Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Team–Maxmin Equilibrium: Efficiency Bounds and Algorithms

Nicola Basilico
University of Milan
Via Comelico, 39/41
Milano, Italy
nicola.basilico@unimi.it

Andrea Celli, Giuseppe De Nittis, Nicola Gatti
Politecnico di Milano
Piazza Leonardo da Vinci, 32
Milano, Italy
{andrea.celli, giuseppe.denittis, nicola.gatti}@polimi.it

Abstract

The *Team-maxmin equilibrium* prescribes the optimal strategies for a team of rational players sharing the same goal and without the capability of correlating their strategies in strategic games against an adversary. This solution concept can capture situations in which an agent controls multiple resources—corresponding to the team members—that cannot communicate. It is known that such equilibrium always exists and it is unique (except degenerate cases) and these properties make it a credible solution concept to be used in real–world applications, especially in security scenarios. Nevertheless, to the best of our knowledge, the Team–maxmin equilibrium is almost completely unexplored in the literature. In this paper, we investigate bounds of (in)efficiency of the Team–maxmin equilibrium w.r.t. the Nash equilibria and w.r.t. the Maxmin equilibrium when the team members can play correlated strategies. Furthermore, we study a number of algorithms to find and/or approximate an equilibrium, discussing their theoretical guarantees and evaluating their performance by using a standard testbed of game instances.

Introduction

The computational study of game–theoretic solutions concepts is among the most important challenges addressed in the last decade of Computer Science (Deng, Papadimitriou, and Safra 2002). These problems acquired particular relevance in Artificial Intelligence, where the goal is to design physical or software agents that must behave optimally in strategic situations. In addition to the well–known Nash equilibrium (Nash 1951), other solution concepts received attention in the Artificial Intelligence literature thanks to their application in security domains. Examples include Maxmin equilibrium for zero–sum games under various forms of constraints over the actions of the players (Jain et al. 2010) and Stackelberg (a.k.a. leader–follower) equilibrium (Conitzer and Sandholm 2006).

While a large part of the literature focuses on 2–player games, few results are known about games with more players—except for games with a very specific structure, for example, congestion games (Nisan et al. 2007). In this paper, we focus on the Team–maxmin equilibrium proposed by (von Stengel and Koller 1997). It applies to zero–sum games between a team and an adversary. The team is defined as a set of players with the same utility function U_T and without the capability of synchronizing their actions. The adversary is a single player with utility function $-U_T$. These games can model many realistic security scenarios, for example those where multiple non–coordinating agents share the common objective of defending an environment against a malicious attacker. In (Jiang et al. 2013), a security setting of such type is studied and an analysis of the price of mis–coordination in the proposed security games is conducted. The Team–maxmin equilibrium plays a crucial role also in infinitely repeated games and role assignment problems (Moon and Conitzer 2016), where it is necessary to compute threat points. The current approach to tackle these problems, in games with more than two players, is considering the correlated threat point (Kontogiannis and Spirakis 2008) or employing approximation algorithms to avoid the use of mathematical programming (Andersen and Conitzer 2013). Our techniques allow the computation of punishment strategies (leading to the threat points) in the general scenario in which players, other than the defector, cannot coordinate strategy execution.

The study of Team–maxmin equilibrium is almost completely unexplored. It is known that it always exists, it is unique except for degeneracies, and it is the best Nash equilibrium for the team, but, to the best of our knowledge, only two computational works deal with this solution concept. (Borgs et al. 2010) show that the Minmax value (equivalently the Team–maxmin value) is inapproximable in additive sense within $\frac{3}{m^2}$ even in 3–player games with m actions per player and binary payoffs (but nothing is known about the membership to the APX class or some super class); (Hansen et al. 2008) strengthen the previous complexity result and provide a quasi–polynomial time ϵ–approximation (in additive sense) algorithm. Only (Lim 1997; Alpern and Lim 1998) deal with the mathematical derivation for a specific class of games with an adversary, i.e., rendezvous–evasion games. Instead, a number of works deal with team games without adversary. We just cite a few for the sake of completeness. Team games were first proposed in (Palfrey and Rosenthal 1983) as voting games, then studied in repeated and absorbing games to understand the interaction among the players (Bornstein, Erev, and Goren 1994; Solan 2000) and more recently in Markov games with noisy

payoffs (Wang and Sandholm 2002).

