participants. While the literature [1, 12] emphasizes that proper credit should be given based on users' actual contribution to the total energy savings (i.e., fair credit), previous work on energy savings in commercial buildings ignored this challenge. Indeed, computing a fair division of credit for a large number of users with individual preferences and flexibility is not trivial. Second, optimally rescheduling meetings often requires a significant amount of interaction between the system and its users. Previous work [15] often assumes users in key "energy-consuming" meetings would *always* agree to change their meeting time or location based on their algorithms' suggestions. This assumption is unrealistic, so we must model uncertainty in user responses to rescheduling requests. Lastly, physical deployment and demonstration of such a system in a real building is practically challenging, but critically needed.

THINC addresses these three challenges by continuously and interactively operating in three steps for energy efficiency. First, users request meeting slots through THINC while providing some flexibility in their desired slots. Second, THINC finds an optimal schedule and then asks some users to change their meeting time/location to save more energy. Finally, it notifies users of the amount of energy credits they have earned based on their given flexibility. User flexibility allows for energy-efficient scheduling, which leads to cost savings. To incentivize users to offer flexibility, *we want to divide a portion of the entire savings* among users in a fair way.

Our present paper makes four main contributions. First, we argue that the Shapley value [23] solution concept is appropriate for our fair division question. While the Shapley value has desirable theoretical properties, its limitation in scalability is a major hindrance to its use in practice [6, 9]. In response we provide two novel approximation algorithms for scaling up the approximate computation of the Shapley value: (i) approximation algorithms to efficiently estimate the Shapley value based on random sampling and graph partitioning; and (ii) a caching and relaxation method to speed up the computation of the characteristic function. Second, THINC employs a BM-MDP (Bounded-parameter Multi-objective Markov Decision Problem) [16] in reallocating key energy-consuming meetings; it presents two new algorithms for

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

BM-MDPs to handle the two layers of uncertainty: (i) model uncertainty regarding user interactions, i.e., uncertainty in user responses to reschedule requests and (ii) unexpected execution-time uncertainty regarding new meeting requests. Third, THINC is the first agent integrating (i) energy-efficient scheduling of user meeting requests while considering flexibility, (ii) rescheduling of key meetings for more energy savings, and (iii) fair credit allocations based on Shapley value to incentivize users for their energy saving activities (i.e., providing flexibility). Finally, we deploy THINC in the real-world as a pilot project and collect real user data from one of the main libraries at the University of Southern California.

2. BACKGROUND

THINC can be applied to buildings where a large number of meetings occur. For example, [15] discussed buildings where 100s of meetings occur per day and showed via simulations that energy-efficient scheduling could save about $18K/year in a single building. In this work, THINC is *deployed* in a similar educational building as a pilot project. It will later be deployed at other buildings, where hundreds of meetings are scheduled everyday. Figure 1(a) shows our testbed building, one of the main libraries at the University of Southern California (USC). Each study room has different characteristics. The study rooms operate 24 hours a day and 7 days a week. The temperature is regulated according to two set ranges for occupied and unoccupied periods of the day. HVAC (Heating, Ventilating, and Air Conditioning) systems always attempt to reach the pre-set temperature regardless of the presence of people and their temperature preferences.

2.1 Meeting Scheduling

For one of THINC's components, energy-efficient scheduling, we use the algorithm from [15], where user flexibility can be leveraged to gain greater energy savings while maintaining comfort. The input to this algorithm is a set of meeting requests defined as follows. Let T represent the entire set of time slots available and L represent the set of available locations each day. A schedule request r_i is represented as the tuple: $r_i =< T_i, L_i, \delta_i, n_i >$, where $T_i \subset T$ is the set of preferred time slots for the start of the meeting, e.g., T_i=4–5pm means the meeting may start at 4pm or 5pm., $L_i \subset L$ is a set of preferred locations, δ_i is the duration of the meeting and finally, n_i is the number of attendees.

Given a set of requests, R, e.g., 100 meeting requests that arrive during a day, [15] computes an energy-cost minimizing schedule by solving the following mixed integer linear program (MILP). The MILP will attempt to schedule meetings in smaller rooms or off-peak hours or back-to-back in order to save energy.

$$\min \quad \sum_{i \in R \setminus R^U} \sum_{t \in T} \sum_{l \in L} e_{i,l,t}, \tag{1}$$

$$\text{s.t.} \quad e_{i,l,t} \geq x_{i,l,t} \cdot E_{i,l,t} - \sum_{i' \in R \setminus R^U \setminus \{i\}} x_{i',l,t-1} \cdot C, \tag{2}$$

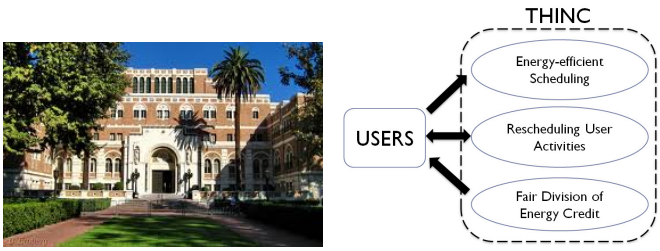
$$e_{i,l,t} \geq 0, \qquad \forall i \in R \setminus R^U, l \in L, t \in T \tag{3}$$

$$\sum_{t \in T} \sum_{l \in L} x_{i,l,t} \cdot S_{i,l,t} \geq B, \qquad \forall i \in R \setminus R^U \tag{4}$$

$$\sum_{i \in R \setminus R^U} x_{i,l,t} \leq 1, \qquad \forall l \in L, t \in T \tag{5}$$

$$\sum_{i' \in R \setminus R^U \setminus \{i\}} \sum_{t'=t}^{t+\delta_i - 1} x_{i',l,t'} \leq M(1 - x_{i,l,t}), \tag{6}$$

$$x_{i,l,t} \in \{0,1\}, \qquad \forall i \in R \setminus R^U, l \in L, t \in T \tag{7}$$



