Taxation Search in Boolean Games*

Vadim Levit
Dept. of Computer Science
Ben-Gurion University
Beer-Sheva, Israel
levitv@cs.bgu.ac.il

Tal Grinshpoun
Dept. of Computer Science
Ben-Gurion University
Beer-Sheva, Israel
grinshpo@cs.bgu.ac.il

Amnon Meisels
Dept. of Computer Science
Ben-Gurion University
Beer-Sheva, Israel
am@cs.bgu.ac.il

Ana L. C. Bazzan
PPGC / UFRGS
Caixa Postal 15064
91501-970, P. Alegre, Brazil
bazzan@inf.ufrgs.br

ABSTRACT

Agents in a Boolean game have a personal goal represented as a propositional logic formula over a set of Boolean variables, where some of these variables are not necessarily held by the agent. The actions available to each agent are assumed to have some cost, and the agent's secondary goal is to minimize its costs. An interesting problem is to find a taxation scheme that imposes additional costs on the agents' actions such that it incentivizes the agents to reach a stable state. The present paper first theoretically outlines the characteristics of Boolean games for which stabilization can be achieved by applying a taxation scheme. Next, a search method for an appropriate taxation scheme is proposed. The proposed method transforms the Boolean game into an Asymmetric Distributed Constraint Optimization Problem (ADCOP). ADCOPs are a natural representation of Boolean games and enable effective search by using existing algorithms. A Boolean game that represents a traffic light coordination game is used throughout the paper as a clarifying example. Finally, an experimental evaluation of the traffic light example confirms the applicability of the proposed search method and outlines some attributes of the game and the search process.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms

Keywords

Boolean games, Taxation, Search, Traffic light coordination, ADCOP

*Supported by the Lynn and William Frankel center for Computer Sciences and the Paul Ivanier center for Robotics and Production Management.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

1. INTRODUCTION

Boolean games are a family of games based on propositional logic [14, 2]. In a Boolean game each participant (agent) holds a distinct set of Boolean variables, and has some personal goal it attempts to satisfy. An agent's personal goal is represented as a propositional logic formula over some set of Boolean variables, where some of these variables are not necessarily held by the agent. The actions that an agent can take consist of assigning values to the variables it holds.

In a recently proposed generalization and more expressive version of Boolean games, termed *Cooperative Boolean Games* [7], the actions available to an agent are assumed to have some cost, and the agent's secondary goal is to minimize its costs. The present paper, following other recent studies [9, 11], refers to these cooperative Boolean games.

A common objective in game theoretic research is to reach some sort of stabilization. In Boolean games this usually means finding a *Pure-strategy Nash Equilibrium (PNE)*, which is a state in which no agent has an incentive to unilaterally change its selected action. The problem is that such a state does not necessarily exist in every Boolean game.

In games with no PNE state some of the agents may be manipulated by some *external principal* in order to achieve stabilization. Two manipulation schemes for Boolean games were recently proposed. In one scheme the Boolean games model is slightly altered to include a set of environmental variables [11]. The agents do not have direct access to these environmental variables, so they only have beliefs regarding these variables' valuations. The principal may use this disinformation in order to manipulate the agents into a stable state by selectively communicating truthful information regarding the environmental variables to some of the agents. A different type of manipulation, which is at the focus of the present paper, involves a *taxation scheme* that imposes additional costs on the actions of agents. By changing the costs of agents in this way, the taxation incentivizes the agents to achieve some objective [9, 10]. In the case that several taxation schemes are applicable, one can attempt to find the most fitting scheme according to some criterion (e.g., minimization of the overall amount of imposed tax). In both manipulation schemes the objective may be something beyond mere stabilization, such as reaching some socially desirable outcome.

Taxation may not always enable the principal to achieve its goal. The reason for this stems from the fact that taxation only affects the costs of actions for agents. The minimization of costs is only the secondary goal for every agent participating in a Boolean game, the primary goal being their personal goals (whose gain is higher than all costs in the game) [9, 10].

The first contribution of the present paper is a theoretical characterization of Boolean games for which stabilization can be achieved by applying a taxation scheme. Once such classification is achieved, the remaining task involves the actual search for a suitable taxation scheme.

It has been proven that deciding whether there is a PNE in a Boolean game is $\sum_2^p$-complete [2][1]. Searching for a taxation scheme is at least as hard, since for every potential taxation scheme one must verify whether the resulting Boolean game has a PNE. Since finding a taxation scheme or even deciding whether a PNE exists are computationally hard questions, one may approach the problem by the use of an intelligent search algorithm or heuristics that could successfully prune the search space in a Boolean game. However, a search algorithm for taxation schemes on Boolean games has not been designed yet. The second contribution of the present paper is an effective search algorithm for a taxation scheme. In order to accomplish this task, the present study proposes the transformation of the Boolean game into a distributed constraints problem, for which there are a variety of search techniques. Since Boolean games are both *distributed* between different agents and are generally *asymmetric* (in the sense that a state of the game affects differently each of the participating agents), the most natural formalization to use for this purpose is *Asymmetric Distributed Constraint Optimization Problems (ADCOP)* [12, 13].

For the purpose of clarity, the transformation into an ADCOP and the description of the search process are demonstrated through an example game, which is a Boolean game representation of a traffic light coordination game (based on [16, 6]). However, the transformation into an ADCOP can be applied to any Boolean game and is described in detail for the general case. The final contribution of the paper is an experimental evaluation of the traffic light example (on randomly generated grids), which confirms the applicability of the proposed search method and outlines some attributes of the game and of the search process.