Original contributions We provide two main contributions. First, we study the relationship, in terms of efficiency for the team, between Nash equilibrium (i.e., when players are not teammates), Team–maxmin equilibrium, and Correlated–team maxmin equilibrium (i.e., the Maxmin equilibrium when all the team members can play in correlated strategies and therefore can synchronize the execution of their actions). We show that, even in the same instances with binary payoffs, the worst Nash equilibrium may be arbitrarily worse than the Team–maxmin equilibrium that, in its turn, may be arbitrarily worse (in this case only asymptotically) than the Correlated–team maxmin equilibrium. We provide exact bounds for the inefficiency and we design an algorithm that, given a correlated strategy of the team, returns in polynomial time a mixed strategy of the team minimizing the worst–case ratio between the utility given by the correlated strategy and the utility given by the mixed strategy. Second, we provide some algorithms to find and/or approximate the Team–maxmin equilibrium, we discuss their theoretical guarantees and evaluate them in practice by means of a standard testbed (Nudelman et al. 2004). We also identify the limits of such algorithms and discuss which ones are the best to be adopted depending on the instance to be solved. For the sake of presentation, the proofs of the theorems are presented in (Basilico et al. 2016).

Preliminaries

A normal–form game is a tuple (N, A, U) where: $N = \{1, 2, \ldots, n\}$ is the set of players; $A = \times_{i \in N} A_i$ is the set of player i's actions, where $A_i = \{a_1, a_2, \ldots, a_{m_i}\}$; $U = \{U_1, U_2, \ldots, U_n\}$ is the utility function of player i, where $U_i : A \rightarrow \mathbb{R}$. A strategy profile is defined as $s = (s_1, s_2, \ldots, s_n)$, where $s_i \in \Delta(A_i)$ is player i's mixed strategy and $\Delta(A_i)$ is the set of all the probability distributions over A_i. As customary, $-i$ denotes the set containing all the players except player i. We study games in which the set of players $T = \{1, 2, \ldots, n-1\}$ constitutes a team whose members have the same utility function U_T. Player n is an adversary of the team and her utility function is $-U_T$.

When the teammates cannot coordinate since, for example, there is no communication, and each player takes decisions independently, the appropriate solution concept is the Nash equilibrium, which prescribes a strategy profile where each player i's strategy s_i is a best response to s_{-i}. In 2–player zero–sum games, a Nash equilibrium is a pair of Maxmin/Minmax strategies and can be computed in polynomial time. In arbitrary games, the computation of a Nash equilibrium is PPAD–complete even when the number of players is fixed (Daskalakis, Goldberg, and Papadimitriou 2009). Instead, when the teammates can coordinate, we distinguish two forms of coordinations: *correlated*, in which a correlating device decides a joint action (i.e., an action profile specifying one action per teammate) and then communicates each teammate her action, and *non–correlated*, in which each player plays independently from the others.

When the coordination is non–correlated, players are subject to the inability of correlating their actions, and their strategy s_i is mixed, as defined above for a generic normal–form game. In other words, teammates can jointly decide their strategies, but they cannot synchronize their actions, which must be drawn independently. The appropriate solution concept in such cases is the Team–maxmin equilibrium.

A Team–maxmin equilibrium is a Nash equilibrium with the properties of being unique (except for degeneracies) and the best one for the team. These properties are very appealing in real–world settings, since they allow one to avoid the equilibrium selection problem which affects the Nash equilibrium. In security applications, for instance, the equilibrium uniqueness allows one to perfectly forecast the behavior of the attacker (adversary). When the number of players is given, finding a Team–maxmin equilibrium is FNP–hard and the Team–maxmin value is inapproximable in additive sense even when the payoffs are binary (Hansen et al. 2008)[1]. In (Hansen et al. 2008), the authors provide a quasi–polynomial–time ϵ–approximation (in additive sense) algorithm. Furthermore, a Team–maxmin equilibrium may contain irrational probabilities even with 2 teammates and 3 different values of payoffs[2]. To the best of our knowledge, no experimental evaluation of algorithms for finding the Team–maxmin equilibrium is present in literature.