(a) Actual research testbed (library) (b) THINC

Figure 1: Research testbed & THINC

In this MILP,[1] $x_{i,l,t}$ is a binary variable that is set to 1 if meeting request r_i is scheduled in location l starting at time t and is set to zero otherwise (for $x_{i,l,t}$, $l \in L_i$ and $t \in T_i$), $e_{i,l,t}$ is a continuous variable that corresponds to the energy consumed because of scheduling meeting i in location l at time t, and $S_{i,l,t}$ is a value that indicates the satisfaction level obtained with users in meeting request r_i for scheduling the meeting in location l at time t. Constraint (2) computes the energy consumed by scheduling a meeting ($x_{i,l,t} \cdot E_{i,l,t}$), subtracting a constant amount C if meetings are scheduled back-to-back. The key take-away of this constraint is that it allows the scheduling agent to consolidate similar meetings in one room, which makes HVACs operate efficiently to maintain the desired temperature level and thus leads to energy savings. Constraint (4) is for checking if the computed schedule maintains the given comfort level B. Constraints (5) and (6) are the allocation restrictions that for each assignment slot, only one meeting can be scheduled considering the given time duration of meeting. We let $x^*(R)$ denote the optimal assignment of meeting requests R.

There is also a quantitative metric of flexibility provided in [15] that we repeat here for convenience. The flexibility of meeting request r_i is represented by a tuple $\alpha_i :< \alpha_i^T, \alpha_i^L >$. Specifically: $\alpha_i^T = \frac{|T_i|-1}{|T|-\delta_i} \times 100$ is the time flexibility of meeting i (in %) ($|T|$ is 24 hours per day). For example, given a range of T_i = 4–7pm on Monday ($|T_i| = 4$) and a meeting time duration of 2 hours, α_i^T = (4-1)/(24-2) × 100 = 13.64%. However, if T_i = 4pm only, then α_i^T = 0%, i.e., there is no time flexibility. Similarly, $\alpha_i^L = \frac{|L_i|-1}{|L|-1} \times 100$ is the location flexibility of meeting i (%) ($|L| > 1$).

2.2 Coalitional Games & Shapley Value

Cooperative game theory [17] allows players to band together and form coalitions. Formally, a cooperative game is defined by a pair (N, v), where $N = \{1, 2, \ldots, n\}$ is a set of players, and v is a characteristic function specifying the value created of different subsets (i.e., coalitions) of the players in the game. Specifically, the characteristic function, $v(S)$, associates with every subset S of N a value $v(S)$, the value of the coalition S.

In a cooperative game, we often want to encourage the grand coalition N to form. The challenge is to allocate the overall payoff $v(N)$ among the players in a fair way so that they will not deviate and form their own coalitions. Several solution concepts such as the Shapley value [23], the core [10], and the nucleolus [22] exist to guide allocation. These solution concepts all find a vector $\mathbf{x} \in \mathbb{R}^N$ that represents the allocation to each player.

The Shapley value yields a *unique* allocation $\mathbf{x}(v) = \phi(N, v)$ that is also *fair*. Specifically, the Shapley value satisfies the efficiency, symmetry, dummy player, and additivity properties which axiomatize fairness. Other concepts in cooperative game theory

[1]This MILP is an abbreviated version of an SMILP in [15].

such as the core and the nucleolus focus on yielding *stable* outcomes, but not necessarily fairness, which is of key interest in our work. Furthermore, the existence and uniqueness of the core are not guaranteed.

We use two (equivalent) definitions of Shapley value in our paper. The Shapley value is obtained by averaging the marginal contributions over all possible coalitions. Specifically, the Shapley value for player i is:

$$\phi_i(N,v) = \sum_{s=0}^{n-1} \frac{s!(n-1-s)!}{n!} \sum_{S \subseteq N \setminus \{i\}, |S|=s} (v(S \cup \{i\}) - v(S)) \quad (8)$$

where $\phi_i(N,v)$ is the savings due to $i \in N$ in the game (N,v).

An alternative definition of the Shapley value can be expressed in terms of all possible orders of the players N. Let $O : \{1,\ldots,n\} \rightarrow \{1,\ldots,n\}$ be a permutation that assigns to each position k the player $O(k)$. Let us denote by $\pi(N)$ the set of all possible permutations with player set N. Given a permutation O, let us denote by $P^i(O)$ the set of predecessors of the player i in the order O (i.e., $P^i(O) = \{O(1),...,O(k-1)\}$, if $i = O(k)$). Thus, the Shapley value can be expressed in the following way:

$$\phi_i(N,v) = \sum_{O \in \pi(N)} \frac{1}{n!}(v(P^i(O) \cup i) - v(P^i(O))), \quad i = 1,\ldots,n.$$

3. RELATED WORK

Energy Conservation in Commercial Buildings: The previous work that is closest to the research presented in this paper is [15, 16], over which THINC provides a clear-cut advance in the state of the art of *agent technology*. Within a single agent, THINC, our work presents an end-to-end integration of improvements upon the existing capabilities of previous work as well as additional new capabilities. Specifically, Kwak *et al.* [16] exclusively focus on individual meeting *rescheduling* by interacting with users under uncertainty, but not the scheduling of entire groups of meetings based on flexibility. In addition, we also provide credit allocation, a key novelty in our research contribution. We have also developed new algorithms for solving the BM-MDP model introduced in [16]. On the other hand, while we do use energy-efficient meeting scheduling algorithms from [15], we relax the unrealistic assumption that users would always agree to any change suggested to their meeting times/locations. THINC recognizes that users may not always follow its recommendations, and thus reschedules identified key meetings using BM-MDPs while considering uncertainty in user behavior. Furthermore, neither [15] nor [16] address credit allocation to users which is the major part of this paper. Thus, both [15] and [16] only attacked a *part of* the energy-efficient scheduling challenge, which is why neither has been deployed in real buildings. In contrast, integrating all of the required capabilities enabled us to deploy THINC in an educational building as a pilot project; providing a holistic, end-to-end agent in the real-world. Recently, there has been some work focusing on energy-aware scheduling in commercial buildings [18]. The authors do not consider meeting rescheduling, nor do they consider credit allocation.