The plan of the paper is as follows. First, Boolean games are presented in detail in Section 2. Two propositions that characterize the existence of a PNE in a Boolean game are presented. The traffic lights example of a Boolean game is described in Section 3. First, the traffic lights problem is described and its Boolean game representation, then an important proposition is proven – showing that all traffic lights problems have a PNE. Section 4 describes the search procedure for taxation. It starts by defining the problem and the framework of ADCOPs, proceeds to describe the construction procedure for an ADCOP that serves for taxation search on a given Boolean game, and then presents the required modifications to an existing ADCOP algorithm so it could handle k-ary constraints. An extensive empirical evaluation of the proposed search for taxation is in Section 5. Section 6 outlines our conclusions.

[1] $\sum_2^p = NP^{NP}$ is the class of all the languages that can be recognized in polynomial time by a nondeterministic Turing machine equipped with NP oracles [17].

2. BOOLEAN GAMES

A Boolean game [14, 2] contains a set of agents $A = \{1, ..., n\}$, the players of the game. Each agent $A_i \in A$ controls a set of Boolean variables (φ_i is 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 $p \in \varphi_i$. It is required that $\varphi_1, ..., \varphi_n$ form a partition of the game variables Φ. In other words every variable is controlled by some agent and no variable is controlled by more than one agent ($\varphi_i \cap \varphi_j = \emptyset$ for $i \neq j$ and $\bigcup_{i \in A} \varphi_i = \Phi$).

Each agent has a personal goal, represented by a Boolean formula. Thus, γ_i represents the goal of agent A_i. Every goal γ_i may contain the variables of agent A_i and possibly variables controlled by other agents.

A *choice* of agent A_i, defined by a function $v_i : \varphi_i \to \mathbb{B}$, is an allocation of truth or falsity to all of the agent's variables, φ_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* $(v_1, ..., v_n) \in V_1 \times ... \times V_n$ is a collection of choices, one for each agent. It is clear that every outcome uniquely defines a valuation for all variables in the game and we often think of outcomes as valuations.

We assume that actions available to agents have *costs* defined by a cost function $c : \Phi \times \mathbb{B} \to \mathbb{R}_{\geq}$, so that $c(p, b)$ is the cost of assigning the value $b \in \mathbb{B}$ to variable $p \in \Phi$ [7].

Consequently, as in [7, 9, 10], a Boolean Game is a $2n+3$ tuple:

$$G =< A, \Phi, c, \gamma_1, ..., \gamma_n, \varphi_1, ..., \varphi_n >$$

where $A = \{1, ..., n\}$ is the set of agents, $\Phi = \{p, q, r, ...\}$ is a finite set of Boolean variables, $c : \Phi \times \mathbb{B} \to \mathbb{R}_{\geq}$ is a cost function for available assignments, $\gamma_1, ..., \gamma_n$ are the goals of agents A, and $\varphi_1, ..., \varphi_n$ is a partition of variables Φ over agents A.

The primary aim of each agent A_i is to choose an assignment to the variables φ_i under its control, so as to satisfy its personal goal γ_i. The main difficulty is that γ_i may contain variables controlled by other agents who are also trying to choose values for their variables, so as to get their goals satisfied, and their goals may be dependent on variables controlled by agent A_i. If an agent can achieve its personal goal in more than one way, then it will prefer to minimize costs. If the agent cannot get its goal achieved it will prefer to choose a valuation that minimizes the costs.

2.1 Taxation scheme

A *taxation scheme* [9, 10] defines additional costs on actions, over those given by the cost function c. We model a taxation scheme as a function $\tau : \Phi \times \mathbb{B} \to \mathbb{R}_{\geq}$, so that $\tau(p, b)$ is the tax that should be levied on the agent controlling variable $p \in \Phi$ in case the value $b \in \mathbb{B}$ is assigned. While the cost function c is fixed for any given Boolean game G, the taxation scheme can be changed to fit our requirements. Agents always seek to minimize their costs, so by assigning different taxations we can incentivize agents to performing some actions over others.

One important assumption we make is that while taxation schemes can influence the decision making of rational agents, they cannot change the personal goal of an agent. Therefore, if an agent has a chance to achieve its goal, it will take it, no matter what the taxation incentives are.

2.2 Impact of a taxation scheme on the existence of a pure-strategy Nash equilibrium

Let us consider the following example to illustrate a Boolean game with no PNE for any taxation scheme. Throughout the paper the symbol $\overline{a}$ is used to denote the negation of a (i.e., $\neg a$). This is clearer to the eye for formulas that include the negation of longer literals.

The example game consists of two agents $A = \{1, 2\}$, each of them controlling a single variable: A_1 controls variable a ($\varphi_1 = \{a\}$) and A_2 controls variable b ($\varphi_2 = \{b\}$, so $\Phi = \{a, b\}$). The cost function will remain undefined for this example due to its irrelevance. The personal goal of agent A_1 is $\gamma_1 = (a \wedge b) \vee (\overline{a} \wedge \overline{b})$, whereas the personal goal of agent A_2 is $\gamma_2 = (a \wedge \overline{b}) \vee (\overline{a} \wedge b)$. The matrix form of the game is depicted in Figure 1:

$A_1 \backslash A_2$	$b = F$	$b = T$
$a = F$	γ_1	γ_2
$a = T$	γ_2	γ_1

Figure 1: An example of a Boolean game with no PNE for any taxation scheme

Now we can ensure that the given Boolean game has no PNE for any taxation scheme. Consider the outcome $\{a = F, b = F\}$. This pure strategy is not a Nash equilibrium because of agent A_2. Given the current outcome, the agent does not achieve its goal, but if it changes the assignment of its variable ($b = T$) it will achieve its goal. By definition of a Boolean game, the primary aim of each agent is to satisfy its personal goal (if possible), so the given pure strategy is not a Nash equilibrium. Another possible outcome is $\{a = F, b = T\}$. In this case agent A_1 does not achieve its goal, but by changing the variable it controls ($a = T$) the agent's goal will be achieved. Thus, the given strategy is also not a PNE. The remaining two outcomes are symmetric to the considered ones and are therefore also not PNEs. After considering all possible pure strategies we conclude that the given Boolean game has no PNE for any taxation scheme.