When players can synchronize their actions, the team strategy is said to be correlated. Given the set of team action profiles defined as $A_T = \times_{i \in T} A_i$, a correlated team strategy is defined as $p \in \Delta(A_T)$. In other words, teammates can jointly decide and execute their strategy. The team is then equivalent to a single player whose actions are the joint team action profiles. In such case, the appropriate solution concept for the team and the adversary is a pair of Maxmin/Minmax strategies that, for the sake of clarity, we call *Correlated–team maxmin equilibrium*. This equilibrium can be found by means of linear programming since it can be formulated as a maxmin problem in which the max player's action space is given by the Cartesian product of the action space of each teammate. Notice that the size of the input is exponential in the number of teammates and therefore approximation algorithms for games with many team members are necessary in practice.

Furthermore, it is not known the price—in terms of inefficiency—paid by a team due to the inability of synchronizing the execution of their actions. This would allow one to understand how the Team–maxmin equilibrium is inefficient w.r.t. the Correlated–team maxmin equilibrium, or equivalently, how well the Team-maxmin equilibrium approximates the Correlated-team maxmin equilibrium. Another open problem is studying the gain a set of players sharing the same goal would have in forming a team and coordinating their mixed strategies (i.e., how the Nash equilibrium is inefficient w.r.t. the Team-maxmin equilibrium, or

[1] Rigorously speaking, (Hansen et al. 2008) studies the Minmax strategy when there is a single max player and multiple min players. The problem of finding the Team–maxmin equilibrium in zero–sum adversarial team games can be formulated as the problem of finding such Minmax strategy and *vice versa*.

[2] The proof, provided in (Hansen et al. 2008), contains a minor flaw. In (Basilico et al. 2016), we provide a correct revision of the proof with all the calculations, omitted in the original proof.

equivalently, how well the Nash equilibrium approximates the Team-maxmin equilibrium).

Nash, Team-maxmin, and Correlated–team maxmin equilibria

We study the relationships between Nash equilibrium and Team–maxmin equilibrium in terms of efficiency for the team. In our analysis, we resort to the concept of Price of Anarchy (POA), showing that Nash equilibrium—precisely, the worst case Nash equilibrium—may be arbitrarily inefficient w.r.t. the Team–maxmin equilibrium—corresponding to the best Nash equilibrium for the team. In this case the POA provides a measure about the inefficiency that a group of players with the same goal would have if they do not form a team. To have coherent results with the definition of POA, we consider games with payoffs in the range $[0, 1]$. We observe that our results will hold without loss of generality since, given any arbitrary game, we can produce an equivalent game in which the payoffs are in such a range by using an affine transformation. Furthermore, for the sake of presentation, we consider only games in which $m_1 = \ldots = m_n = m$. The generalization of our results when players may have different numbers of actions is straightforward.

Theorem 1 *The Price of Anarchy (*POA*) of the Nash equilibrium w.r.t. the Team–maxmin equilibrium may be* POA$= \infty$ *even in games with 3 players (2 teammates), 2 actions per player, and binary payoffs.*

In order to evaluate the inefficiency of the Team–maxmin equilibrium w.r.t. the Correlated–team maxmin equilibrium, we introduce a new index similar to the *mediation value* proposed in (Ashlagi, Monderer, and Tennenholtz 2008) and following the same rationale of the POA. We call such an index Price of Uncorrelation (POU) and we define it as the ratio between the team's utility provided by the Correlated–team maxmin equilibrium and that provided by the Team–maxmin equilibrium. POU provides a measure of the inefficiency due to the impossibility, for the teammates, of synchronizing the execution of their strategies.

Definition 1 *Let us consider an n–player game. The Price of Uncorrelation (*POU*) is defined as* POU $= \frac{v_C^{team}}{v_M^{team}} \geqslant 1$ *where v_C^{team} is the Correlated–team maxmin value of the team and v_M^{team} is the Team–maxmin value of the team.*

We initially provide a lower bound over the worst–case POU.

Theorem 2 *The* POU *of the Team–maxmin equilibrium w.r.t. the Correlated–team maxmin equilibrium may be* POU$= m^{n-2}$ *even in games with binary payoffs.*

Now, we provide an upper bound over the worst–case POU.

