

Game Theoretical Insights in Strategic Patrolling: Model and Algorithm in Normal-Form

Nicola Gatti

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy
ngatti@elet.polimi.it

Abstract. In artificial intelligence literature there is a rising interest in studying strategic interaction situations. In these situations a number of rational agents act strategically, being in competition, and their analysis is carried out by employing game theoretical tools. One of the most challenging strategic interaction situation is the strategic patrolling: a guard patrols a number of houses in the attempt to catch a rob, which, in its turn, chooses a house to rob in the attempt to be not caught by the guard. Our contribution in this paper is twofold. Firstly, we provide a critique concerning the models presented in literature and we propose a model that is game theoretical satisfactory. Secondly, by exploit the game theoretical analysis to design a solving algorithm more efficient than state-of-the-art's ones.

1 Introduction

The study of strategic interaction situations, commonly named *non-cooperative games*, has been receiving more and more attention in artificial intelligence literature [7]. For instance, the problem of automating agents in negotiations [4] and in auctions [7] is usually modeled as a strategic interaction problem. Commonly, strategic interaction situations are tackled by employing game theoretical tools [3], in which one distinguishes the mechanism (i.e., the rules according which agents interact) from the strategies (i.e., the behaviors of the agents in the game). Given a mechanism, rational agents should behave in order to maximize their revenue.

An interesting open strategic interaction problem is the *strategic patrolling* [8, 9]. This problem is characterized by a *guard* that decides what houses to patrol and how often and by a *robber* that decides what house to strike. Obviously, the guard will not know in advance exactly where the robber will choose to strike. Moreover, the guard does not know with certainty what adversary it is facing. A common approach for choosing a strategy for agents in such a scenario is to model the scenario as a Bayesian game [3]. A Bayesian game is a game in which agents may belong to one or more types; the type of an agent determines its payoffs. The probability distribution over agents' types is common knowledge. The appropriate solution concept for these games is the Bayes-Nash equilibrium [3].

In [8] the authors propose a model for the strategic patrolling and an algorithm to solve it. Exactly they model the situation as a Bayesian game. The guard's actions are all the possible routes of houses, while the robber's action is the choice of a single house to rob. The robber can be of several types with a given probability distribution. Moreover, the rob can observe the actions undertaken by the guard and choose its optimal action on the basis of this observation. We show in this paper that the model proposed in [8] is not game

theoretically satisfactory. Indeed, we show that such model does not effectively capture the possibility available to the rob to observe the actions undertaken by the guard. A further issue risen by [8] concerns time required to compute solutions. Although the algorithm proposed by the authors finds solutions that are computationally less hard than Bayes-Nash, the computation of a solution does not result affordable even in very simple settings.

This paper provides two original contributions. The first contribution concerns the design of a strategic interaction model for the strategic patrolling that is game theoretically satisfactory. Precisely, we provide a critique to the model presented in [8], showing why it is not satisfactory in real-world settings. Subsequently, we provide a satisfactory Bayesian game model. The second contribution concerns the design of an efficient solving algorithm. Algorithmic game theory literature provides a number of on-the-shelf algorithms able to solve a large class of games [7]. However, these algorithms have exponential complexity in the worst-case and cannot address real-world settings. The exploitation of game theoretical analysis can lead to improve the efficiency of the solving algorithms and therefore to address real-world problems. This approach, although it is very preliminary, has been successfully followed in [2, 4], where the authors provide efficient algorithms for bargaining situations. The contribution of game theoretical analysis in the design of efficient algorithms can be twofold. Firstly, game theoretical analysis can be employed to reduce the space of search, e.g. by excluding all the strategy profiles that can be assured to be not of equilibrium independently of the parameters of the game. Secondly, it can be employed to "guide" the searching algorithm, e.g. by choosing specific orders over the strategy profiles according which the algorithm searches for the equilibrium [10]. In this paper we exploit game theoretical analysis to the limited extent of the first issue: the reduction of the space of search. We propose an algorithm much more efficient than on-the-shelf ones, its space of search being dramatically reduced with respect to the one considered by these algorithms. However, the space of search of the proposed algorithm raises exponentially in the size of the problem and therefore the algorithm needs to be improved by considering also the second issue concerning game theoretical analysis: the exploitation of information to efficiently guide the search. This second issue will be considered in future works.