Definition 1. *A* partial valuation (PV) *for agent A_i is the valuation of all variables Φ in the Boolean game except for the variables controlled by agent A_i. The partial valuation of agent A_i is denoted by $v_{-i} = (v_1, ..., v_n) \setminus v_i$.*

Definition 2. *A* special partial valuation (SPV) *is a* partial valuation *v_{-i} for agent A_i such that there exists at least one choice v_i that combined with v_{-i} satisfies the goal γ_i, and there exists another choice v_i' that does not satisfy the goal γ_i.*

Definition 3. *A* special outcome (SO) *is an outcome that at least one agent does not achieve its personal goal and excluding that agent's choice results in a* special partial valuation.

Proposition 1. *Given a Boolean game, an outcome is not a pure-strategy Nash equilibrium state for any taxation scheme if and only if the outcome is a special outcome.*

Proof. In case the outcome is a special outcome, then by definition of SO there is at least one agent that does not achieve its personal goal and excluding that agent's choice results in a special partial valuation. Suppose WLOG that A_i is such an agent, then by the definition of SPV there exists at least one choice v_i that satisfies the agent's personal goal. The agent does not achieve its personal goal given the current outcome but by changing its choice it can achieve the goal. Following the assumption on taxation schemes the agent will achieve its personal goal if it is possible no matter what the taxation scheme is. Thus, there is at least one agent that wants to change its choice in the given outcome. Consequently, the current outcome is not a PNE state for any taxation scheme.

In case the outcome is not a special outcome, then each agent either achieves its personal goal or excluding that agent's choice does not result in a SPV. In case an agent achieves its goal then there exists a taxation scheme that minimizes the cost of its current choice. In case an agent (A_i) does not achieve its goal then there does not exist a choice v_i that helps the agent achieving its goal, since this is not a SPV. Hence, there is a taxation scheme that minimizes the cost of its current choice. Since the taxations on the actions of each agent are independent, there is a clear rule to creating a taxation scheme so that no agent wants to change its choice, which means that the outcome is a PNE state given this taxation scheme. □

Proposition 2. *A Boolean game has no pure-strategy Nash equilibrium for any taxation scheme if and only if every outcome is a special outcome.*

Proof. In case every outcome is a special outcome, then following Proposition 1, each of the outcomes is not a PNE state for any taxation scheme. Consequently, the Boolean game has no PNE for any taxation scheme.

In case there exists at least one outcome that is not a special outcome, then following Proposition 1 there exists a taxation scheme that converts the given outcome to a PNE state. Thus, the Boolean game has a PNE given this taxation scheme. □

3. TRAFFIC LIGHT COORDINATION

The increase demand for mobility in our society poses challenges that have to be addressed by the area of intelligent transportation systems. Among the efforts currently under investigation or deployment, one traditional (but nonetheless important) effort is related to optimization methods and traffic control by means of traffic signal controllers (in short, traffic lights).

3.1 Approaches for traffic light coordination

Signalized intersections are operated by traffic lights that implement the signal timing. A signal-timing plan is a unique set of timing parameters comprising the cycle length L (the length of time for the complete sequence of the phase changes), and the split (the division of the cycle length among the various movements or phases).

Traffic signals can be operated in a variety of modes. For the purpose of this paper, we are interested in the coordinated control. The goal of coordinated systems (also called synchronized or progressive systems) is to synchronize traffic signals along an arterial in order to allow platoon of vehicles, traveling at a given speed, to cross the arterial without stopping at red lights. Thus, if appropriate signal plans are selected to run at adjacent traffic signals, a "green wave" is

built. This is achieved by means of a so-called *offset* (time between the beginning of the green phase of two consecutive traffic signals) that is computed based on the desired speed and on the distance between intersections. Another important concept is the bandwidth. It is the time difference between the first and the last vehicle that can pass through without stopping.

Well designed synchronized signal plans can achieve acceptable results in one flow direction. Thus one may expect the other direction to have more delays. Although it is theoretically possible to set the synchronization for more than one flow, the bandwidth decreases in more constrained problems. For example, when the synchronization is to be set in two directions of an arterial, the bandwidth generally decreases. The difficulty is that the geometry of the arterial is fixed and with it the spacing between adjacent intersections. If one wants to have a long bandwidth in one traffic direction (e.g., bandwidth is equal to the green time), this may have consequences in other directions.

Nonetheless, synchronized or coordinated traffic lights in arterials are commonly seen, especially in cities with well behaved traffic patterns. There are several methods to compute the synchronization. Optimization of traffic lights in an off-line way is the basis of well established algorithms such as TRANSYT [18], which generates optimal coordinated plans for fixed-time operation. One drawback of this method is that plans are computed for a static situation, based on historical data. Alternatives are SCOOT, SCATS, and TUC, which are based on real-time data. However, all these approaches focus on synchronization of traffic lights in an arterial. The main difficulty to extend the synchronization to a network or to more directions of traffic is the fact that in some key intersections conflicts may appear because different directions compete for bandwidth. The conventional approach is to let a traffic expert solve these conflicts. Alternatives to such approach seek to replace the traditional arterial green wave by shorter green waves in segments of the network. This can be done, e.g., using negotiation over the question of which traffic direction shall be given more bandwidth.