Cooperative Game Theory and Fair Division in Energy Systems: Alam *et al.* [2] showed that agents can coordinate and regulate the exchange of energy between homes, which leads to energy surpluses. Each agent's contribution is computed using the Shapley value and an approximation method is used to speed up this computation. Stein *et al.* [24] introduce an online incentive compatible mechanism to schedule cooperative but self-interested agents for charging their electric vehicles while focusing on the fairness using pre-commitment in smart grid domain, which is not directly applicable in commercial buildings. Kamboj *et al.* [14] explore how a coalition of Electric Drive Vehicles (EDVs) may make more profit in electric power markets. However, this work uses proportional allocation and not Shapley value. THINC is different in that it is an integrated agent that focuses on fair credit allocations, based on novel efficient Shapley value computation while exploiting the domain properties. This is for incentivizing users to participate in this energy saving process in commercial buildings.

Shapley Value Approximation Techniques: Mann *et al.* [19] proposed a Monte-Carlo simulation technique for approximating the Shapley value and applied it to analyze the US electoral-voting system. Owen's [20] multilinear extension method for approximating the Shapley value in weighted voting games is linear in the number of players. Fatima *et al.* [9] also provided an approximation method for the Shapley value which is linear in the number of players for k-majority games. Our approximation technique is different from previous work as we exploit domain properties to integrate a novel graph partitioning algorithm, caching technique, and an LP relaxation method to approximate the Shapley value and simultaneously speed up its computation. In addition, THINC integrates this technique within an agent that (re)schedules meetings.

Robust MDP & Multi-objective Optimization Techniques: Chatterjee *et al.* [7] considered MDPs with multiple discounted reward objectives. Uncertainty in MDPs has also been considered [5, 13]. Our work is different as we simultaneously consider both multiple objectives and model uncertainty.

Human-subject Experiments with Shapley Value: To the best of our knowledge, there has been no work on perception of the Shapley value, as done in our work. However, studies with human subjects playing a weighted voting game show that Shapley value can be used as a method to predict human negotiation schemes [4].

4. THINC

THINC is made up of three specific algorithms (as shown in Figure 1(b)): (i) the scheduling algorithm described in Section 2.1, (ii) novel approximation algorithms that efficiently compute fair individual allocations based on the Shapley value, and (iii) a new robust algorithm that reschedules user meetings under uncertainty.

4.1 Fair Division of Credit

In our problem, users indicate their flexibility which determines their marginal contributions to the total energy savings. Given the energy savings, the idea is to allocate some energy credit (e.g., a significant portion of the savings) to individual users. In allocating credit, equal allocation may not be perceived as fair as shown in [1, 12] and our survey results (Section 5.1.1). Furthermore, proportional allocation based on flexibility fails in practice because the amount of flexibility does not necessarily reflect users' true contributions to energy savings. For example, out of two users A and B, let A offer 80% flexibility late at night, while B offers 40% flexibility during peak hours. Since B requests a meeting at a peak time/location (where given individual flexibility can be jointly exploited with others for more energy savings, e.g., back-to-back effect described in Section 2.1) and A at an off-peak time/location, flexibility of B may lead to more energy savings as compared to A due to the exploitation of joint flexibility. Therefore, flexibility of B has a greater effect in this case and hence B's compensation should be higher. If we used proportional allocation, A would get higher compensation which will not be perceived as fair.

Our energy-cost minimizing scheduling problem can be framed as a coalitional game, (N,v), where:

- N is a finite set of players, indexed by i. In our case, N indicates the set of meeting requests.

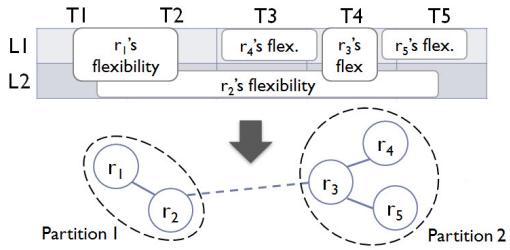


Figure 2: Illustrative example: L_i & T_i mean available rooms and time slots, respectively. Each meeting request r_i has a set of preferred locations and time, which indicates location and time flexibility.

- S is a coalition $\subseteq N$. In our case, it is a subset of meeting requests that provide flexibility.[2] So a coalition is formed from meetings that provide flexibility.
- $v : 2^N \mapsto \mathbb{R}$ is a characteristic function. In our case, $v(S)$ is the total energy-savings obtained when requests in S are flexible and requests in $N \setminus S$ are not flexible. Formally: $v(S) = \hat{e}(S) - e(S)$, where $e(S)$ is the energy consumption when meeting requests in S provide flexibility and $\hat{e}(S)$ is the energy consumption when requests in S do not provide flexibility, while requests in $N \setminus S$ are held constant as not providing flexibility (i.e., requests not providing flexibility are considered to be fixed to most preferred time/location as determined by data collected on all meetings).[3]

For our game, we appeal to the Shapley value [23] solution concept for guidance on how to fairly allocate credit. The Shapley value is computationally complex ($2^n \times 2 \times O(v)$ for each player), where $O(v)$ is the complexity of the characteristic function [23]. The computational challenge for computing the Shapley value in THINC is actually two-fold. First, computing the Shapley value for a single meeting request is challenging because we need to know the marginal contribution to all possible coalitions (Equation (8)). Second, we need to solve the MILP (Eq. (1)–(7) in Section 2.1) many times for computing the characteristic function values, and it is difficult to scale up this computation to a large number of meeting requests. For instance, as shown in Figure 2, let us assume that there are five meeting requests $r_1, r_2, \ldots, r_5$ with flexibility. Even in this small example, to compute the exact Shapley value for each meeting request, we are required to repeatedly compute $v(S)$ 64 (= $2^5 \times 2$) times in total, which is computationally expensive. Given these difficulties, we turn to approximation methods.

4.1.1 Approximate Shapley computation

We efficiently approximate the Shapley value using: (i) sampling and (ii) graph partitioning.

Sampling: Random sampling can be used to approximate the Shapley value [6, 9, 19, 20]. In particular, Castro *et al.* [6] presented the *ApproShapley* algorithm, a sampling mechanism for polynomial-time approximation of the Shapley value. In ApproShapley, the characteristic function value is repeatedly computed ($m \times 2$) times per each player, where m is the number of samples. In the above example, for each meeting request, we now only need to compute $v(S)$ 20 (= 10×2) times with 10 samples, which is smaller than the exact Shapley value computation.