This paper is structured as follows. The next section reviews the strategic patrolling model presented in [8] and provides a critique to it. Section 3 proposes a satisfactory game model for the considered situation. Section 4 provides some game theoretical insights concerning the proposed model and Section 5 exploits these to design a solving algorithm. Section 6 closes the paper.

2 Basic Strategic Patrolling Model and Critique

We briefly review the model proposed in [8]. The strategic situation to be considered is constituted by m houses, denoted by $1, \dots, m$, and two agents: a *guard*, denoted by g , and a *robber*, denoted by r . Essentially, g chooses a patrolling strategy, i.e. a route of houses, in the attempt to catch r , which, in its turn, chooses the house to rob in the attempt to be not caught by g . For the sake of simplicity, the following assumptions are commonly made:

- time is discretized in turns;
- g takes one turn to patrol one house independently of the patrolled house;
- r takes d turns to rob one house independently of the robbed house;
- the time needed by g to move between two houses is negligible.

Agents act simultaneously and their available actions are:

- g : it can choose a route of d houses to patrol, e.g. $\langle 1, 2, 3, \dots \rangle$;
 r : it can choose one house to rob, e.g. 1.

Possible outcomes are the following: if the house chosen by r is within the route chosen by g , then g catches r ; otherwise r robs the house. Players' preferences over the outcomes are expressed by the following payoffs:

- g : it assigns the outcome wherein r is caught an evaluation x^0 and assigns each outcome wherein house i is robbed an evaluation x^i . If g catches r , then g 's payoff is x^0 , otherwise its payoff is x^i where i is the robbed house. Customarily, it is assumed that $x^0 > \max\{x^i\}$ with $i > 0$;
- r : it assigns each house i an evaluation y^i and assigns its catch an evaluation y^0 . If r is caught by g , then r 's payoff is y^0 , otherwise its payoff is y^i where i is the robbed house. Customarily, it is assumed that $y^0 < \min\{y^i\}$ with $i > 0$.

Finally, it is assumed that g 's preferences are common knowledge, while r 's ones not. Precisely, it is assumed that r can be of n types with a given probability distribution. We denote type i of r by r_i . According to Harsanyi such a game is casted into an imperfect information game wherein *nature*, denoted by N , chooses initially the type of r and g does not perfectly know which game is playing [3]. An example with $m = 2$, $n = 2$, and $d = 1$ is depicted in Fig. 1

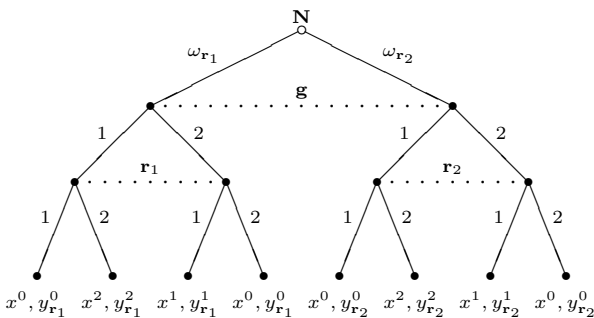


Figure 1. Game tree with two houses, denoted by 1 and 2, and with $d = 1$.

The appropriate solution concept for a game such the one we are dealing with is the Bayes-Nash equilibrium [3]. It prescribes one strategy σ_g^* for g and one, generally different, strategy $\sigma_{r_i}^*$ for each r_i . The peculiarity of this solution concept is that g maximizes its

expected payoff according to its beliefs, i.e. the probability distribution over r 's types. It can be showed – we omit the pertinent proof for reason of space – that agents' equilibrium strategies prescribe that g randomizes over all the possible routes wherein houses are patrolled only one time, e.g. with $d = 2$ all routes $\langle i, j \rangle$ such that $i \neq j$. It can be also showed that, in order for a strategy profile to be an equilibrium, at least one r 's type must randomize.