An approach based on *Distributed Constraint Optimization Problems (DCOPs)* was proposed in [6]. The constraints in this problem arise from the fact that, in each node of the graph, a traffic signal cannot coordinate to establish a synchronization with neighbors located in a different direction at the same time. A conflict occurs when two neighbors want to coordinate in two different traffic directions. Junges and Bazzan [16] have later extended this scenario to bigger networks, aiming at investigating computational issues related to DCOP performance, such as time to reach an agreement and number of exchanged messages.

In the present paper, the goal of the coordination among agents is to synchronize the traffic lights in adjacent intersections in order to allow vehicles traveling at a given speed to cross the intersections without stopping at red lights. The criteria for obtaining the optimum signal timing at a single intersection is that it should lead to the minimum overall delay at the intersection. In general, the more neighbors that are synchronized, the shorter the queues.

In our setting we use a grid, in which synchronization can be achieved in either south-north and north-south or east-west and west-east directions. We assume one-way traffic to simplify the system.

3.2 Boolean game representation of the traffic light coordination

One can think of traffic light coordination as a Boolean game, in which each traffic light is an agent $a_{i,j}$ controlling a single Boolean variable $p_{i,j}$ that indicates the synchronization direction ($True$ = SN/NS and $False$ = EW/WE). The cost function $c(p_{i,j}, b)$ will be the amount of vehicles in the lane incoming to the traffic light that controls variable $p_{i,j}$ from the opposite direction of the variable's synchronization. The personal goal of an agent $a_{i,j}$ is to be synchronized with two adjacent agents in the same direction to create a *"mini-green-wave"*, i.e., $\gamma_{i,j} = (p_{i-1,j} \wedge p_{i,j} \wedge p_{i+1,j}) \vee (\overline{p_{i,j-1}} \wedge \overline{p_{i,j}} \wedge \overline{p_{i,j+1}})$. The personal goal of agents that reside on the edges of the grid are slightly simplified.

Let us consider a simple example that illustrates the general setup of the Boolean game representation.

Suppose we have a grid as depicted in Figure 2:

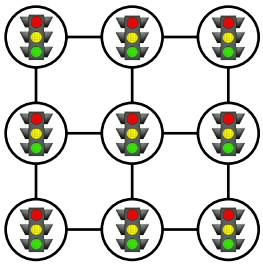

Figure 2: Network of 9 intersections

We have 9 agents $A = \{a_{1,1}, a_{1,2}, ..., a_{3,3}\}$, each controlling a single variable $\varphi_{i,j} = \{p_{i,j}\}$ (so $\Phi = \{p_{1,1}, p_{1,2}, ..., p_{3,3}\}$). The personal goals are $\gamma_{1,1} = (p_{1,1} \wedge p_{2,1}) \vee (\overline{p_{1,1}} \wedge \overline{p_{1,2}})$, $\gamma_{1,2} = (p_{1,2} \wedge p_{2,2}) \vee (\overline{p_{1,1}} \wedge \overline{p_{1,2}} \wedge \overline{p_{1,3}})$, ... , $\gamma_{2,2} = (p_{1,2} \wedge p_{2,2} \wedge p_{3,2}) \vee (\overline{p_{2,1}} \wedge \overline{p_{2,2}} \wedge \overline{p_{2,3}})$, ... Let $r_{i,j \to k,l}$ be the amount of vehicles in the lane between traffic light i, j and traffic light k, l, then $c(p_{i,j}, True) = r_{i-1,j \to i,j}$ and $c(p_{i,j}, False) = r_{i,j-1 \to i,j}$.

PROPOSITION 3. *Every Boolean game representing a traffic light coordination problem has at least one taxation scheme that ensures a pure-strategy Nash equilibrium.*

PROOF. Consider an outcome for the Boolean game constructed from the traffic light coordination problem, in which every agent assigns *True* to its variable[2]. In this particular outcome all the agents achieve their goals (by definition), so this is not a special outcome. Following Proposition 2, there is at least one taxation scheme that ensures a PNE. □

4. SEARCHING FOR TAXATION

Given a Boolean game one wants to find a taxation scheme that ensures the existence of a PNE. There may be many appropriate taxation schemes, so one can search for the optimal scheme according to some criteria. The most common taxation scheme criterion is minimizing the *overall tax* in the game [9]. The overall tax of a game is defined by the present paper in the following way:

$$T(G) = \sum_{p \in \Phi, b \in \mathbb{B}} \tau(p, b) \quad (1)$$

[2]This is one of the possible outcomes, but in the general case the Boolean game may contain more such outcomes.

One can think of this approach as minimizing the degree of intervention in the game. Alternatively, there are many other, more social, criteria for taxation schemes. These include the minimax approach, in which the maximal tax is minimized, as well as an approach that minimizes the difference in taxes [9]. Additional criteria look at the end result of the game and not only at the taxation itself. These include *egalitarian social welfare*, which looks at how well the "worst-off" agent is treated, and *horizontal equity*, in which the difference in taxes is minimized separately for each *class* of agents [5]. The relevant classes of entities of Boolean games for the above social criterion are the agents that achieve their personal goal and those that do not. For simplicity, the present study will focus on minimizing the overall tax, although any of the above mentioned criteria could be applied with some small adjustments.