Graph Partitioning: In addition to using ApproShapley, we can partition the entire meeting request set into multiple *independent* subsets, which reduces the overall computational burden. This idea is justified by the inessential axiom defined below. The entire meeting request set R can be represented as an unweighted undirected graph denoted $G = (V, E)$. As shown in Figure 2, each vertex in V represents a meeting request in R. If the flexibility ranges of any two meeting requests overlap, then those meeting requests are connected as an unordered pair in the graph defining the edge set E (with edge weight defined by the amount of overlap). For example, in Figure 2, r_2 and r_3 overlap, defining an edge between them. We can define a notion of *independence* between two meeting request subsets R_m and R_n, where $R_m, R_n \subseteq R$, as follows. Two important technical lemmas then follow:

Definition Independence: R_m and R_n are independent if $e(R_m \cup R_n) = e(R_m) + e(R_n)$, where $e(R)$ is the energy consumption of the given meeting request set R.[4]

LEMMA 4.1. *The characteristic function v for independent meetings in our coalitional game is inessential [11].*

PROOF. (Sketch) Let us assume that two meeting request subsets R_1 and R_2 ($\subseteq R$) are independent. Recall that, in our problem, $v(R)$ indicates energy savings caused by a joint flexibility in R: $v(R) = \hat{e}(R) - e(R)$. In addition, due to the independence between R_1 and R_2, $e(R_1 \cup R_2) = e(R_1) + e(R_2)$ (i.e., satisfies the inessential property both with (e) and without flexibility ($\hat{e}$)).

$$\begin{aligned} v(R_1 \cup R_2) &= \hat{e}(R_1 \cup R_2) - e(R_1 \cup R_2) \\ &= [\hat{e}(R_1) + \hat{e}(R_2)] - [e(R_1) + e(R_2)] \quad (\because e, \hat{e}\text{: inessential}) \\ &= [\hat{e}(R_1) - e(R_1)] + [\hat{e}(R_2) - e(R_2)] = v(R_1) + v(R_2). \quad \square \end{aligned}$$

LEMMA 4.2. *Assume that two meeting request subsets R_1 and R_2 ($\subseteq R$) are independent. If meeting request i is in R_1, then the Shapley value satisfies: $\phi_i(R_1 \cup R_2, v) = \phi_i(R_1, v)$.*

PROOF. (Sketch) Let $S = S_1 \cup S_2$, where $S_1 \subseteq R_1$ and $S_2 \subseteq R_2$. Since R_1 and R_2 satisfy independence, S_1 and S_2 also hold the same property. Then, the equation (8) can be rewritten as follows:

$$\phi_i(R_1 \cup R_2, v) = \sum_{s=0}^{n-1} \frac{s!(n-1-s)!}{n!} \times$$

$$\left\{ \sum_{S=S_1 \cup S_2 \subseteq R_1 \cup R_2 \setminus \{i\}, |S|=s} v(S_1 \cup S_2 \cup \{i\}) - v(S_1 \cup S_2) \right\}$$

$$= \sum_{s=0}^{n-1} \frac{s!(n-1-s)!}{n!} \times$$

$$\left\{ \sum_{S=S_1 \cup S_2 \subseteq R_1 \cup R_2 \setminus \{i\}, |S|=s} [v(S_1 \cup \{i\}) + v(S_2)] - [v(S_1) + v(S_2)] \right\}$$

($\because$ the inessential property of $v(R_1 \cup R_2)$ & $S_1 \cup \{i\} \in R_1, S_2 \in R_2$)

$$= \sum_{s=0}^{n-1} \frac{s!(n-1-s)!}{n!} \sum_{S=S_1 \subseteq R_1 \setminus \{i\}, |S|=s} (v(S_1 \cup \{i\}) - v(S_1))$$

$$= \phi_i(R_1, v). \quad \square$$

Based on these two properties, the graph G can be partitioned and the Shapley value for meetings in each partition can be computed separately — thus partitioning can speed up computation of Shapley value. Please note that only when there are non-overlapping meetings (i.e., complete independence), we can partition without loss in accuracy of Shapley value, as shown in Lemma

[2]This definition can be easily extended to the case where each separate coalition is defined based on a discretized level of flexibility.
[3]$e(S)$ is computed using the MILP in Section 2.1.
[4]Please note that we need to run the MILP to test for independence.

4.2. However, as shown in Figure 2, if there are partitions that cut across an edge, some loss in accuracy occurs; but we can minimize this loss by finding partitions that minimize the number of edges cut. This trade-off in number of partitions and accuracy will be discussed in the evaluation section.

4.1.2 Approximate characteristic value computation

In our work, the characteristic function, $v(S)$, itself is computationally intensive because it is an MILP. To compute the Shapley value, we need to solve multiple instances of these MILPs. Thus, we introduce efficient methods to approximate the characteristic value computation by relying on (i) caching and (ii) LP relaxation.

Caching: This technique exploits the following property:

Definition Exchangeability: v is exchangeable if, for every permutation π of $S \subseteq N$, $v(S) = v(\pi(S))$ [3].

$v(S)$ in our problem is exchangeable. Thus, we can further speed up the Shapley value computation by storing evaluations of $v(S)$. In this way, the characteristic function value of each coalition and all its permutations is computed only once.

LP Relaxation: It is natural then to use MILP relaxation to approximately compute $v(S)$. We specifically relax the integrality constraint (7) in (MILP) to $0 \leq x_{i,l,t} \leq 1$ for getting a linear program (LP). The optimal solution of (LP) is a lower bound on the optimal value of (MILP). We empirically show the strength of the LP relaxation for our specific MILP in the evaluation section.

4.2 THINC Rescheduling Algorithm

While THINC performs energy-efficient scheduling, it may perceive that shifting some carefully selected meetings can lead to significant energy savings. THINC then makes suggestions to involved users on how to best reschedule their meetings while ensuring a balance between energy savings and user comfort (hence, our multi-objective MDP). We cannot know the exact likelihood that users will comply with suggestions, and we may also be uncertain about the reward from energy-savings and user comfort (hence the model uncertainty). We provide new algorithms for BM-MDPs in THINC in addressing these challenges.