The above model is satisfactory when g and r act simultaneously. However, in real-world applications it is unreasonable to assume that r always acts at the turn where g starts to patrol. This is essentially due to two reasons. Firstly, g cannot synchronize the beginning of its patrolling route with r 's action, since g cannot observe r . Secondly, r could wait for one or more turns before choosing the house to rob in order to observe g 's strategy and take advantage from this observation. Thus, there is a discrepancy between the situation captured by the above model, i.e. r cannot make anything but choosing the house to rob, and the real-world situation, i.e. r can wait for some turns observing g 's strategy. This discrepancy must be carefully studied in order to evaluate the effectiveness of the above model. Exactly, we need to verify whether in real-world situations r violates the protocol prescribed by the above model. Technically speaking, we need to verify whether r can improve its revenue by waiting. In the affirmative case, r will wait, violating thus the protocol, and then the above model will be not satisfactory. In what follows we show that on the equilibrium path r waits.

At first, if r waits for one or more turns, the game could close after d turns. However, the above model captures a strategic situation d turns long and does not prescribe how g behaves after $t = d$. Since we are limiting our analysis to the above model, we can only assume that g repeats its equilibrium strategies at every d turns. With this extended model it can be showed that r can improve its expected utility by waiting for one or more turns in order to partially observe the route of g and exploiting this information to choose its strategy. (This is essentially because the model does not perfectly captures the situation we are considering: it implicitly assumes that r can enter the house to rob only at every d turns, meanwhile r could enter it at every turn.) We report an example. Consider a setting with three houses, $d = 2$, one r 's type, and $y^i = y^j = y^H$ for all $i, j > 0$. Call α^{ij} the probability prescribed by σ_g^* to make the route $\langle i, j \rangle$. It can be easily showed that $\alpha^{ij} = \frac{1}{6}$ for all $i \neq j$ with $i, j > 0$. If r immediately enters the house to rob, playing at the initial turn, its optimal strategy is to randomize with probability $\frac{1}{3}$ among the three houses and its expected utility is $\frac{2}{6}y^H + \frac{4}{6}y^0$. If r waits for one turn to observe g 's action, it can improve its expected utility. Precisely, if r has observed that g has patrolled the house i in the first turn, then r 's expected utility of choosing house i in the second turn is $\frac{4}{6}y^H + \frac{2}{6}y^0$ that, being $y^H > y^0$, is strictly greater than r 's expected utility of robbing at the initial turn. Therefore, r will wait for one turn rather than to rob a house immediately.

3 The Proposed Normal-Form Model

The failure of the model previously described is due to the neglecting of the possibility that r can wait: since in real-world situations r can improve its revenue by waiting, then it will violate the protocol. To overcome the drawbacks of this model, we must take into account the real-world possibility that r can wait for one or more turns. Two routes can be followed:

1. we cast the game into an extensive-form game and we explicitly take into account the action *wait* for r by introducing it at every decision node of r ;

2. we develop a normal-form game wherein the action *wait* is not explicitly taken into account, but, when the game is played in real-world situations, \mathbf{r} cannot improve its expected utility by waiting.

In this paper we limit our study to normal-form models for patrolling and therefore we follow the second route. The development of an appropriate extensive-form game will be explored in future works.

The model we propose is simple. We initially describe it and subsequently we discuss why it is satisfactory. The model prescribes that \mathbf{g} and \mathbf{r} act simultaneously. The actions available to \mathbf{r} are the same ones in the model presented in the previous section. The actions available to \mathbf{g} are the following:

\mathbf{g} : it chooses a house to patrol among all the possible ones.

The strategy of \mathbf{g} will be repeated at every turn. For instance, if the strategy chosen by \mathbf{g} is to patrol house 1, then \mathbf{g} will always patrol it. Practically, on the equilibrium path \mathbf{g} 's strategy will be fully mixed and therefore \mathbf{g} will randomize over all the houses at every turn with the same probability distribution.

