Boolean Games for Charging Electric Vehicles

Vadim Levit, Tal Grinshpoun, and Amnon Meisels  
Department of Computer Science  
Ben-Gurion University of the Negev  
Beer-Sheva, Israel  
Email: {levit,grinshpo,am}@cs.bgu.ac.il

Abstract—Electric Vehicles (EVs) must coordinate their charging time-slots in order to balance the load of the smart grid. The present paper models the coordination of charging among EV agents as a Boolean game, where the balancing of the global load is achieved by a principal through the use of environment variables. The iterative Boolean game of charging and avoiding overload is defined and two families of strategies for the principal are proposed. The global goal is to have no overloaded time-slots. Personal goals of agents are composed of their preferred time-slots. To achieve the global goal, the principal informs selected agents of overloaded time-slots through their environment variables. One family of strategies for the principal is shown to always converge to a solution, where the global goal of no overloaded time-slot is achieved. The other family of strategies attempts to achieve the personal goals of all the agents. An experimental evaluation of the two families of strategies on randomly generated EV charging games is presented. The proposed strategies quickly reach a solution that achieves the global goal and a large majority of the agents’ personal goals.
goal of no overloaded time-slot is achieved. The other family of strategies attempts to achieve the personal goals of all agents by also allowing messages notifying of free (unblocked) time-slots and maintaining nogoods. This family is shown to always terminate. An experimental evaluation of the proposed strategies on randomly generated EV charging games is presented. The strategies quickly reach a solution that achieves the global goal and a large majority of the agents’ personal goals.

The plan of the paper is as follows. Boolean games are presented in detail in section II. The extension of iterative Boolean games is introduced. The representation of the EV charging problem as an iterative Boolean game is given in section III. Several communication strategies for the principal are proposed in section IV. The convergence conditions of these strategies are presented and proven. An extensive empirical evaluation of the proposed communication strategies is in section V. Section VI outlines our conclusions.

II. BOOLEAN GAMES

A Boolean Game [1], [2] contains a set of agents \( A = \{A_1, \ldots, A_n\} \), the players of the game, and a set of Boolean variables \( \Phi \). Each agent \( A_i \in A \) controls a subset of the Boolean variables \( \phi_i \) (the set of variables controlled by agent \( A_i \)). Controlling variables means that the agent \( A_i \) has a unique ability within the game to set the values for each variable \( p \in \phi_i \). While it is required that no variable is controlled by more than one agent \((\phi_i \cap \phi_j = \emptyset \text{ for } i \neq j)\), the resulting subsets \( \phi_1, \ldots, \phi_n \) do not necessarily form a partition of the variables \( \Phi \). That is, there are possibly some variables that are not under the control of any of the players in the game. Let \( \varphi_E = \Phi \setminus (\phi_1, \ldots, \phi_n) \) be the variables that are not under any agent’s control; these are termed the environment variables [3].

Each agent has a personal goal, represented by a Boolean formula, \( \gamma_i \), that represents the goal of agent \( A_i \). Every goal \( \gamma_i \) contains variables of agent \( A_i \) and possibly other variables.

Let \( \mathbb{B} = \{\top, \bot\} \) be the set of Boolean truth values, with “\( \top \)” being truth and “\( \bot \)” being falsity. Following [3] these symbols will be used to denote both the constants for truth and falsity, as well as the respective truth values. A choice of agent \( A_i \), defined by a function \( \nu_i : \phi_i \rightarrow \mathbb{B} \), is an allocation of truth or falsity to all of the agent’s variables, \( \phi_i \). Let \( V_i \) denote the set of all available choices for agent \( A_i \). The intuitive interpretation of \( V_i \) is that it defines all actions or strategies available to agent \( A_i \).

An outcome \( (v_1, \ldots, v_n) \in V_1 \times \cdots \times V_n \) is a collection of choices, one for each agent. Every outcome uniquely defines a valuation for all non-environment variables in the game and one often thinks of outcomes as valuations.

The agents have beliefs about the values of environment variables. Denote by \( \beta_i : \varphi_E \rightarrow \mathbb{B} \) the beliefs of agent \( A_i \). Although these beliefs may be incorrect, they are the basis for the decisions of agents about the choices they make.