Some intuition about how to find a suitable taxation scheme follows from Proposition 1. Given a Boolean game, the search process must select only those outcomes that are not special outcomes. For each of these outcomes the search process must find the appropriate taxation scheme (Proposition 1 ensures that such taxation exists) and finally select the taxation that minimizes the *overall tax* $T(G)$. It is important to note that in the case that the game already has a PNE the search process should return an "empty" taxation ($T(G) = 0$).

A problem arises when one attempts to translate the above intuition into a search process for Boolean games. To the best of the authors' knowledge there are currently no algorithms or heuristics for searching for taxation or for general purpose search in Boolean games. The search task in the present paper involves exhaustive search of the entire search space of the Boolean game. This means that for the sake of finding a taxation scheme one should first go over all possible outcomes and then find the appropriate taxation scheme for each outcome. Even performing only the first stage of the search results in the exploration of an exponential number of states with no possibility of pruning.

Several studies aim at reducing the computational effort for some Boolean games tasks. Bonzon et al. [1] exploit the dependency structure between the personal goals of the various agents to facilitate the computation of PNE, by partly decomposing a game into several sub-games that are only loosely related. In another study [3] the authors connect between Boolean games and CP-nets [4]. Sauro and Villata [19] further study the dependency structure and propose a reduction that reduces the search space when searching for coalitions. Dunne and Wooldridge [8] study cases in which Boolean games may become tractable. One such case is to search for an alternative type of equilibria, and another case refers to specific Boolean games for which finding equilibria is easier than in general. As far as we can tell, none of the above approaches can assist one in searching for a taxation scheme.

In the absence of an effective search algorithm for Boolean games, one needs to turn to an alternative formalization for which search techniques that enable pruning of the search space do exist. Boolean games are both *distributed* between different agents and are generally *asymmetric* in the sense that a state of the game may differently affect each of the participating agents. Consequently, the most natural formalization to use for this purpose is *Asymmetric Distributed Constraint Optimization Problems (ADCOP)* [12, 13]. By transforming the Boolean game into an ADCOP, one can exploit existing ADCOP search algorithms. In the following subsections a short reminder of the ADCOP model is presented, as well as the process of casting the Boolean game as an ADCOP and performing the search process.

4.1 ADCOP

An ADCOP [12, 13] is a tuple

$$< A, X, D, R >$$

where $A = \{A_1, A_2, ..., A_n\}$ is a finite set of agents. $X = \{X_1, X_2, ..., X_m\}$ is a finite set of variables. Each variable is held by a single agent (an agent may hold more than one variable). $D = \{D_1, D_2, ..., D_m\}$ is a set of domains. Each domain D_i consists of the finite set of values that can be assigned to variable X_i. R is the set of relations (constraints). Each constraint $C \in R$ is a function $C : D_{i_1} \times D_{i_2} \times ... \times D_{i_k} \rightarrow \prod_{j=1}^{k} \mathbb{R}_{\geq}$ that defines a non-negative cost for every participant in every value combination of a set of variables. The *asymmetry* of constraints in the ADCOP model stems from the potentially different costs for every participant.

An *assignment* (or a label) is a pair including a variable, and a value from that variable domain. A *partial assignment (PA)* is the set of assignments, in which each variable appears at most once. *vars*(PA) is the set of all variables that appear in PA, $vars\ (PA) = \{X_i | \exists a \in D_i \bullet (X_i, a) \in PA\}$. A constraint $C \in R$ is *applicable* to PA if $X_{i_1}, X_{i_2}, ..., X_{i_k} \in vars(PA)$. The cost of *partial assignment* PA is the sum of all applicable constraints to PA over the assignments in PA. A *full assignment* is a partial assignment that includes all the variables ($vars(PA) = X$). A *solution* is a full assignment of minimal cost.

4.2 Taxation search using ADCOP

Following the intuition presented in the beginning of the section, we describe the construction procedure of an ADCOP. Searching this ADCOP should reveal the taxation scheme that ensures the existence of a PNE and imposes the minimal overall tax. Given a Boolean game G we define the ADCOP as follows:

- The set of agents in the ADCOP is exactly the set of agents from G.
- The variables of the ADCOP are exactly the variables from G with the same variable allocation.
- Every domain D_i consists of two values (0 represents *False* and 1 refers to *True*).
- For every agent A_i construct a constraint. This constraint includes valuations of variables that appear in γ_i (in the following the variables that appear in γ_i will be denoted by $(v_{i_1}, ..., v_{i_k})$). The constraints of the ADCOP are not in the form of a table, but are computed during search from a formula that takes constant computation time. The detailed description of the constraints is below.

One wants to select only those outcomes that are not SO, so for every special outcome the cost should be larger than the maximal possible overall tax. Taxation values may of course be infinitely large, but since we are searching for a taxation

scheme with the minimal overall tax, we always refer to the minimal needed value. Thus, the maximal possible overall tax M can be computed using the following equation:

$$M = \sum_{p \in \Phi} |c(p, True) - c(p, False)| \quad (2)$$

Out of all the outcomes that are not SO, we want to find the one that minimizes the overall tax. Therefore, the cost of a constraint should be the needed (minimal) tax. For all such outcomes the tax is needed only in the case where the achievement of the personal goal is independent of the agent's choice v_i. This outcome is a PNE only in case the following equation holds for every agent A_i and every valuation $v_i' \in V_i$ of the agent:

$$\sum_{p \in \varphi_i} c(p, v_i(p)) \leq \sum_{p \in \varphi_i} c(p, v_i'(p)) \quad (3)$$

where $v_i(p)$ means the Boolean value of variable p according to the choice v_i (of agent A_i). Thus, taxation must be apllied in case the equation does not hold. In such case one can rewrite Equation 3 by adding taxation in the following manner:

$$\sum_{p \in \varphi_i} c(p, v_i(p)) \leq \sum_{p \in \varphi_i} c(p, v_i'(p)) + \tau(p, v_i'(p)) \quad (4)$$