Agents' payoffs are exactly the same ones we defined in the previous section. We provide agents' expected utilities since they will be fundamental in the analysis we carried out in the next section. The expected utility of \mathbf{r} can be easily calculated. Precisely, called α^i the probability prescribed by $\sigma_{\mathbf{g}}^*$ to patrol house i , the expected utility for \mathbf{r}_j of robbing house i is $EU_{\mathbf{r}_j}(i) = (1 - \alpha^i)^d \cdot y_{\mathbf{r}_j}^i + (1 - (1 - \alpha^i)^d) \cdot y_{\mathbf{r}_j}^0$. Essentially, it is the convex combination between $y_{\mathbf{r}_j}^i$, i.e. the \mathbf{r}_j 's evaluation of house i , and $y_{\mathbf{r}_j}^0$, i.e. the evaluation of \mathbf{r}_j 's caught, where the parameter of the convex combination is $(1 - \alpha^i)^d$, i.e. the probability that \mathbf{g} will never patrol house i for d turns.

The calculation of \mathbf{g} 's expected utility is more complicated. We give it by degrees. Suppose initially that \mathbf{r} can be only of one type. Called β^i the probability prescribed by $\sigma_{\mathbf{r}}^*$ to rob house i and supposed that \mathbf{g} will follow a mixed strategy based on probabilities α^l 's for the next $d - 1$ turns, the expected utility for \mathbf{g} of patrolling house j at the current turn is:

$$EU_{\mathbf{g}}(j) = \sum_{i=1, i \neq j}^m \left[x^i \cdot \beta^i \cdot (1 - \alpha^i)^{d-1} \right] + x^0 \cdot \left(\beta^j + \sum_{i=j, i \neq j}^m \left[\beta^i \cdot \left(1 - (1 - \alpha^i)^{d-1} \right) \right] \right)$$

Essentially, $EU_{\mathbf{g}}(j)$ gives the expected utility to choose house j at the current turn given that \mathbf{g} will employ a mixed strategy from turn $t = 1$ to turn $t = d$. Suppose now that \mathbf{r} can be of different types. The formula of $EU_{\mathbf{g}}(j)$ is defined as a weighted sum of a number of terms. The weights are the types' probabilities. The terms to sum are defined exactly as in the previous formula of $EU_{\mathbf{g}}(j)$ and refer to the single types. The formula of $EU_{\mathbf{g}}(j)$ is:

$$EU_{\mathbf{g}}(j) = \sum_{k=1}^n \left[\omega_{\mathbf{r}_k} \cdot \left(\sum_{i=1, i \neq j}^m \left[x^i \cdot \beta_{\mathbf{r}_k}^i \cdot (1 - \alpha^i)^{d-1} \right] + x^0 \cdot \left(\beta_{\mathbf{r}_k}^j + \sum_{i=1, i \neq j}^m \left[\beta_{\mathbf{r}_k}^i \cdot \left(1 - (1 - \alpha^i)^{d-1} \right) \right] \right) \right) \right]$$

Now we produce some considerations concerning the proposed model. Precisely, we need to verify whether in real-world situations agents will violate the protocol prescribed by the proposed model. Consider \mathbf{r} . The proposed model does not take into account the possibility that \mathbf{r} can wait for some turns, but in real-world setting it can.

Anyway, if \mathbf{r} waits for one or more turns, it can be easily observed that \mathbf{r} 's expected utility and \mathbf{g} 's one do not change. Then this possibility does not affect the employment of the model in real-world situations. Consider \mathbf{g} . The proposed model requires that \mathbf{g} employs the same strategy at every turn, but in real-world situations it can employ different strategies at different turns. Anyway, \mathbf{g} cannot make anything better than employing the same strategy at every turn, since it has not any information concerning when \mathbf{r} acts. It can be showed – we omit the pertinent proof for reason of space – that \mathbf{g} 's optimal strategy in the proposed model is consistent to \mathbf{g} 's optimal strategy in the extensive-form game wherein the \mathbf{r} 's action *wait* is explicitly taken into account.