Theorem 3 *Given any n–player game and a Correlated–team maxmin equilibrium providing the team a utility of v, it is always possible to find in polynomial time a mixed strategy profile providing a utility of at least $\frac{v}{m^{n-2}}$ to the team and therefore* POU *is never larger than m^{n-2}.*

We observe that Theorem 2 shows that the upper bound of POU is at least m^{n-2}, while Theorem 3 shows that POU cannot be larger than m^{n-2}. Therefore, POU is arbitrarily large only asymptotically. In other words, POU $= \infty$ only when m or n go to ∞[3]. More importantly, the proof of Theorem 3 provides a polynomial–time algorithm to find a mixed strategy of the team given a correlated strategy and this algorithm is the best possible one in terms of worst–case minimization of POU. The algorithm computes mixed strategies for the team members as follows. Given the Correlated–team maxmin equilibrium $p \in \Delta(A_1 \times \ldots \times A_{n-1})$, the mixed strategy of player 1 (s_1) is such that each action a_1 is played with the probability that a_1 is chosen in p, that is $s_1(a_1) = \sum_{a_{-1} \in A_{-1}} p(a_1, a_{-1})$. Every other team member $i \in N \backslash \{1, n\}$ plays uniformly over the actions she plays with strictly positive probability in p. Since the computation of a Correlated–team maxmin equilibrium can be done in polynomial time, such an algorithm is a polynomial–time approximation algorithm for the Team–maxmin equilibrium.

Furthermore, notice that POU rises polynomially in the number of actions m and exponentially in the number of players n. Interestingly, the instances used in the proof of Theorem 2 generalize the instances used in the proof of Theorem 1. Indeed, it can be observed that the POA of the Nash equilibrium w.r.t. the Team–maxmin equilibrium is ∞ in the instances used in the proof of Theorem 2. Therefore, there are instances in which the worst Nash equilibrium is arbitrarily worse than the Team–maxmin equilibrium and, in its turn, the Team–maxmin equilibrium is arbitrarily worse (in this case only asymptotically) than the Correlated–team maxmin equilibrium.

For the sake of completeness, we state the following result, showing the lower bound of POU.

Theorem 4 *The* POU *of the Team–maxmin equilibrium w.r.t. the Correlated–team maxmin equilibrium may be* POU$= 1$ *even in games with binary payoffs.*

Algorithms to find and/or approximate a Team-maxmin equilibrium

In the following, we describe four algorithms to find/approximate the Team–maxmin equilibrium.

Global optimization The problem of finding the Team–maxmin equilibrium can be formulated as a non–linear non–convex mathematical program as follows:

[3] A more accurate bound can be obtained by substituting m with the size of the equilibrium support, showing that the inefficiency increases as the equilibrium support increases.

Algorithm 1 `SupportEnumeration`

1: $v^* = +\infty$
2: **for all** $i \in T$ **do**
3: $\quad P_i = \{(V_i^1, m_i^1), (V_i^2, m_i^2), \dots \mid \forall j, V_i^j \subseteq A_i, \sum_{a \in V_i} m_i^j(a) = \Gamma\}$
4: $C = \times_{i \in T} P_i$
5: **for all** $\big((V_1, m_1), (V_2, m_2), \dots, (V_{n-1}, m_{n-1})\big) \in C$ **do**
6: $\quad$ **for all** $i \in T$ **do**
7: $\quad\quad s_i(a_i) = \begin{cases} \frac{m_i(a_i)}{\Gamma}, & \text{if } a_i \in V_i \\ 0 & \text{otherwise} \end{cases}$
8: $\quad v^* = \max\{v^*, \min_{s_n} U_T(s_1, s_2, \dots, s_{n-1})\}$
9: **return** v^*

$$
\begin{aligned}
\max_{v, s_i} \quad & v \\
\text{s.t.} \quad & v - \sum_{a_T \in A_T} U_T(a_T, a_n) \prod_{i \in T} s_i(a_i) \leqslant 0 && \forall a_n \in A_n \\
& \sum_{a_i \in A_i} s_i(a_i) = 1 && \forall i \in T \\
& s_i(a_i) \geqslant 0 && \forall i \in T, a_i \in A_i
\end{aligned}
$$

In order to find an exact (within a given accuracy) Team–maxmin equilibrium, we resort to global optimization tools. Global optimization obviously requires exponential time. In particular, we use BARON (Tawarmalani and Sahinidis 2005) solver, since it is the best performing solver for completely continuous problems among all the existing global optimization solvers (Neumaier et al. 2005). Most importantly, BARON, if terminated prematurely, returns a lower bound, corresponding to the value of the best solution found so far, and an upper bound, corresponding to the tightest upper bound over the Team–maxmin value found so far.