Additionally, the actions available to agents have costs defined by a cost function \( c : \Phi \times \mathbb{B} \rightarrow \mathbb{R}_{\geq 0} \), so that \( c(p, b) \) is the cost of assigning the value \( b \in \mathbb{B} \) to variable \( p \in \Phi \) [9]. Consequently, as in [3], a Boolean game, \( G \), is a tuple of size \( 3n + 4 \):

\[
G = \langle A, \Phi, \varphi_1, \ldots, \varphi_n, \gamma_1, \ldots, \gamma_n, \beta_1, \ldots, \beta_n, c, v_E \rangle
\]

where \( A = \{A_1, \ldots, A_n\} \) is the set of agents, \( \Phi = \{p, q, r, \ldots\} \) is a finite set of Boolean variables, \( \varphi_1, ..., \varphi_n \) are the variables of agents \( A \), \( \gamma_1, ..., \gamma_n \) are the goals of agents \( A \), \( \beta_1, ..., \beta_n \) are the beliefs of agents \( A \), \( c : \Phi \times \mathbb{B} \rightarrow \mathbb{R}_{\geq 0} \) is a cost function, and \( v_E : \varphi_E \rightarrow \mathbb{B} \) is the true valuation function for environment variables.

The primary aim of each agent \( A_i \) is to choose an assignment to the variables \( \varphi_i \) under its control, so as to satisfy its personal goal \( \gamma_i \). The main difficulty is that \( \gamma_i \) may contain variables not under the agent’s control (i.e., environment variables or variables controlled by other agents). If an agent can achieve its personal goal in more than one way, it will prefer to minimize costs. In case the agent cannot achieve its goal it will prefer to choose a valuation that minimizes its costs.

A. Manipulating Boolean games through communication

Following the model of [3], we assume the existence of an external principal that is able to communicate with the players of the game. More specifically, the principal may announce the true values of environment variables to the players. Although the principal is obliged to tell (announce) “the truth” and “nothing but the truth”, the model does not force the principal to tell “all the truth”. This means that while the principal’s announcements must all be truthful, the principal may announce the values of only a subset of the environment variables. The principal may announce the values of different variables to different agents. This form of announcements was termed complex in [3].

When an agent receives an announcement from the principal it updates the beliefs it has about the subset of environment variables in the announcement. Consequently, the principal may carefully choose the announcements it sends to the agents in order to modify their behavior. For instance, the principal may be able to perturb the game into reaching some sort of stabilization, e.g., having a Pure-strategy Nash Equilibrium (PNE), which is a state in which no agent has an incentive to unilaterally change its selected action. Furthermore, the principal may be able to navigate the game towards particular outcomes, so as to achieve some global goal of the system (in contrast to the personal goals of the agents).

B. Iterative Boolean games

In game theory, a repeated game refers to a finite or infinite sequence of repetitions of some base game. In the
classic form of repeated games usually the exact same game is played in each repetition. The actions of players may vary throughout the iterations due to reactions to other players’ actions in previous iterations. The present study introduces iterative Boolean games, in which the outcome of an iteration of the game is reflected in changes to the game of the successive iteration. The change could be in the payoff values of the agents, or in the case of the present study the change could be reflected in the valuations of environment variables. The changes could be automatically computed by the agents according to some known function. In the present study the changes to the environment variables are computed by an external principal, who informs the agents selectively of the new values. The iterative Boolean games proposed by the present paper resemble the games of Matsubara and Yokoo [10], in which the base game changes throughout the iterations. There, the change is in the payoff values of games of successive iterations. The motivation of Matsubara and Yokoo is to impose cooperative behavior among the agents. A very recent study of Boolean games proposes iterated Boolean games [11] and focuses on infinite series of repeated Boolean games. The goals of the games of [11] are expressed as formulae of linear temporal logic and the agents interact over an infinite series of iterations. The iterated Boolean games of [11] are very different than the iterative playing proposed in the present paper. Here, the environment is assumed to change as the iterative game continues. Consequently, the change is in the valuations of the environment variables.

In the present study the base games are Boolean games in the form presented in section II. In the single-shot version the valuations of the environment variables, \( v_E \), are fixed [3]. However, changes in the environment may have an interesting effect on the agents’ beliefs. Following the introduction of a principal (section II-A), an agent receiving an announcement knows that the received valuations (and consequently its own beliefs) are correct, because the announcements of the principal are truthful. The principal’s announcements are the sole source of information regarding the true valuations of environment variables, and therefore agents’ beliefs are only updated according to the information in the announcements. However, in case the environment changes, the correct knowledge of the valuations may become obsolete in subsequent iterations. In some scenarios, the principal may be able to apply a strategy that leads to the achievement of some global goal after several iterations of the game. In such scenarios the principal must have some knowledge regarding the expected changes in the valuations of the environment variables.