As a concrete example of how THINC can reschedule meetings, suppose two meeting requests (r_1 and r_2), which are originally scheduled in (10am, Room A) and (10am, Room B) respectively, are identified for rescheduling. THINC's policy may suggest r_1 and r_2 to be rescheduled to different times but the same location as r_3 (12pm, Room B). This way, the agent can consolidate all three meetings ($r_1 - r_3$) together in a smaller room B which is less expensive to heat/cool. Now, assuming that r_1 can only be scheduled either at 10am or 12pm, the best scenario is to reschedule r_1 to (10am, Room B) and r_2 to (11am, Room B) so that all three meetings only use Room B from 10am to 12pm, sequentially. Let us also assume that the r_2 is less likely to agree to reschedule, and the likelihood of r_1 and r_3 is high. In this situation, if r_2 does not comply (given low likelihood) then THINC's computed policy needs to provide an alternative action given r_2's refusal, while considering the likelihood of acceptance for this alternative. In particular, THINC instead suggests rescheduling r_3 to (11am, Room B) and r_1 to (12pm, Room B), which is highly likely to be accepted by users. In addition, if an unexpected new meeting request (r_4) arrives and is identified as an energy-consuming meeting to be rescheduled, then the rescheduling policy *may* need to change.

We thus provide two novel algorithms in this work: (i) a robust multi-objective MDP algorithm for solving BM-MDPs which allows for stochasticity in user response at planning time and (ii) replanning methods to handle execution-time uncertainty (e.g., due to the arrival of r_4). We first discuss our robust multi-objective MDP algorithm. Earlier work [16] provides "optimistic" or "pessimistic" heuristics to solve BM-MDPs, but without any performance guarantee. Instead, our present robust BM-MDP can be solved exactly by finite horizon value iteration. First, we compute the robust optimal expected value using the robust value iteration [5] for each objective. Next, given an optimal robust value for each objective, we compute the regret across all objectives. Lastly, we choose a robust policy that minimizes the regret.

STEP I: Computing the robust optimal value for each objective: We denote the (finite) state space (set of meeting requests) by S and the (finite) space of actions by A (i.e., the set of energy-efficient meeting rescheduling suggestions by THINC). We fix a finite time horizon $\mathbb{T} = \{0, 1, \ldots, T\}$, i.e., we always use a T period lookahead policy when (re)planning. Let I index a set of reward functions $r^i : S \times A \to \mathbb{R}$ to allow for multiple objectives. These include energy efficiency and comfort of different users. We let $\tau(j \mid s,\, a)$ denote the transition probabilities, the probability of transitioning to state $j \in S$ given $(s,\, a)$. For each state and action, we let $\boldsymbol{R}^i(s, a)$ denote the uncertainty set for reward r^i, and we let $\boldsymbol{\tau}(s, a)$ denote the uncertainty set for τ. The uncertainty set defines possible realizations of the uncertain parameters, e.g., uncertainty set of reward for comfort may be the interval say 1–5. For emphasis, both of these uncertainty sets depend on the current state-action pair. We let $\boldsymbol{\tau} = \{\boldsymbol{\tau}(s, a)\}_{s,\, a}$ and $\boldsymbol{R}^i = \{\boldsymbol{R}^i(s, a)\}_{s,\, a}$ be the collections of these uncertainty sets. For fixed $i \in I$, we want to maximize the worst-case reward, i.e., $\max_{a \in A} \min_{\tau \in \boldsymbol{\tau},\, r^i \in \boldsymbol{R}^i} \mathbb{E}^{\pi}_{\tau}\left[\sum_{t=0}^{T} \gamma^t r^i(s_t)\right]$, where $\mathbb{E}^{\pi}_{\tau}[\cdot]$ explicitly indicates the dependence on the transition probabilities and the policy, and s_t is the state at time t. Because the uncertainty sets only depend on the current state-action pair, we can write the Bellman equation for the above robust MDP

$$V^i_t(s) = \max_{a \in A} \min_{\tau \in \boldsymbol{\tau}(s,a),\, r^i \in \boldsymbol{R}^i(s,a)} \left\{ r^i(s) + \gamma \sum_{j \in S} \tau(j \mid s,\, a)\, V^i_{t+1}(j) \right\}$$

where V^i_t is the time t value function. The values $\{V^i_t(s)\}_{s \in S,\, t \in \mathbb{T}}$ can be computed through maximin value iteration since we have a finite time horizon [5].

STEP II: Computing the regret across all objectives: Because our problem is multi-objective, we want a policy that accounts for all objectives $i \in I$. So, we introduce a notion of regret that accounts for all objectives. We will use a vector-valued value function $\{W_t\}_{t \in \mathbb{T}} \subset \mathbb{R}^{|I|}$ where $W_t(s) = \left(W^i_t(s)\right)_{i \in I}, \forall t \in \mathbb{T}$. For a given policy π (which is different from the policy that computed V^i_t in Step I), the quantity

$$\max_{i \in I} \min_{\tau \in \boldsymbol{\tau}(s,a),\, r^i \in \boldsymbol{R}^i(s,a)} \left\{ r^i(s) + \gamma \sum_{j \in S} \tau(j \mid s,\, a)\, W^i_{t+1}(j) - V^i_t(s) \right\}$$

is the regret at time t in state $s \in S$ for action a, where V^i_t for each objective i is given as a constant from Step I. Notice that this definition takes the minimum over all reward functions. The quantity $r^i(s) + \gamma \sum_{j \in S} \tau(j \mid s,\, a)\, W^i_{t+1}(j)$ is the value for objective i, and $V^i_t(s)$ is the optimal value for objective i.