The taxation can be found by solving a system of linear equations that is constructed following Equation 4. In the specific case of a Boolean game that represents a traffic light coordination problem, every agent A_i owns exactly one variable $p_i \in \varphi_i$. Thus, Equations 3 and 4 can be simplified to:

$$\tau(p, \overline{v_i(p)}) =$$

$$\begin{cases} c(p, v_i(p)) - c(p, \overline{v_i(p)}) & \text{if } c(p, v_i(p)) > c(p, \overline{v_i(p)}) \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where $\overline{v_i(p)}$ refers to the negation of $v_i(p)$. Consequently, the above equation describes the minimal taxation to a single variable p that ensures that the agent A_i holding variable p will not have any incentive to change p's valuation. This incentive exists whenever the current cost $c(p, v_i(p))$ is higher than the cost of the negation $c(p, \overline{v_i(p)})$. So in such cases the minimal needed tax is a value that makes both relevant costs equal, i.e., the tax equals the difference between the two costs.

Finally, the cost function is defined as follows:

$$C_{A_i}(v_{i_1}, ..., v_{i_k}) =$$

$$\begin{cases} M + \epsilon & \text{if } (v_{i_1}, ..., v_{i_k}) \text{ is } \textit{part of a SO} \\ \sum_{p \in \varphi_i, b \in \mathbb{B}} \tau(p, b) & \text{otherwise} \end{cases} \quad (6)$$

for any $\epsilon > 0$. The term *part of a SO* relates to a valuation of γ_i that satisfies Definition 3. Equation 6 ensures that no special outcomes would be selected if there is at least one outcome which is not special. Note that the constraints defined above are *asymmetric* because each constraint incurs a cost on a single agent.

The ADCOP's solution is a full assignment $(v_1, v_2, ..., v_n)$ that represents a PNE state when the appropriate taxation scheme is used. The taxation scheme is calculated during the search process and can be stored along with its matching assignment.

Note that the taxation scheme only adds costs to the negative valuations. The reason for this is that the solution is a potential PNE (unless there is no taxation scheme that achieves stabilization for this Boolean game), so the taxation scheme only needs to make sure that the negative valuations are taxed accordingly, so as to remove the agents' incentives to move away from this state. In case the Boolean game already has at least one PNE (without taxation) then the ADCOP's solution will be a PNE state.

The correctness of this approach stems from Proposition 1. The ADCOP will check all the outcomes that can have a PNE state with some taxation scheme. In case there are no such outcomes then the cost of the solution will be higher than M, so the cost must be verified in order to know if there is an appropriate taxation scheme or not. According to Proposition 3, this verification is not needed in the special case of a Boolean game representation of the traffic light coordination problem. The correctness of the overall tax's minimality comes directly from the correctness of the chosen ADCOP algorithm.

Endriss et al. [9] suggest that taxation schemes may also be used to incentivize agents to reach some socially desirable outcome. For instance, in the traffic lights coordination problem such a social outcome may be the formation of a long green wave (longer than the mini-green-wave of size 3 that serves as an agent's personal goal). In order to facilitate such a social outcome, the only change to the ADCOP is the addition of another constraint to one of the agents. The cost of this (global) constraint is $M + \epsilon$ (for any $\epsilon > 0$) for valuations in which the social outcome is not achieved (and 0 otherwise).

4.3 k-ary SyncABB-1ph

Grubshtein et al. [12] introduced several complete ADCOP algorithms. The most simple, yet effective, complete ADCOP algorithm is SyncABB-1ph, which is an asymmetric version of the famous Synchronous Branch & Bound (SyncBB) algorithm [15]. After each step of the algorithm, when an agent adds an assignment to the *Current Partial Assignment (CPA)* and updates one direction of the bound, the CPA is sent back to the assigned agents to update its bound by the costs of all backwards directed constraints (back-checking). This is done by replacing the **CPA_MSG** message sent after each value assignment to the next agent with a **CPA_BACK_MSG** message to the preceding agent.

The problem with all the presented ADCOP algorithms, including SyncABB-1ph, is that they were developed to handle binary constraints, whereas the constraints in the above constructed ADCOP are k-ary. Thus, we slightly adjust the SyncABB-1ph algorithm to handle k-ary constraints:

As in the original pseudo-code [12], A_i refers to the agent that currently holds the CPA, A_j represents the agent that initiated the current back-checking, A_n is the last agent in the order, and B denotes the current bound. The modifications we made are to ensure that the CPA has all the assignments needed for calculating the constraint cost and that the last assignment was added by a neighbor of the current agent (line 2). The second condition assures that every constraint cost is evaluated only once for every full assignment. Moreover, the constraint cost is calculated using the entire CPA rather than with a single agent (line 3). In case A_i is the last agent in this particular constraint then the cost is added as in the regular SyncBB algorithm (with the slight change of the constraint being k-ary).