In iterative Boolean games, the objective of the principal is somewhat different than the one in the single-shot Boolean games. In the single-shot version, the principal’s goal is to choose the subset of announcements that ensures stabilization and possibly the achievement of some global goal. However, in the iterative version, the principal may think of some communication protocol in which it sends different subsets of announcements in each iteration. The principal’s objective in this case is that the iterative game will converge. A run of an iterative Boolean game is considered converged if it reaches a game, which is stable (PNE), achieves some global goal, and does not yield additional changes in the resulting game. A key element of the iterative games of the present paper is that there exists a state (some base game during the iterations) after which the valuations of environment variables never change. Such a game is presented in section III.

In his seminal work, Aumann investigates repeated games with incomplete information [12]. In these games, some or all of the players do not know the exact state of the world, and therefore are not certain about their real payoff matrices. This form of incomplete information resembles the beliefs of players regarding the valuations of environment variables. However, while the true state of the world remains constant throughout the repetitions in Aumann’s repeated games, the environment in the presently proposed iterative Boolean games does change. Thus, a different solution method is needed for iterative Boolean games of the present study.

### III. Charging Game for Electric Vehicles

A Charging Game for Electric Vehicles (CGEV) is an iterative Boolean game representation of an EV charging problem. We first describe a single iteration within the game. The agents playing the game represent EVs and their Boolean variables are the time-slots in which they are interested in charging their batteries. Agent preferences are represented by a Boolean formula. For simplicity, the formula is considered to be in disjunctive normal form (DNF), where each clause represents a combination of time-slots that the agent wants. The disjunction of clauses that forms the entire DNF formula represents the set of desired time-slot combinations by the agent. A positive literal means that the agent wants to charge at the respective time-slot. The actions of agents, which are value assignments that reflect their time-slot requests, may incur costs. Consequently, the cost of a positive literal reflects the price that the smart grid charges for this time-slot. The agent may also incorporate its own convenience into the price, which enables the agent to prioritize its preferred time-slots without removing any of the possibilities to achieve its personal goal (which is always more important than costs). A negative literal means that the agent does not need to charge at that time-slot, thus it incurs no cost.

The main constraint in CGEV, which makes this game interesting, is the limitation on the amount of energy at each time-slot, i.e., maximal load. Maximal loads may differ between time-slots, but are always finite, known, and constant throughout the iterative game. The smart grid charges
different prices according to the maximal loads of the time-slots, where time-slots with lower maximal loads are expected to be more expensive. These prices are incorporated in the costs of positive literals that are determined by the agents before the iterative game begins. The actual costs of the agents are only the costs stemming from the last iteration of the game. The principal of the game attempts to achieve the global goal, which is a load profile in which none of the time-slots is overloaded, i.e., exceeds its maximal load. For this purpose the principal uses environment variables, which are Boolean statements of load per each time-slot. A “\( \top \)” valuation of such a variable means that the respective time-slot will be overloaded according to the current agents’ requests. A “\( \bot \)” valuation means that all the agents’ requests can be fulfilled for this time-slot. The agents’ beliefs about the values of the environment variables are incorporated in their personal goals along with their preferences. This means that an agent will not ask for a time-slot that it believes is overloaded. An example of the structure of a personal goal is given below, after the following formalities.

Each iteration in a CGEV is a Boolean game, as described in section II, with the following specifics. \( T \) signifies the set of time-slots in the game. The maximal load of each time-slot \( t_j \in T \) is represented as a function \( L : T \rightarrow \mathbb{N} \). The set of Boolean variables of each agent \( A_i \in A \) is \( \varphi_i = \{x_{i,j} | t_j \in T \} \), where \( x_{i,j} \) is a Boolean variable that reflects the request of agent \( A_i \) to charge at time-slot \( t_j \). The set of environment variables is \( \varphi_E = \{e_j | t_j \in T \} \), where \( e_j \) is an environment variable that corresponds to time-slot \( t_j \). \( \beta_{i,j} \) refers to the belief of agent \( A_i \) about \( v_E(e_j) \), the true value of an environment variable \( e_j \).