Reconstruction from correlated strategies We approximate the Team–maxmin equilibrium by using a simple variation of the algorithm described previously to find a mixed strategy from a correlated one. First, the algorithm finds a Correlated–team maxmin by means of linear programming. Second, we derive the mixed strategy. The algorithm can be parametrized by exchanging player 1 with each player of the team. This leads to $n - 1$ different mixed strategies from the same correlated strategy. The algorithm returns the best one for the team. Since the Correlated–team maxmin equilibrium is always better than the Team–maxmin equilibrium, this algorithm assures an approximation factor of at least $\frac{1}{m^{n-2}}$ showing that the problem is in Poly–APX when n is given.

Support enumeration In (Hansen et al. 2008), the authors show how in n–players finite strategic games the minmax value of a player can be approximated (from above) within an arbitrary additive error $\epsilon > 0$. The algorithmic approach to guarantee such approximation leverages the concept of *simple* strategies as introduced in (Lipton and Young 1994) and can be exploited to approximate the Team–maxmin value, as we fully report in Algorithm 1.

The algorithm enumerates joint action multi–sets (specifying, for each player i, a subset of actions that can contain duplicate elements) of cardinality $\Gamma = \left\lceil \frac{\ln |A_i|}{2\epsilon^2} \right\rceil$. In Algorithm 1, we denote a multiset with the pair (V_i, m_i) where $v_i \subseteq A_i$ and $m(a_i)$ returns the multiplicy of $a_i \in A_i$. The strategy for each player is then obtained from an uniform distribution over the considered multi–set (for example, if action a_i has k duplicates it will be selected with probability k/Γ). The adversary's best response and the corresponding value for the team is then computed. The algorithm returns the joint support maximizing the value of the team. By adapting the analysis made in (Hansen et al. 2008), it can be easily shown that Algorithm 1 approximates the Team–maxmin value with additive error of at most ϵ with a number of iterations equal to $\binom{m+\Gamma-1}{\Gamma}^{n-1}$.

In the table below we report, for some m, the number of iterations required by the algorithm to assure a given approximation with additive error not larger than ϵ.

m	5	5	5	10	10	10
ϵ	0.9	0.5	0.1	0.9	0.5	0.1
iterations	$> 2^4$	$> 2^{12}$	$> 2^{41}$	$> 2^{11}$	$> 2^{21}$	$> 2^{87}$

Algorithm 2 `IteratedLP`

1: $\forall i \in T, s_i^{cur} \leftarrow \hat{s}_i$
2: **repeat**
3: $\quad$ **for all** $i \in T$ **do**
4: $\quad\quad$
$$
\begin{aligned}
v_i^* = \max \quad & v_i \\
v_i \leqslant & \sum_{a_T \in A_T} x_{a_i} U_T(a_T, a_n) \prod_{j \in T \setminus \{i\}} s_j^{cur}(a_j) && \forall a_n \in A_n \\
& \sum_{a_i \in A_i} x_{a_i} = 1 && \forall a_i \in A_i : \\
& && x_{a_i} \geqslant 0
\end{aligned}
$$
5: $\quad i' = \arg\max_{i \in T} v_i^*$
6: $\quad s_i^{cur}(a_i) = \begin{cases} x_{a_i}^* & \text{if } i = i' \\ s_i^{cur}(a_i) & \text{otherwise} \end{cases}$
7: **until** convergence **or** timeout
8: **return** s^{cur}

Iterated linear programming In Algorithm 2 we propose a method we call `IteratedLP` based on solving iteratively a Maxmin problem between 2 players (a member of the team and the adversary) by linear programming.