Algorithm 1 k-ary SyncABB-1ph: back-checking

when received $\langle$**CPA_BACK_MSG**, $CPA, cost\rangle$ **do**
1: $j \leftarrow CPA.lastId$
2: **if** all variables from A_i's constraint are in the CPA **and** A_j is a neighbor of A_i **then**
3: $f \leftarrow$ cost of the constraint with the CPA
4: **else**
5: $f \leftarrow 0$
6: **if** $cost + f \geq B$ **then**
7: send $\langle$**CPA_MSG**, $CPA\rangle$ to A_j
8: **else if** $A_i \neq A_1$ **then**
9: send $\langle$**CPA_BACK_MSG**, $CPA, cost + f\rangle$ to A_{i-1}
10: **else if** $A_j = A_n$ **then**
11: $B \leftarrow cost + f$
12: broadcast $\langle$**NEW_SOLUTION**, $CPA, B\rangle$
13: send $\langle$**CPA_MSG**, $CPA\rangle$ to A_n
14: **else**
15: $CPA.cost \leftarrow cost + f$
16: send $\langle$**CPA_MSG**, $CPA\rangle$ to A_{j+1}

5. EXPERIMENTAL EVALUATION

The Boolean games used in the following experiments represent traffic lights coordination problems of different grid sizes – 3x3, 4x4, 5x5, and 6x6. Problems were randomly generated and the reported results are averages over 100 different experiments for each setting.

5.1 Problem generation

For each experiment a random problem was generated. First, a traffic lights coordination problem was generated by randomly selecting the number (in the range $[0, max-cost)$) of vehicles in the lane between every two adjacent traffic lights. Next, the appropriate Boolean game representing the generated traffic lights coordination problem was constructed according to the rules described in Section 3.2. Then, an ADCOP problem was generated from the Boolean game using the procedure described in Section 4.2. Finally, the problem was solved using the k-ary SyncABB-1ph algorithm that was presented in Section 4.3.

5.2 Experimental results

The first part of the experimental evaluation is aimed to help one understand the properties of the randomly generated games. For this purpose two measures are considered.

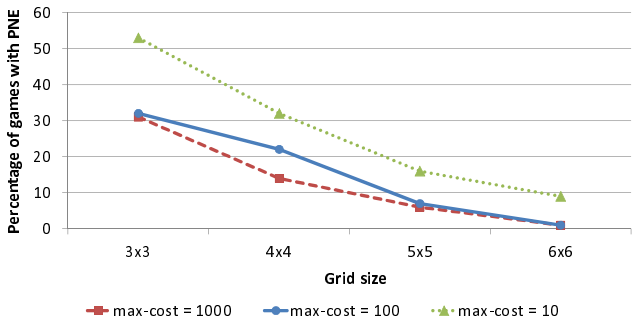

Figure 3: Percentage of games that have a PNE without any taxation scheme

Figure 3 presents the first measure – the percentage of games that originally have a PNE state (without taxation). It is easy to see that the probability for a PNE drops when the problems become larger and when the range of costs (size of max-cost) is wider. The costs affect the existence of a PNE because two valuations of the same variable have different costs, which cannot be balanced for an equilibrium. The fact that a larger range of costs increases the probability that the costs will be different, explains the effect of max-cost. The effect of the grid size can also be explained, since it is clearly harder to find stable states when there are more players participating in the game.

The second measure is the size of the overall tax with respect to the original costs of the game. The percentage of the overall tax is calculated as follows:

$$\frac{\sum_{p\in\Phi, b\in\mathbb{B}} \tau(p,b)}{\sum_{p\in\Phi, b\in\mathbb{B}} c(p,b)} * 100\% \quad (7)$$

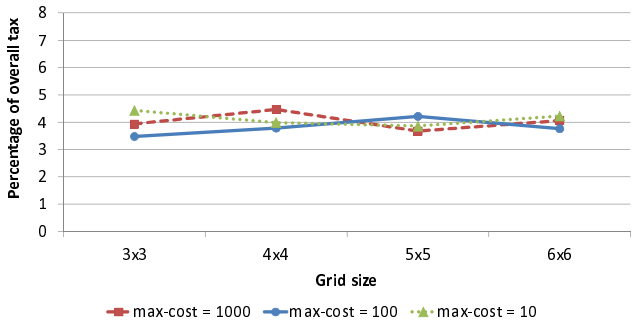

Figure 4: Percentage of overall tax

While it was shown that both the dimensions of the problem and the range of costs affect the existence of a PNE, the needed tax load to achieve stabilization does not seem to be affected by these parameters. Figure 4 shows that the overall needed tax is only about 4% in all of the problem settings in the evaluation.

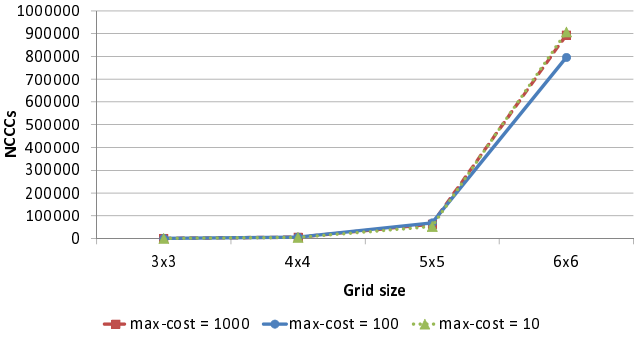

Figure 5: Mean number of NCCCs

In order to evaluate the algorithm, we consider the mean number of *Non-Concurrent Constraint Checks (NCCCs)*, which is a commonly used measure for the runtime performance of distributed constraints algorithms [20]. The exponential growth of the computational load with respect to the problem size is clearly seen in Figure 5 and is of no surprise as ADCOPs are NP-Hard problems. In contrast, the range of max-cost does not seem to have any effect on the performance.