As an example, consider agent \( A_i \) that wishes to charge in one of the following time-slot combinations: \((t_1,t_2)\) or \((t_1,t_3)\). The resulting personal goal is \( \gamma_i = (x_{i,1} \land \neg e_1 \land x_{i,2} \land \neg e_2) \lor (x_{i,1} \land \neg e_1 \land x_{i,3} \land \neg e_3) \). Since the agent is not necessarily aware of the true values of the environment variables, its actual personal goal is \( \gamma_i = (x_{i,1} \land \neg \beta_{i,1} \land x_{i,2} \land \neg \beta_{i,2}) \lor (x_{i,1} \land \neg \beta_{i,1} \land x_{i,3} \land \neg \beta_{i,3}) \). In our setting, an agent asks for a time-slot only when it believes the time-slot is available, thus \( x_{i,j} \) is always followed with a conjunction with \( \beta_{i,j} \).

Before the first iteration the agents assume that all the time-slots are available, and set their beliefs to “\( \bot \)” accordingly. In the following iterations, agents update their beliefs only when receiving explicit information from the principal. This process is described in detail in [3].

The iterations of the Boolean game are controlled by the principal. Algorithm 1 displays the operations of the principal after each iteration of the game.

First, the agents’ valuations (requested time-slots) are received (line 1). Following the new information, the values of environment variables, \( v_E \), must be updated (line 2):

\[
v_{E}(e_j) = \begin{cases} 
\top & \text{if } |\{A_i | x_{i,j} = \top\}| > L(t_j) \\
\bot & \text{otherwise}
\end{cases}
\]

Then, the principal decides which announcements to send to each agent according to some strategy (line 3). The different strategies are presented in the next section. Finally, the principal informs the agents (if any announcements are needed) or terminates the iterative Boolean game.

It is clear that regardless of the principal’s strategy, after every iteration all agents are in a stable state (PNE), since they chose their best response according to their beliefs. Following the definition of convergence in section II-B, in order to prove that a strategy converges one needs to show that it terminates and achieves the global goal (see Corollary 3).

IV. COMMUNICATION STRATEGIES

Two families of strategies are presented. These families differ in the types of messages that the principal sends the agent.

A. Block-only strategies

In order to achieve the global goal one can define a family of strategies that uses only “block” messages for overloaded time-slots. When an agent receives such a message for a given time-slot, it blocks the agent from choosing the corresponding time-slot in order to achieve its personal goal.

Definition A persistent strategy is one that guarantees the generation of messages whenever the global goal is not achieved.

Proposition 1: A persistent strategy that uses only “block” messages terminates in at most \(|A| \cdot |T|\) iterations.

Proof: One can define a function that counts the number of agents’ beliefs that have the value “\( \top \)”. More formally:

\[
f(G) = |\{\beta_{i,j} | \beta_{i,j} = \top\}|
\]

Since the principal makes only “block” announcements, the function \( f(G) \) is monotonically increasing. At each iteration there is at least one such message, otherwise the algorithm terminates. Moreover, \( f(G) \) is bounded by \(|A| \cdot |T|\), thus every strategy that uses only “block” messages terminates after at most \(|A| \cdot |T|\) iterations.

<table>
<thead>
<tr>
<th>Algorithm 1 The principal’s framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Valuations ( \leftarrow ) receive agents’ valuations</td>
</tr>
<tr>
<td>2: update ( v_E ); according to Valuations</td>
</tr>
<tr>
<td>3: Messages ( \leftarrow ) generate messages</td>
</tr>
<tr>
<td>4: if ( \forall A_i \in A ) Messages( (A_i) = \emptyset ) then</td>
</tr>
<tr>
<td>5: TERMINATE</td>
</tr>
<tr>
<td>6: else</td>
</tr>
<tr>
<td>7: for all ( A_i \in A ) send Messages( (A_i) ) to ( A_i )</td>
</tr>
</tbody>
</table>
Proposition 2: A persistent strategy that uses only “block” messages always achieves the global goal.

Proof: Since the strategy is persistent, it can terminate only when no overloaded time-slot exists, i.e., when the global goal is achieved. By Proposition 1, the strategy terminates in a finite number of iterations. ■

Corollary 3: A persistent strategy that uses only “block” messages always converges.

1) Strategy 1: In this strategy the principal informs exactly \(|A| - L(t_j)\) agents for every overloaded time-slot \(t_j\). Such announcements ensure no overload at time-slot \(t_j\) in the next iterations.