It works by maintaining a current solution s^{cur} which specifies a strategy for each team member. It is initialized (Line 1) with a starting solution $\hat{s}$ which, in principle, can prescribe an arbitrary set of strategies for the team (e.g., uniform randomizations). Then for each team member i (Line 3), we instantiate and solve the specified linear program (Line 4). The decision variables of this LP are v_i and, for each action a_i of player i, x_{a_i}. We maximize v_i subject to the upper bound given by the first constraint, where we assumed that the strategy of player i (relabeled with x to distinguish it) is a variable while the strategies of the other team members are constants, set to the associated value specified by the current solution. (Notice that, in the LP, a_j is the action that team member j plays in the team action profile a_T). The optimal solution of the LP is given by v_i^* and x^*, representing the Maxmin strategy of team member i once the strategies of teammates have been fixed to the current solution. Once the LP has been solved for each i, the algorithm updates the current solution in the following way (Line 6):

the strategy of the team member that obtained the best LP optimal solution is replaced with the corresponding strategy from the LP. This process continuously iterates until convergence or until some timeout is met. At each iteration of the algorithm, the value of the game increases (non–strictly) monotonically. We run it using multiple random restarts, i.e., generating a set of different initial assignments $\hat{s}$ (Line 1). Once convergence is achieved, we pass to the next random restart generating a new strategy profile for the team.

A crucial question is whether there are initializations that are better or worse than others. We can prove the following.

Theorem 5 *When the algorithm is initialized with a uniform strategy for every player, the worst–case approximation factor of Algorithm 2 is at least* $\frac{1}{m^{n-1}}$ *and at most* $\frac{1}{m^{n-2}}$. *When instead the algorithm is initialized with a pure strategy, the worst–case approximation factor is 0.*

We leave open the problem of studying how the worst–case approximation factor varies for other initializations.

Experimental evaluation

Experimental setting

Our experimental setting is based on instances of the `RandomGames` class generated by GAMUT (Nudelman et al. 2004). Specifically, once a game instance is generated, we extract the utility function of player 1 and assign it to all the team members. Furthermore, in each generated game instance, the payoffs are between 0 and 1. We use 100 game instances for each combination of n and m where $n \in \{3, 4, 5\}$ and m is as follows:

$$m = \begin{cases} 5 \text{ to } 40, & \text{step} = 5, & n = 3 \\ 50 \text{ to } 150, & \text{step} = 10, & n = 3 \\ 5 \text{ to } 50, & \text{step} = 5, & n = 4 \\ 5 \text{ to } 30, & \text{step} = 5, & n = 5 \end{cases}.$$

Algorithms are implemented in Python 2.7.6, adopting GUROBI 6.5.0 (Gurobi Optimization 2015) for linear mathematical programs, AMPL 20160310 (Fourer, Gay, and Kernighan 1989) and BARON 14.4.0 (Tawarmalani and Sahinidis 2005; Sahinidis 2014) for global optimization programs. We set a timeout of 60 minutes for the resolution of each instance. All the algorithms are executed on a UNIX computer with 2.33GHz CPU and 128 GB RAM.

Experimental results

Global optimization The average quality of the solutions is reported in Fig. 1 in terms of ratio between the lower (a.k.a. primal) bound and the upper (a.k.a. dual) bound returned by BARON once terminated. When BARON finds the optimal solution (up to an accuracy of 10^{-9}), the lower bound equals the upper bound achieving a ratio of 1. This happens with $n = 3$ up to $m = 15$ (except for some outliers), with $n \in \{4, 5\}$ and $m = 5$. For larger instances with $n = 3$ up to $m = 130$, with $n = 4$ up to $m = 45$ and with $n = 5$ up to $m = 20$, BARON returns an approximate solution with a ratio always larger than 0.7. With $n = 3$ and $m \in \{140, 150\}$, BARON returns a very high upper bound such that the ratio is close to zero. With larger instances, BARON does not run due to memory limits. Hence, BARON demonstrates to be an excellent tool to approximate the Team–maxmin equilibrium especially with $n = 3$ (note that, with $n = 3$ and $m = 130$, the number of outcomes is larger than 2 millions).