STEP III: Choosing the robust policy that minimizes the regret: In state s at time t, we choose a regret minimizing action

$$\pi^*_t(s) \in \arg\min_{a \in A} \max_{i \in I} \min_{\tau \in \boldsymbol{\tau}(s,a),\, r^i \in \boldsymbol{R}^i(s,a)} \left\{ r^i(s) + \gamma \sum_{j \in S} \tau(j \mid s,\, a)\, W^i_{t+1}(j) - V^i_t(s) \right\},$$

and then we set

$$W_t^i(s) = \min_{\tau \in \boldsymbol{\tau}(s,\pi_t^*(s)),\, r \in \boldsymbol{R}^i(s,\pi_t^*(s))} \left\{ r^i(s) + \gamma \sum_{j \in S} \tau(j \mid s,\, \pi_t^*(s))\, W_{t+1}^i(j) \right\},\ (\forall s \in S,\, \forall i \in I).$$

The optimal action is chosen in consideration of all objectives $i \in I$, and then each component of W_t is updated separately assuming the *same* optimal action is taken in each update. The resulting policy $\pi^* = (\pi_t^*)_{t \in \mathbb{T}}$ is the optimal regret minimizing policy.

During the execution of such a policy, THINC sometimes encounters unexpected situations, e.g., in the example above r_4 arrived when it was not part of the initial BM-MDP state-space and was an energy-consuming meeting in need of rescheduling. THINC's key insight is to continue to use the current BM-MDP policy to the extent possible, replanning only when new meetings are seen to potentially interfere with that policy. One alternative approach is to avoid BM-MDP planning altogether and only react to the current state. Another alternative is to stay completely committed to the original policy until completion while ignoring the new meetings. THINC rejects both of these extreme approaches and occupies a middle ground: it does use a BM-MDP policy, but when new meetings arrive, it checks if they interfere with the current policy. Specifically, the majority of incoming meeting requests propose locations and times that do not affect the current policy, allowing THINC to accrue the benefits of its optimal planning (carried out to completion) in *majority* of cases; but THINC will occasionally compute a new policy if the new meetings are seen to potentially interfere. Using real meeting arrival data in a large university building, THINC demonstrates that this "middle-ground" approach outperforms the two extreme approaches in our domain.

5. EMPIRICAL VALIDATION

We evaluate THINC in this section. For the evaluation, we built upon the simulation testbed developed in [16] by using a large data set of real meeting requests and building statistics collected from the testbed building. For experiments with meetings, we selected data from our library, where 100 meetings may arrive per day. Our experiments were run on Intel Core2 Duo 2.53GHz CPU with 8GB memory. We solved MILPs using CPLEX version 12.1. We ran all algorithms for 100 independent trials and report average values.

5.1 Shapley Value Evaluation

5.1.1 Fair Division: Why Shapley Value?

The Shapley value gives a *theoretically* fair allocation and has been previously applied in energy domains [2, 24]. However, we wished to check user reactions in our own domain, i.e., whether people believe that the Shapley value produces fair allocations of energy credits. So, we launched a survey on Amazon Mechanical Turk (AMT) and collected data for 53 unique samples. We showed survey participants two different allocations: one based on Shapley value and the other based on equal division. We then asked survey participants to rate fairness of each allocation scheme on a scale of 1 to 7 while varying information, where 7 indicates high fairness. We found that people perceive Shapley value based allocations to be more fair than those based on equal division. The average fairness rating over all users for Shapley based allocation is 5.2, as compared to 3.6 for equal division and this result is statistically significant (paired t-test; $p \leq 0.04$).

5.1.2 Approximation

Table 1: Runtime Comparison (hours) (# of meetings: 100)

# of samples \ # of partitions	5	10	20
20	0.19	0.07	0.04
50	0.49	0.17	0.11
100	0.97	0.33	0.20

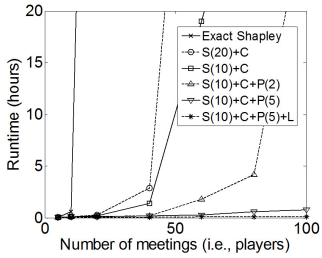


Figure 3: Runtime comparison – S: Sampling (# of samples), C: Caching, P: Partitioning (# of partitions), L: LP Relaxation

We already know that the Shapley value is computationally expensive for our setting. As shown in Figure 3 for the illustration purpose, as the number of meetings (x-axis) increases from 5 to 100, the average runtime (y-axis) of the Shapley value computation increases exponentially — in fact the computation was not completed within a reasonable amount of time. As shown in the figure, the overall runtime could be significantly improved (sped up by orders of magnitude) by combining our approximation methods.

As we provide a set of different Shapley approximation algorithms, we need to understand the contribution of different combinations of our approximation methods. In particular we need to derive settings that would allow the right tradeoff between solution quality and efficiency for our actual setting involving 100 meeting inputs per day. We thus evaluated potential speed-up by using graph partitioning on top of ApproShapley in conjunction with caching and LP relaxation. To perform graph partitioning, we relied on the *METIS* library[5], an open-source library for partitioning graphs based on the multilevel recursive-bisection and multi-constraint partitioning schemes. We tested the performance of our approximation algorithms using real meeting data while varying the number of samples and partitions. Table 1 shows the average runtime when a large number of meeting requests are given (100). Even with a large number of meeting requests, we were able to complete the overall computation in a timely fashion.

We next investigated the solution quality while keeping the same condition that was used during the runtime comparison. Figure 4 plots the average error (i.e., the average relative variance) (y-axis) against the number of partitions (5–20; x-axis) with a fixed number of samples (100) for ApproShapley.We see that as the number of partitions increases, the overall runtime decreases (Table 1) while the average error increases (Figure 4). We conclude that the combination of 100 samples and 5 partitions provides a reasonable solution (about 10% error) in a timely fashion (within 1 hour) when a large number of meeting requests arrive.

So far, we analyzed two different layers of approximations presented in our work. The question now is that how close our approximate solutions are to the true Shapley value with different combinations of these approximations. Thus, we measured the average deviation of a combination of our approximation algorithms (i.e., sampling, caching with partitioning using 20 samples and 2 partitions; $\phi_{SCP}^{20,2}$) from the exact Shapley value (ϕ_S). We conducted this experiment on 20 sampled days selecting 5 meetings per day, from real meeting data. We used a small number of meetings (5) in this test as the exact Shapley value cannot scale up beyond that.