Algorithm 2 Strategy 1 (generate messages)

1: \(\text{ConflictTimeSlots} \leftarrow \{e_j | e_j = \top\}\)
2: \textbf{for all} \(e_j \in \text{ConflictTimeSlots} \) \textbf{do}
3: \(\text{ConfAgents} \leftarrow \{A_i | x_{i,j} = \top\}\)
4: \(\text{OkAgents} \leftarrow \text{pick } L(t_j) \text{ agents from } \text{ConfAgents}\)
5: set \(\beta_{i,j} \leftarrow \top\) for all \(A_i \in (A \setminus \text{OkAgents})\)

For every overloaded time-slot \(t_j\), the set of agents that requested this time-slot is calculated (Algorithm 2, line 3). “block” messages are generated for all agents except exactly \(L(t_j)\) agents that chose that time-slot (line 5). In the pseudo-codes of all strategies, \(\beta_{i,j}\) denotes the generation of a message for agent \(A_i\) about time-slot \(t_j\).

Proposition 4: Strategy 1 terminates in at most \(|T|\) iterations.

Proof: For each overloaded time-slot \(t_j\) the principal sends “block” messages to exactly \(|A| - L(t_j)\) agents. In following iterations this time-slot cannot be overloaded, since only “block” messages are sent. Consequently, at each iteration the principal ensures that there is no overload for at least one additional time-slot. There are \(|T|\) different time-slots, thus strategy 1 terminates in at most \(|T|\) iterations. ■

The weak point of strategy 1 is that the principal may prevent agents that actually did not choose the time-slot from ever choosing it in future iterations.

2) Strategy 2: An intuitive improvement to strategy 1 is to reduce the amount of “block” messages sent by the principal for an overloaded time-slot.

Algorithm 3 Strategy 2 (generate messages)

1: \(\text{ConflictTimeSlots} \leftarrow \{e_j | e_j = \top\}\)
2: \textbf{for all} \(e_j \in \text{ConflictTimeSlots} \) \textbf{do}
3: \(\text{ConfAgents} \leftarrow \{A_i | x_{i,j} = \top\}\)
4: set \(\beta_{i,j} \leftarrow \top\) for a single agent \(A_i \in \text{ConfAgents}\)

In this strategy (Algorithm 3), the principal informs only one of the agents that selected an overloaded time-slot (line 4). This choice enables other agents to choose this time-slot in future iterations and possibly achieve more personal goals.

The weak point of the above family of strategies is that there is no guarantee to the amount of personal goals achieved. One can create a game where these strategies will prevent all the agents from achieving their personal goals.

B. Nogood-based strategy

One way to improve the block-only strategies is to introduce a new type of messages – “unblock”. These messages enable agents to choose a formerly blocked time-slot. This operation can increase the number of personal goals achieved.

When two types of messages are used, the principal’s strategy may enter an infinite loop. To prevent the principal from entering such a state one introduces nogoods. A nogood describes the principal’s belief regarding a state in which an agent cannot achieve its personal goal. Using such nogoods the principal can try to avoid the states where the agent does not achieve its personal goal by selecting an appropriate choice of “block” messages.

Proposition 5: The principal can track the beliefs of every agent at every iteration.

Proof: In the first iteration the principal informs the agents that all environment variables are “unblocked”. As a result, the principal knows all of the agents’ beliefs at the first iteration. Following the assumption on belief updates of agents, the beliefs from any previous iterations are updated with the principal’s announcements. Thus, the principal can track every agent’s beliefs at every iteration. ■

Following Proposition 5 the principal knows every agent’s belief when the agent does not achieve its personal goal.

Proposition 6: At each iteration the principal can infer which agents can achieve their personal goal (according to their beliefs).

Proof: The primary aim of an agent is to choose a valuation that achieves its personal goal. The secondary aim of an agent is to minimize its costs. Thus, if an agent cannot achieve its personal goal it will prefer not to use any time-slot, since using a time-slot is more “expensive” than not using it. The principal can infer whether an agent \(A_i\) can achieve its personal goal according to its beliefs, by using the following equation:

\[
\bigcup_{t_j \in T} x_{i,j}
\]

If an agent does not choose to use any time-slot, the agent is unable to achieve its personal goal. ■

Following Propositions 5 and 6, one can define a nogood-based strategy. In such a strategy, every time an agent cannot achieve its personal goal, the principal will store the agent’s beliefs (only blocked time-slots) as a nogood to prevent resuming to the given state. Every time the principal wants
to inform the agent with a “block” message it must check whether the belief obtained from such an announcement is a superset of some nogood ∈ nogoods, and send it only in case it is not. For simplicity, from now on βi ∉ nogoods means that βi is not a superset of any nogood ∈ nogoods.