4 Game Theoretical Insights

A game such as the one we are dealing with can be solved by employing on-the-shelf algorithms. Specifically, such a problem can be casted into a linear-complementarity problem and then solved by employing the Lemke-Howson's algorithm [6]. However, the computational complexity of the Lemke-Howson's algorithm is exponential in the size of the problem, i.e. the number m of houses and the number n of \mathbf{r} 's types. Practically, the production of exact solutions in real-world situations is not affordable and the computation of approximate solutions for very simple problems requires long time, e.g. more than 30 minutes with $m = 3$ and $n = 7$ [8]. The drawbacks related to on-the-shelf algorithms are due to the principle on which they are based: they search for an equilibrium strategy profile among all the possible ones neglecting any information concerning the specific problem to solve. Since the space composed of all the possible strategy profiles raises exponentially in the size of the problem, the search is inefficient also with very simple problems. This makes the study of real-world strategic situations by employing on-shelf-algorithm unaffordable. A route to follow to solve more efficiently strategic situations is to exploit game theoretical analysis. Precisely, the game theoretical analysis allows one to derive insights concerning regularities and singularities of the problem that can be employed to reduce the space of strategy profiles among which the algorithm searches for the equilibrium. Examples of similar works can be found in [1, 2, 4]. In what follows we game theoretically analyze the proposed game in the attempt to produce several insights to employ in the design of a solving algorithm more efficient than the state-of-the-art.

Considering \mathbf{g} 's strategies, we can state the following lemma.

Lemma 4.1 *On the equilibrium path \mathbf{g} 's strategy cannot be pure.*

Proof. The proof is by contradiction. Assume σ^* to be an equilibrium strategy profile wherein \mathbf{g} 's strategy is pure. On the equilibrium path every \mathbf{r} 's type believes that \mathbf{g} employs a pure strategies choosing a specific house to patrol (say house i). On the basis of these beliefs, since any \mathbf{r} 's type strictly prefers not to be caught rather than to be caught, no \mathbf{r} 's type will choose house i . On the basis of this fact, \mathbf{g} can improve its expected utility by patrolling a house different from i . We reach a contradiction and then σ^* is not an equilibrium. \square

We can state the following lemma, whose proof is omitted being similar to the proof of Lemma 3.1 but much longer.

Lemma 4.2 *On the equilibrium path \mathbf{g} 's strategy prescribes that every house can be patrolled with a strictly positive probability.*

Considering strategies of \mathbf{r} 's types, we can state the following lemma.

Lemma 4.3 *On the equilibrium path at least one \mathbf{r} 's type employs a mixed strategy.*

Proof. The proof is by contradiction. Assume σ^* to be an equilibrium strategy profile wherein the strategy of all r 's types is pure. It can be easily showed that g 's optimal strategy is a unique action, expect for a null-measure subset [5] of the space of the parameters. However, by Lemma 3.1, there is not any equilibrium wherein g 's strategy is pure. We reach a contradiction and therefore σ^* is not an equilibrium. \square

5 Improving Solving Algorithm Efficiency

In this section we show how the previous three lemmas can be employed to reduce the space of the strategy profiles among which one can search for an equilibrium. Precisely, we can exclude a large number of strategy profiles that we can assure to be not of equilibrium independently of the values of the agents' parameters, e.g. x^0 and x^1 . Although the proposed algorithm searches within a space of strategy profiles dramatically reduced with respect the state-of-the-art's one, this space raises exponentially in the size of the problem. Therefore, in order to tackle real-world problems, the proposed algorithm must be improved by introducing heuristics that efficiently guide the search. We will discuss this topic in future works.