In order to understand the impact of the pruning that was enabled by the transformation into an ADCOP, one can compare to a naive algorithm that exhaustively traverses the entire search space of the Boolean game. The naive algorithm must run through every possible outcome, check if it is a special outcome, and in case it is not an SO – calculate the overall tax. The approximate complexity of this naive approach is $n*2^n$ (there are 2^n possible outcomes and the algorithm must traverse all the variables in order to calculate the overall tax).

Grid size	k-ary SyncABB-1ph	Naive approach
3x3	550	4,608
4x4	6,468	1,048,576
5x5	63,841	838,860,800
6x6	893,863	2,473,901,162,496

Table 1: Runtime performance of k-ary SyncABB-1ph *vs.* a naive approach

Table 1 compares the NCCC results of the k-ary SyncABB-1ph algorithm with the approximate number of operations performed by the naive approach (problems with max-cost = 1000). Although the computational time of an NCCC may be somewhat different than that of the naive approach's operation, the difference in orders of magnitude between the two alternatives establishes the great impact of the pruning that is achieved when the ADCOP representation is used.

6. CONCLUSIONS

Taxation schemes that impose additional costs on the actions of agents in a Boolean game may in some cases incentivize the agents to reach a stable state. The present paper theoretically outlines the characteristics of Boolean games for which stabilization can be achieved by applying a taxation scheme. When a Boolean game is one that meets the theoretical criteria, one must search for the most appropriate taxation scheme.

The present paper proposes a method that effectively searches for the taxation scheme. The proposed method transforms the Boolean game into an Asymmetric Distributed Constraint Optimization Problem (ADCOP). The resulting ADCOP enables an effective search by using an existing algorithm with some minor adjustments.

The method is evaluated on Boolean games that represent a traffic light coordination game of different grid sizes. The runtime performance of the proposed method was shown to be better by several orders of magnitude than a naive approach for finding a taxation, that exhaustively goes over all the possible outcomes of the Boolean game. The substantial advantage of the proposed method stems from the effective pruning of the search space that is inherent to the used ADCOP algorithm.

7. REFERENCES

[1] E. Bonzon, M.-C. Lagasquie-Schiex, and J. Lang. Dependencies between players in Boolean games. *Int. J. Approx. Reasoning*, 50(6):899–914, June 2009.

[2] E. Bonzon, M.-C. Lagasquie-Schiex, J. Lang, and B. Zanuttini. Boolean games revisited. In *ECAI*, pages 265–269, 2006.

[3] E. Bonzon, M.-C. Lagasquie-Schiex, J. Lang, and B. Zanuttini. Compact preference representation and Boolean games. *Autonomous Agents and Multi-Agent Systems*, 18:1–35, 2009. 10.1007/s10458-008-9040-2.

[4] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional Ceteris Paribus preference statements. *Journal of Artificial Intelligence Research*, 21:2004, 2003.

[5] J. J. Cordes. Horizontal equity. In *The Encyclopedia of Taxation and Tax Policy*. Urban Institute Press, 1999.

[6] D. de Oliveira, A. L. C. Bazzan, and V. R. Lesser. Using cooperative mediation to coordinate traffic lights: a case study. In *AAMAS*, pages 463–470, 2005.

[7] P. E. Dunne, W. van der Hoek, S. Kraus, and M. Wooldridge. Cooperative Boolean games. In *AAMAS (2)*, pages 1015–1022, 2008.

[8] P. E. Dunne and M. Wooldridge. Towards tractable Boolean games. In *AAMAS*, pages 939–946, 2012.

[9] U. Endriss, S. Kraus, J. Lang, and M. Wooldridge. Designing incentives for Boolean games. In *AAMAS*, pages 79–86, 2011.

[10] U. Endriss, S. Kraus, J. Lang, and M. Wooldridge. Incentive engineering for Boolean games. In *IJCAI*, pages 2602–2607, 2011.

[11] J. Grant, S. Kraus, M. Wooldridge, and I. Zuckerman. Manipulating Boolean games through communication. In *IJCAI*, pages 210–215, 2011.

[12] A. Grubshtein, T. Grinshpoun, A. Meisels, and R. Zivan. Asymmetric distributed constraint optimization. In *Proc. of the 11th international workshop on Distributed Constraint Reasoning at IJCAI'09*, Pasadena CA, United States, July 2009.

[13] A. Grubshtein, R. Zivan, T. Grinshpoun, and A. Meisels. Local search for distributed asymmetric optimization. In *AAMAS*, pages 1015–1022, 2010.

[14] P. Harrenstein, W. van der Hoek, J.-J. Meyer, and C. Witteveen. Boolean games. In *Proceedings of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge*, pages 287–298, San Mateo, CA, 2001. Morgan Kaufmann Publishers.

[15] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *Proceedings of the Third International Conference on Principles and Practice of Constraint Programming (CP-97)*, pages 222–236, 1997.

[16] R. Junges and A. L. C. Bazzan. Evaluating the performance of DCOP algorithms in a real world, dynamic problem. In *AAMAS (2)*, pages 599–606, 2008.

[17] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[18] Robertson. TRANSYT: A traffic network study tool. Rep. LR 253, Road Res. Lab., London, 1969.

[19] L. Sauro and S. Villata. Dependency in cooperative Boolean games. *Journal of Logic and Computation*, 2011.

[20] R. Zivan and A. Meisels. Message delay and DisCSP search algorithms. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 46:415–439, 2006.