Algorithm 4 Nogood-based (generate messages)

1: generate “unblock” messages
2: generate “block” messages
3: for all Ai ∈ A s.t. ∪x∈T xui = ⊥ do
4: set nogoodsi ← nogoodsi ∪ {βi}

The strategy, presented in Algorithm 4, uses two types of messages – “block” messages prevent agents to use a time-slot and “unblock” messages disable the effect of “block” messages.

Algorithm 5 Nogood-based (generate “unblock” messages)

1: OkSlots ← {e|e = ⊥}
2: for all Ai ∈ A s.t. ∪x∈T xui = ⊥ do
3: set βi,j ← ⊥ for all tj ∈ OkSlots s.t. βi,j = ⊤

The principal generates “unblock” messages only for agents that do not achieve their personal goal. It finds these agents by using Equation 3 (Algorithm 5, line 2). For these agents only available time-slots can be unblocked (line 3).

Algorithm 6 Nogood-based (generate “block” messages)

1: ConfTimeSlots ← {e|e = ⊤}
2: for all ej ∈ ConfTimeSlots do
3: ConfAgents ← {Ai|xui,j = ⊤}
4: InformAgents ← pick |ConfAgents| − L(tj) agents from ConfAgents s.t. βi ∪ {βi,j = ⊤} ∉ nogoods
5: set βi,j ← ⊤ for all Ai ∈ InformAgents

The principal generates “block” messages only for overloaded time-slots (Algorithm 6, lines 1-2). The principal considers only agents that selected these time-slots (line 3). The principal verifies that agents’ beliefs, updated with the respective “block” messages, are not supersets of existing nogoods. It then generates “block” messages for a subset of these agents (lines 4-5).

In order to maintain nogoods the principal must store all sets of agent’s beliefs (only blocked time-slots) with which the agent cannot achieve its personal goal. At every iteration the principal stores the beliefs of the agents that did not achieve their personal goal (Algorithm 4, lines 3-4).

Proposition 7: The Nogood-based strategy terminates in at most |A| ⋅ (2|T| ⋅ (|T| + 1) + |T|) iterations.

Proof: Suppose the principal informs an agent by a “block” message. According to the proof of Proposition 6, such a message can be generated only when the agent’s personal goal is achieved. Thus, no nogood will be added for the agent at the current iteration. Therefore, the amount of blocked beliefs increases for the agent and the amount of nogoods remains unchanged. If the principal informs an agent by an “unblock” message, the agent is in a state where it does not achieve its personal goal. Moreover, the current agent’s belief is a nogood that is not in nogoods, or at least there were no announcements generated for this agent after this nogood was added to nogoods (Algorithm 6, line 4). Consequently, the amount of the agent’s blocked beliefs decreases but the amount of nogoods increases.

One can define a function that counts the number of nogoods and the number of blocked beliefs for a single agent:

\[ f(A_i) = |\text{nogoods}_i| ⋅ (|T| + 1) + |\{\beta_{i,j} : \beta_{i,j} = \top\}| \] (4)

where |nogoodsi| is the amount of nogoods for agent Ai and |{βi,j : βi,j = ⊤}| is the amount of blocked beliefs of the agent. If the principal informs the agent by a “block” message, then \( f(A_i) \) increases. If the principal’s message is “unblock” then \( f(A_i) \) also increases, since the amount of nogoods has a larger weight \( (|T| + 1) \). Thus, \( f(A_i) \) monotonically increases after every announcement of the principal. The function \( f(A_i) \) is bounded by \( 2^{|T|} \cdot (|T| + 1) + |T| \), since \( |\text{nogoods}_i| \) is bounded by \( 2^{|T|} \) (all possible beliefs are nogoods) and \(|\{\beta_{i,j} : \beta_{i,j} = \top\}| \) is bounded by \( |T| \) (all time-slots are blocked according to the agent’s beliefs). Consider the function:

\[ F(G) = \sum_{A_i \in A} f(A_i) \] (5)

The function in (5) monotonically increases after every announcement and is bounded by \(|A| \cdot (2^{|T|} \cdot (|T| + 1) + |T|)\). It follows that the nogood-based strategy terminates in at most \(|A| \cdot (2^{|T|} \cdot (|T| + 1) + |T|)\) iterations.