[5] http://glaros.dtc.umn.edu/gkhome/views/metis

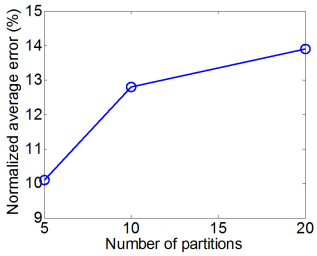


Figure 4: Solution quality

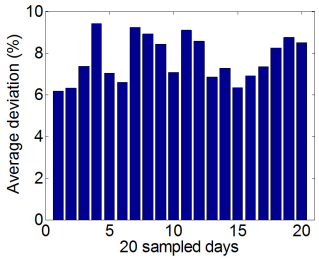


Figure 5: Average deviation (%)

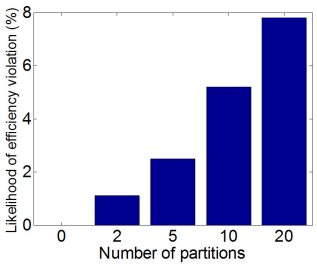


Figure 6: Efficiency violation (%)

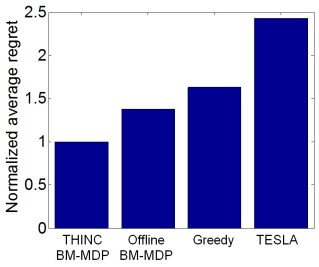


Figure 7: Solution quality

Figure 5 shows the average deviation of $\phi_{SCP}^{20,2}$ in percentage (y-axis) on 20 sampled days (x-axis). As shown in the figure, our approximation method generally followed the exact Shapley allocations, and the average deviation of ϕ_{SC}^{20} from ϕ_S was 7.73% (6.18–9.43%), which was fairly small.

It is important to verify that our approximation methods are still able to generate solutions close to theoretically fair allocations even when the problem size increases. Given the limited scalability of Shapley value, we instead focus on showing what properties out of the four that axiomatize fairness in the Shapley value are satisfied by our approximations. Our approximate allocations automatically satisfy the additivity and dummy player properties, but they do not always guarantee satisfaction of the efficiency and symmetry properties.[6] We can test empirically how often our approximation algorithms violate the efficiency and symmetry properties.

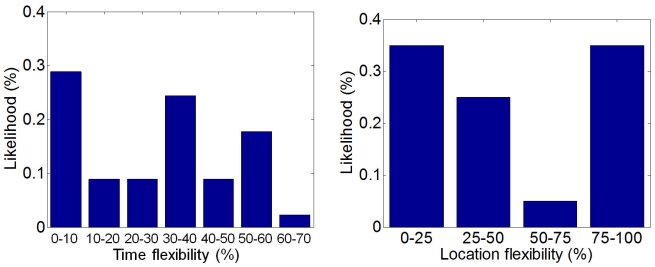
Figure 6 shows the likelihood that allocations computed from a graph partitions violate efficiency (in percentage) on the y-axis while varying the number of partitions on the x-axis. Intuitively, as the number of partitions increases, the likelihood that the efficiency property is violated also increases. However, the overall likelihood was still less than 8%. In particular, when we use 5 partitions, the likelihood was less than 3%. With respect to the symmetry, the maximum violation rate was less than 9.2% when the number of partitions varied from 0 to 20. These results show that our allocations approximately satisfy the properties that axiomatize fairness.

5.2 Performance of replanning BM-MDP

In this section, we first tested if our robust multi-objective MDP algorithm that solves BM-MDPs could generate robust well-balanced solutions (i.e., lower average regret) as compared to the standard MDP with a unified reward based on the weighted sum method and the average model from uncertainty sets, and the pessimistic heuristic for solving BM-MDPs [16]. The uniform weight distribution was applied to the weighted sum method. 50 different instances were used.[7] Each problem is based on real meeting data. On average, the MDP showed the worst result among three (2.13 times higher regret than our method) and the pessimistic heuristic achieved 1.19 times higher regret than ours, which clearly shows that our method is even more robust than the best known algorithm for solving BM-MDPs.

We then evaluated the performance of the replanning BM-MDP against three approaches while rescheduling meetings under uncertainty at both planning and execution time: (i) full-online replanning: it chooses the local best action at every time point, (ii) full-offline BM-MDP: it commits to the original policy until completion while ignoring the new meetings, and (iii) TESLA [15] that assumes users would always agree to reschedule their meetings. We compared these four approaches on 100 different instances in simulation and reported the average performance.

Figure 7 shows the normalized performance (y-axis) of each algorithm compared to the average regret achieved by THINC's MDP. As the figure shows, the offline BM-MDP achieved about 1.38 times higher regret as compared to the replanning MDP performance, and the reactive strategy achieved about 1.63 times higher regret. TESLA showed the worst result (i.e., highest average regret), and it can be arbitrarily bad as it does not consider any uncertainty while rescheduling user meetings. Our replanning BM-MDP strategy is most robust as compared to the others.



(a) Time flexibility (b) Location flexibility

Figure 8: Measured users' flexibility

Table 2: Rescheduling real meetings: uncertainty in user reactions

	% of no-response	% of rejection
First suggestion	35.0	48.0
Second suggestion	26.5	40.5
Third suggestion	20.7	39.8

5.3 Deployed Application

We deployed our integrated agent THINC as a pilot project at one of the main libraries at USC (Figure 1(a)). We wanted to test the performance of THINC in this smaller building first before deploying it at a much bigger building where there are indeed hundreds of meetings per day. 45 students used THINC during the pilot deployment. Figure 8 shows the students' reported time and location flexibility. The x-axis shows the discretized flexibility level and the corresponding frequency is reported on the y-axis. Participants reported varying levels of time and location flexibility. The average time flexibility was 27.05%, and time flexibility ranged between 0.0% and 68.18%. The average location flexibility was 42.48%, and location flexibility ranged from 0.0 to 100.0%. This shows that, *in practice*, people are willing to provide a reasonable amount of flexibility allowing significant energy savings.