On the basis of Lemma 3.2 every equilibrium strategy profile for the game we are dealing with is characterized by $\alpha^i \in (0, 1)$ for any $i \in \{1, \dots, m\}$. Since these variables are bound by the equation $\sum_{i=1}^m \alpha^i = 1$, the number of free variables related to g 's strategy is $m - 1$. Furthermore, on the basis of Lemma 3.3 we know that every equilibrium strategy profile is characterized by at least one r 's type that randomizes. The exact number of r 's types that randomize and the number of actions over which each specific randomizing type randomizes in an equilibrium strategy profile can be determined by studying the pertinent solving equation sets and by excluding singularities of these. For the sake of clarity, we study the possible randomizations of r 's types by degrees: at first when the number of r 's types is one and subsequently when r 's types are more than one.

5.1 The Base Case: One Robber's Type

We consider the situation in which the number of r 's types is one. As customarily in game theory, in a two-player game, the randomization probabilities related to each player are computed in such a way the other player can effectively randomize, i.e. every action over which a player randomizes gives it the same expected utility and no other action gives it more than randomizing. In the game we are studying, the randomization probabilities of g will be computed in such a way the actions over which r randomizes give r the same expected utility and *vice versa*. Technically speaking, we have two equation sets: the first one, say Φ_g , wherein the variables are the randomization probabilities of g , i.e. α^i 's, and the equations are of the form $EU_r(i) = EU_r(j)$ for all actions i, j over which r randomizes, the second one, say Φ_r , wherein the variables are the randomization probabilities of r , i.e. β^i 's, and the equations are of the form $EU_g(i) = EU_g(j)$ for all actions i, j over which g randomizes. On the basis of Lemma 3.2, we know that equation set Φ_g is characterized by $m - 1$ variables and that equation set Φ_r is characterized by $m - 1$ independent equations. We need to find the number of actions over which r randomizes in order to have two well-defined equation sets. Since, when r randomizes over m actions, $m - 1$ variables are introduced in equation set Φ_r and $m - 1$ equations are introduced in equation set Φ_g , the appropriate number of actions over which r randomizes on the equilibrium path is m . Notice that, if r randomizes over a number of actions lower than m , then equation set Φ_r would

present a number of variables lower than the number of equations and therefore it does not admit any solution.

Easily, since at the equilibrium both g and r randomize over all the possible actions and since Φ_g and Φ_r , being linear equation sets, admit a unique solution, then the game admits a unique equilibrium strategy. In this equilibrium $\alpha^i, \beta^j \in (0, 1)$. Since agents' equilibrium strategies can be provided in closed form, no search is needed.

By imposing $EU_r(i) = EU_r(j)$ for any $i, j \in \{1, \dots, m\}$, we can calculate the values of α^i . Exactly, called $\gamma(i, j) = \sqrt[d]{\frac{y^i - y^0}{y^j - y^0}}$, by trivial mathematics we obtain: $\alpha^i = \frac{1 + \sum_{j=1}^m [\gamma(i, j) - 1]}{\sum_{j=1}^m [\gamma(i, j)]}$.

By imposing $EU_g(i) = EU_g(j)$ for any $i, j \in \{1, \dots, m\}$, we can calculate the values of β^i . Exactly, called $\varepsilon(i, j) = \frac{(x^0 - x^i) \cdot (1 - \alpha^i)^{d-1}}{(x^0 - x^j) \cdot (1 - \alpha^j)^{d-1}}$, by trivial mathematics we obtain: $\beta^i = \frac{1}{\sum_{j=1}^m [\varepsilon(i, j)]}$.

5.2 The General Case: More Robber's Types

We consider the situation in which the number of r 's types can be any. The analysis is similar to the basic case, but it is more complicated. Meanwhile with a unique r 's type there is a unique possible set of actions over which r can randomize that makes the above equation sets well-defined, i.e. all the m houses, with more r 's types it does not, e.g. r_1 could randomize over $m - 3$ houses and r_2 over 3 houses. Furthermore, among all the possible ways with which r 's types can randomize that makes the pertinent equation sets well-defined, only one leads to an equilibrium. We need therefore to search for this.