Since the nogood-based strategy does not ensure achieving the global goal upon termination, the principal must remember the best state found so far, and return to it if needed. The best state is the one with the maximal amount of achieved personal goals, where only states that achieve the global goal are considered. This resembles anytime algorithms (cf. [13]). In the extreme case where the global goal is never achieved along the way, one must resort to a block-only strategy, following the nogood-based strategy. Nevertheless, we did not encounter such a case in any of our experiments (section V).

V. EXPERIMENTAL EVALUATION

The Boolean games used in the experiments are randomly generated CGEVs, in which the number of agents, \( n \), varies from 10 to 100. In all settings there were 10 time-slots (|T| = 10). The personal goal of each player consisted of 1.10 time-slot combinations, and each combination included 1..5 time-slots; both parameters were randomly chosen for each problem instance using a uniform distribution. We used
a maximal load of $L(t_j) = \frac{|A|}{|T|}$ for each time-slot, in order to assure that most problem instances would not be satisfiable (i.e., not all personal goals could be achieved). The results are averages over 100 different problem instances that were randomly generated for each setting.

A. Experimental results

Two measures were used to compare between the strategies – the number of iterations until termination and the amount of agents that did not achieve their personal goal.

![Figure 1. Number of iterations until termination](image1)

Figure 1 clearly shows that the number of iterations strategy 1 performs is independent of the number of agents. One can predict such a behavior from Proposition 4. The number of iterations strategy 2 performs is linear in the number of agents (as proven by Proposition 1). The nogood-based strategy performs more iterations than strategies 1 and 2 until termination. The main reason for that is the need to visit more belief states in order to achieve more personal goals.

![Figure 2. Amount of unachieved personal goals](image2)

As evident by Figure 2, the nogood-based strategy is more likely to achieve a larger amount of agents whose personal goals are satisfied. This is true since maintaining nogoods helps the principal to update the beliefs of agents, thereby enabling the agents to achieve their personal goal. Strategy 2 is slightly better than strategy 1 due to the principal’s choice of “block” messages. Strategy 2 leaves more ways for agents to achieve their personal goals.

VI. CONCLUSION

The present paper models the coordination of charging among EV agents as a Boolean game, where the balancing of the global load is achieved by a principal through the use of environment variables. The global goal is to have no overloaded time-slots, in conformance with any target load distribution. Personal goals of agents are composed of their preferred time-slots.

The iterative Boolean game of charging and avoiding overload is defined and two families of strategies for the principal are proposed. In one family of strategies the principal only sends messages notifying agents about values of environment variables of overloaded time-slots. This family is shown to always converge to a solution where the global goal of no overloaded time-slot is achieved. The other family of strategies attempts to achieve the personal goals of all agents by also allowing messages notifying about values of environment variables of free time-slots. Some theoretic aspects of these strategies are proven. In an experimental evaluation the different strategies exhibit a clear tradeoff between performance (number of iterations until termination) and solution quality (amount of agents that achieve their personal goal).

The proposed model assumes a strong central position of a principal. The principal performs two main tasks. First, the principal determines which time-slots are overloaded and sets the valuations of the respective environment variables accordingly. Second, the principal decides on the information that will be disclosed to each of the agents. The strong dependence on an external principal, which is a single point of failure, seems to be a major drawback of the proposed model. Another drawback is the dichotomous nature of the environment variables – a time-slot is either overloaded or not. Such dichotomy may not be applicable to all scenarios of interest.

Future work will address the above drawbacks by devising a fully decentralized mechanism. We believe that the inclusion of costs that are calculated independently by the participating agents can resolve the shortcomings of the present partially centralized model. An additional direction for future work is to investigate the use of Boolean games for vehicle-to-grid (V2G) problems, in which the EVs sell part of the energy stored in their batteries back to the grid [14].

ACKNOWLEDGMENT

The paper is in the frame of the project: Diffusion of Mass E-Mobility and Integrating Renewable Energy in Smart Grids and Cities: Intelligent Agents for Efficient Energy Consumption, funded by the Israeli Ministry of Energy and Water. The research was partially supported by the Lynn and William Frankel Center for Computer Science at Ben-Gurion University.
REFERENCES