[6]The formal proof is provided in the following website: `http://teamcore.usc.edu/junyounk/THINC/Supplementary.pdf`

[7]We generate different problem instances while varying the level of uncertainty (0–100%).

As part of the pilot deployment, we identified 20 key meetings for rescheduling. THINC's BM-MDP policy suggested different slots (i.e., a pair of time & location) every 6 hours. The measured uncertainty while interacting with users for rescheduling their meetings was significant (see Table 2), which emphasizes that previous work [15] cannot be applied in real situations. We achieved the average compliance rate of 45% for successfully rescheduling them with 3.6 interactions per user. This result clearly shows that BM-MDP for rescheduling identified meetings is useful rather than simply assuming users will blindly accept every suggestion.

We then divide a portion of our energy savings based on the Shapley value. To test if the users of THINC perceived our credit allocation scheme to be fair, we asked the same participants to rate fairness and their willingness to participate in energy savings on a scale of 1 to 7, where 7 denotes a high rating for fairness and willingness to participate. The average fairness rating is 5.24 and the average willingness to participate rating is 6.0. Thus we can see that users of the system perceive the Shapley based allocation scheme to be highly fair. This average fairness rating is also consistent with the result from the AMT survey, which further supports the use of Shapley value as a fair allocation method.

6. CONCLUSION

THINC advances the state-of-the-art in agent technology for optimizing energy usage in commercial buildings. There are several novelties in this paper. First, we built THINC, the first agent which integrated the following: (i) energy efficient scheduling of user meeting requests with flexibility, (ii) rescheduling of identified key meetings for further energy savings and (iii) fair credit allocation to incentivize users for their energy saving activities. Second, we provided novel approximation algorithms to efficiently compute the Shapley value, and to speed up the characteristic function computation. Third, we proposed a new robust method to optimally reschedule identified meetings while considering uncertainty. We also deployed THINC in the real-world as a pilot project and illustrated that THINC can indeed realize significant savings with user flexibility. While THINC demonstrates a concrete deployed study using concepts from cooperative game theory as a critical first step, a future implementation will need to take the next step to conduct more human subject experiments and investigate topics such as a different modeling choice.

7. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1231001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] W. Abrahmase, L. Steg, C. Vlek, and T. Rothengatter. A review of intervention studies aimed at household energy conservation. *J Environ. Psychol.*, 25:273–291, 2005.

[2] M. Alam, S. D. Ramchurn, and A. Rogers. Cooperative energy exchange for the efficient use of energy and resources in remote communities. In *AAMAS*, 2013.

[3] D. Aldous. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII*, pages 1–198, 1985.

[4] Y. Bachrach, P. Kohli, and T. Graepel. Rip-off: playing the cooperative negotiation game. In *AAMAS*, 2011.

[5] J. Bagnell, A. Y. Ng, and J. Schneider. Solving uncertain markov decision problems. *CMU-RI-TR-01-25*, 2001.

[6] J. Castro, D. Gómez, and J. Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

[7] K. Chatterjee, R. Majumdar, and T. A. Henzinger. Markov decision processes with multiple objectives. In *STACS*, 2006.

[8] M. Conferencing. Meetings in america: A study of trends, costs, and attitudes toward business travel and teleconferencing, and their impact on productivity. *A network MCI Conferencing White Paper*, 2001.

[9] S. S. Fatima, M. Wooldridge, and N. R. Jennings. A linear approximation method for the shapley value. *Artificial Intelligence*, 172(14):1673–1699, Sept. 2008.

[10] D. B. Gillies. Solutions to general non-zero-sum games. *Contributions to the Theory of Games*, 4:47–85, 1959.

[11] G. Hamiache. Associated consistency and shapley value. *International Journal of Game Theory*, 30(2):279–289, 2001.

[12] K. A. Hassett and G. E. Metcalf. Energy tax credits and residential conservation investment: Evidence from panel data. *Journal of Public Economics*, 57(2):201–217, 1995.

[13] G. N. Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

[14] S. Kamboj, W. Kempton, and K. S. Decker. Deploying power grid-integrated electric vehicles as a multi-agent system. In *AAMAS*, 2011.

[15] J. Kwak, P. Varakantham, R. Maheswaran, Y.-H. Chang, M. Tambe, B. Becerik-Gerber, and W. Wood. TESLA: an extended study of an energy-saving agent that leverages schedule flexibility. *Autonomous Agents and Multi-Agent Systems, doi=10.1007/s10458-013-9234-0*, pages 1–32, 2013.

[16] J. Kwak, P. Varakantham, R. Maheswaran, M. Tambe, F. Jazizadeh, G. Kavulya, L. Klein, B. Becerik-Gerber, T. Hayes, and W. Wood. SAVES: A sustainable multiagent application to conserve building energy considering occupants. In *AAMAS*, 2012.

[17] K. Leyton-Brown and Y. Shoham. Essentials of game theory: A concise multidisciplinary introduction. *Synthesis Lectures on AI and ML*, 2(1):1–88, 2008.

[18] A. Majumdar, D. H. Albonesi, and P. Bose. Energy-aware meeting scheduling algorithms for smart buildings. In *Buildsys*, pages 161–168. ACM, 2012.

[19] I. Mann and S. S. Llyod. Values of large games, iv: Evaluating the electoral college by monte-carlo techniques. Technical report, The Rand Corporation, 1960.

[20] G. Owen. Multilinear extensions of games. *Management Science*, 18(5):64–79, Jan. 1972.

[21] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Agent-based control for decentralised demand side management in the smart grid. In *AAMAS*, 2011.

[22] D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on applied mathematics*, 17(6):1163–1170, 1969.

[23] L. S. Shapley. A value for n-person games. *Kuhn HW, Tucker AW, editors. Contributions to the theory of games II, Annals of mathematics studies*, 28:307–317, 1953.

[24] S. Stein, E. Gerding, V. Robu, and N. Jennings. A model-based online mechanism with pre-commitment and its application to electric vehicle charging. In *AAMAS*, 2012.

[25] G. Xiong, C. Chen, S. Kishore, and A. Yener. Smart (in-home) power scheduling for demand response on the smart grid. In *ISGT*, 2011.