At first, we characterize the strategy profiles with respect to the actions over which r 's randomizing types randomize (we exclude all r 's types that do not randomize). We use a $n \times m$ binary matrix R_r where the rows denote the r 's types and the columns denote the houses, i.e.

$$R_r = \left. \begin{array}{c} \overbrace{\begin{bmatrix} 1 & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}}^{m \text{ houses}} \right\} n \text{ types}$$

Precisely, the meaning of R_r is the following: $R_r(i, j) = 1$ means that r_i randomizes over house j , while $R_r(i, j) = 0$ means that r_i does not randomize over house j . Notice that, in order for R_r to be well-defined, the following constraint must hold: $\sum_{j=1}^m R_r(i, j) \neq 1$ for any $i \in \{1, \dots, n\}$ (i.e., a randomizing agent must randomize over two actions at least). We call this constraint $C1$.

Given a matrix R_r we can build equation set Φ_g for the calculation of g 's randomization probabilities. Trivially, in order for Φ_g to be well-defined, two properties must hold: Φ_g must be composed of $m - 1$ independent equations and all α^i 's must be present in Φ_g . These two properties can be translated into the following two constraints over R_r :

- $C2$: for any $j \in \{1, \dots, m\}$ it holds $\sum_{i=1}^n R_r(i, j) > 0$ (i.e., each variable α^j must be present in Φ_g),
- $C3$: $\sum_{i=1}^n [\max\{\sum_{j=1}^m [R_r(i, j)] - 1, 0\}] = m - 1$ (i.e., the number of independent equations must be $m - 1$).

Similarly, given a matrix R_r we can also build equation set Φ_r for the calculation of the randomization probabilities of r 's types. It can showed that, in order for Φ_r to be well-defined, no further constraint over R_r is needed.

With $m = 3$ and $n = 2$, all the feasible R_r s, i.e. all ones that satisfy $C1, C2, C3$, are:

$$\left\{ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \right. \\ \left. \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \right\}$$

Given a matrix R_r , it can be possible to find univocally the values of α^i s and $\beta_{r_i}^j$ s by employing equations similar to the ones employed in the previous section and, subsequently, it is possible to verify whether agents' strategies computed on the basis of R_r lead or not to an equilibrium. Precisely, we need to verify that:

- all $\alpha^i \in (0, 1)$;
- all the $\beta_{r_i}^j$ s prescribed by R_r belong to $(0, 1)$;
- all the randomizing r 's types cannot make anything better than randomizing.

Therefore, we can limit the search for an equilibrium to the search for a feasible R_r that leads to an equilibrium. This allows one to dramatically reduce the space of search and reduce thus the time needed to compute a solution. Consider for instance the setting with $m = 3$, $n = 2$, and $d = 2$. The space over which on-the-shelf algorithms search is the set of vertices of a complex 9-polytope, while the space of all the feasible R_r s is composed of eight elements. We report our algorithm in Algorithm 1. Currently, all the feasible R_r s are statically ordered in lexicographic order.

Algorithm 1: EQUILIBRIUM_FINDER

```

1 for all feasible  $R_r$  do
2   solve  $\Phi_g$ 
3   if all  $\alpha^i \in (0, 1)$  then
4     calculate optimal strategies of randomizing  $r$ 's types on the basis of
5        $\alpha^i$ s
6     if no randomizing type deviates from actions in  $R_r$  then
7       calculate optimal strategies of non-randomizing  $r$ 's types on the
8       basis of  $\alpha^i$ s
9       solve  $\Phi_r$ 
10      if all  $\beta_{r_j}^i \in (0, 1)$  then
11        return  $R_r, \alpha^i$ s, and  $\beta_{r_i}^j$ 