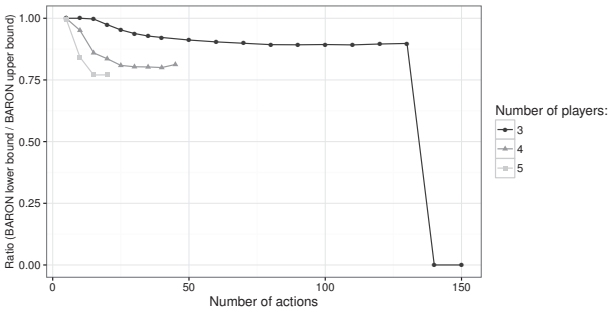
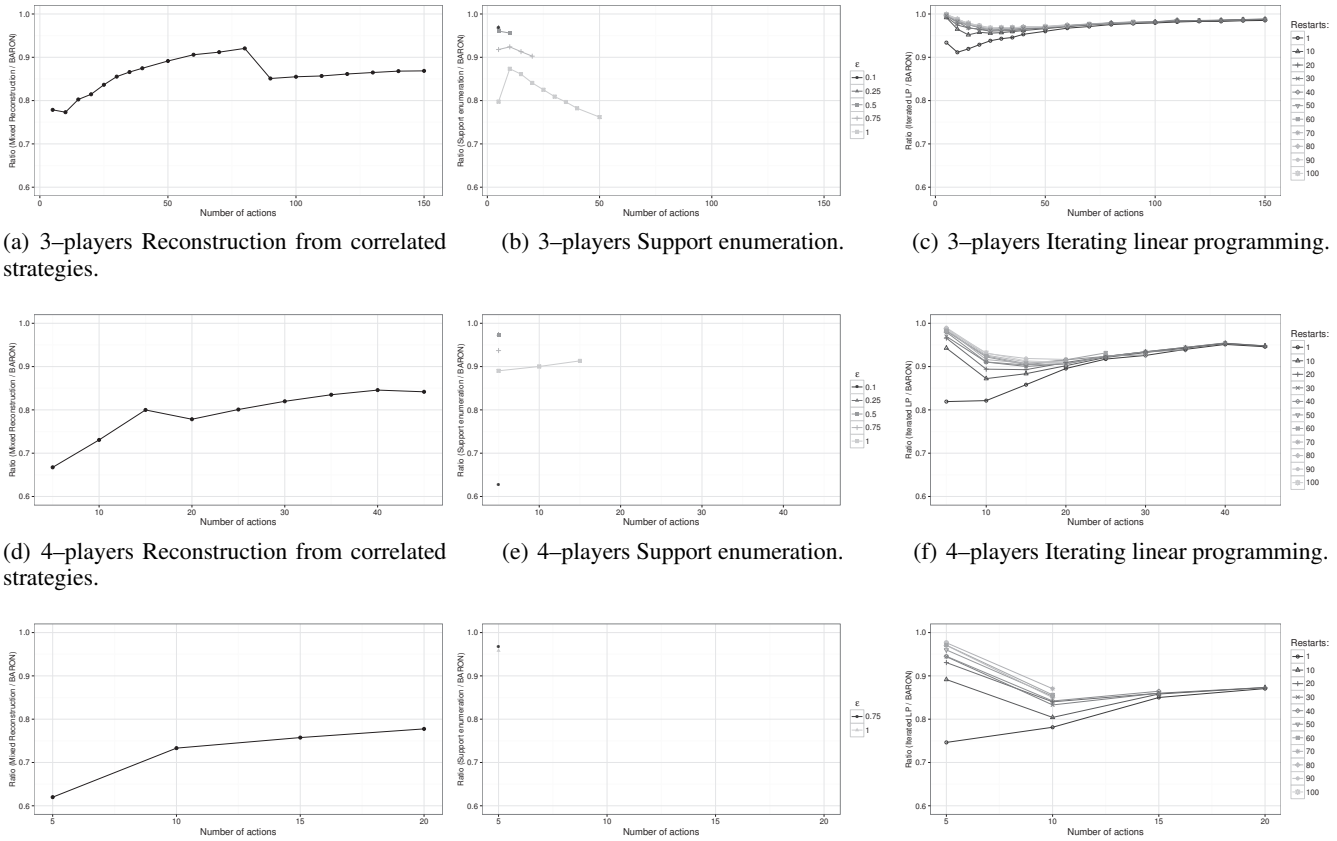


Figure 1: Average (empiric) approximation ratio (lower bound / upper bound) of the solutions returned by BARON.

Other algorithms We report in Fig. 2 the average performance of the other three algorithms described in the previous section in terms of ratio between team value of the strategy returned by each single algorithm and the lower bound returned by BARON. A ratio smaller than 1 means that the given algorithm provides a solution with a quality worse than the solution returned by BARON.