```

5.3 Experimental Considerations

We provide a preliminary experimental evaluation of the proposed algorithm. In order to evaluate it, we compare the average time it requires for the computation of an equilibrium with respect the one required by the algorithm proposed in [8]. Since our algorithm is not directly comparable with the algorithm presented in [8], considering a different model, we have modified our algorithm to solve the model present in [8]. The experimental results reported below refer to this modified version of the algorithm. No significant difference in terms of computational time ($< 5\%$) has been found between the application of our algorithm to the model proposed in [8] and the one we present in Section 3. The algorithm proposed in [8], implemented in CPLEX, requires more than 30 minutes to compute approximate solutions for settings with $m = 3$, $n = 7$ and settings with $m = 4$, $n = 6$. We have implemented our algorithm in C and we have considered all the settings with $m = 3, 4$ and $n \in \{6, \dots, 13\}$. For each setting we have considered 10^3 different agents' payoffs calculated at random in $(0, 1)$. We have used a 1.4GHz CPU with 500MBytes

RAM. Experimental results are reported in Tab. 5.3. Although the proposed algorithm is a prominent step ahead with respect to the state-of-the-art, it cannot address real-world settings. For instance, it requires more than one day computation for settings with $m = 20$ and $n = 10$. The efficiency of the algorithm can be improved by employing heuristics to order dynamically the feasible R_r s.

houses	types						
	6	7	8	9	10	11	12
3	0.007	0.011	0.017	0.024	0.033	0.043	0.055
4	0.190	0.352	0.720	1.015	1.532	1.852	2.231

Table 1. Average time (in seconds) required by Algorithm 1 for the computation of the equilibrium.

6 Conclusions and Future Works

The strategic patrolling is a challenging problem that has found a lot of attention in artificial intelligence literature. In this paper we consider the principal strategic patrolling model presented in literature. We provide two prominent contributions. At first, we show that the model proposed in the state-of-the-art presents some unsatisfactory issues concerning game theory and subsequently we provide a model that is game theoretically satisfactory. Then, we have analyzed the considered game in order to produce some insights concerning regularities and singularities of the corresponding solving equation sets. These insights have been subsequently employed in the design of a solving algorithm. This algorithm has been showed to be much more efficient than state-of-the-arts' ones.

Our intention is to develop the proposed work along two main directions. The first one concerns the provision of an appropriate extensive-form model for the considered strategic interaction situation. We will study furthermore leadership with commitment to mixed strategies in our model. The second one is more general and concerns the development of a general approach to exploit game theoretical analysis to enable algorithms to afford real-world setting game situations, e.g. by employing genetic algorithms.

REFERENCES

- [1] F. Di Giunta and N. Gatti, 'Alternating-offers bargaining under one-sided uncertainty on deadlines', in *Proceedings of ECAI*, pp. 225–229, Riva del Garda, Italy, (2006).
- [2] F. Di Giunta and N. Gatti, 'Bargaining over multiple issues in finite horizon alternating-offers protocol', *Annals of Mathematics in Artificial Intelligence*, **47**(3-4), 251–271, (2006).
- [3] D. Fudenberg and J. Tirole, *Game Theory*, The MIT Press, Cambridge, MA, USA, 1991.
- [4] N. Gatti, F. Di Giunta, and S. Marino, 'Alternating-offers bargaining with one-sided uncertain deadlines: an efficient algorithm', *Artificial Intelligence*, **172**(8-9), 1119–1157, (2008).
- [5] P. R. Halmos, *Measure Theory*, Springer, Berlin, Germany, 1974.
- [6] C. Lemke, 'Some pivot schemes for the linear complementarity problem', *Mathematical Programming Study*, **7**, 15–35, (1978).
- [7] N. Nisam, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*, Cambridge University Press, New York, USA, 2007.
- [8] P. Paruchuri, J. P. Pearce, M. Tambe, F. Ordonez, and S. Kraus, 'An efficient heuristic approach for security against multiple adversaries', in *Proceedings of AAMAS*, pp. 311–318, Honolulu, USA, (2007).
- [9] P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus, 'Security in multiagent systems by policy randomization', in *Proceedings of AAMAS*, pp. 273–280, Hakodate, Japan, (2006).
- [10] R. Porter, E. Nudelman, and Y. Shoham, 'Simple search methods for finding a nash equilibrium', in *Proceedings of AAAI*, p. 664669, San Jose, USA, (2004).