Let us focus on the reconstruction from correlated strategies. This algorithm solves all the instances of our experimental settings, including also the instances that BARON does not solve due to memory limits. However, the quality of the solutions is always worse than the solutions returned by BARON. More precisely, we notice that the ratio is always larger than 0.6 and, with $n = 3$, it is larger than 0.8. Hence, the quality of the solutions w.r.t. the upper bound of BARON is always larger than 0.5, thus achieving at least $1/2$ of the Team–maxmin value. As expected, the quality decreases as the number of players increases. Instead, surprisingly, the quality increases as the number of actions per player increases (we provide a motivation for that below).

Let us focus on the support enumeration algorithm. We report the performance of the algorithm for $\epsilon \in \{0.10, 0.25, 0.50, 0.75, 1.00\}$. As expected, the algorithm does not scale. Indeed, when ϵ is set < 1, the algorithm terminates only for $m \leqslant 20$ with $n = 3$, and only for $m = 5$ with $n \in \{4, 5\}$. This shows that the algorithm can be practically applied only when $\epsilon = 1$, but this amounts to not providing any theoretical bound (indeed, since all the payoffs are between 0 and 1, any strategy profile has an additive gap no larger than 1). However, even when $\epsilon = 1$, the algorithm terminates only for $m \leqslant 50$ with $n = 3$, for $m \leqslant 15$ with $n = 4$, and for $m = 5$ with $n = 5$, that is a strictly smaller subset of instances than the subset solved by BARON. The quality of the solutions returned by the algorithm is rather good, but it is always worse than the solutions returned by



(a) 3–players Reconstruction from correlated strategies.

(b) 3–players Support enumeration.

(c) 3–players Iterating linear programming.

(d) 4–players Reconstruction from correlated strategies.

(e) 4–players Support enumeration.

(f) 4–players Iterating linear programming.

(g) 5–players Reconstruction from correlated strategies.

(h) 5–players Support enumeration.

(i) 5–players Iterating linear programming.

Figure 2: Average approximation performance of the proposed algorithms w.r.t. global optimization.

BARON. Finally, notice that, differently from the previous algorithm, in the case of $n = 3$, the quality of the solution decreases as the number of action increases.

Let us focus on the iterated linear programming. We report the performance of the algorithm for a number of random restarts in $\{1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. This algorithm solves all the instances of our experimental settings, including also the instances that BARON does not solve due to memory limits. Also in this case, the quality of the solutions is always worse than the solutions returned by BARON. Notice that the ratio is very high and very close to 1 for $n = 3$. Interestingly, the number of restarts affects the solution quality essentially only when m is small. Obviously, when m is large, the number of restarts performed by the algorithm reduces, but, surprisingly, the solution quality increases and it is almost the same for every number of restarts. We observe that the number of restarts performed with the largest instances is 30 with $n \in \{3, 4\}$ and 10 with $n = 5$. This algorithm provides the best approximation w.r.t. the previous two algorithms.

Summary Global optimization (BARON) provides the best approximate solutions, but it does not solve all the instances of the experimental setting due to memory limits. The algorithm based on the iterated linear programming allows one to solve larger instances with a relatively small loss in terms of utility. Furthermore, approximating the Team–maxmin equilibrium gets easier as the number of actions per player increases. We observe that this may happen because, as the number of actions per player increases, the probability that the Team–maxmin equilibrium has a small support increases and small supported equilibria should be easier to be found.

Price of Uncorrelation Finally, we empirically evaluate POU in our experimental setting. We do that by calculating the ratio between the value of the Correlated–team maxmin equilibrium—computed exactly—and the lower bound retuned by BARON, which is the algorithm returning always the best approximation of the Team–maxmin equilibrium. This ratio is obviously an upper bound of the actual POU. In Fig. 3, we report the average ratio (for statistical significance, see (Basilico et al. 2016)). Surprisingly, the ratio is very close to 1 even for $n = 5$, while, we recall, the worst–case ratio is m^{n-2}. It can be observed that the ratio is monotonically increasing in n, while the dependency on m is not monotone: there is a maximum small values of m and then the ratio decreases as m increases. For instance, in 3–players

games, the ratio goes asymptotically to about 1.15, showing that the loss is very small empirically. Unexpectedly, this shows that, on average, the loss due to the inability for a team of correlating their strategies is rather small.

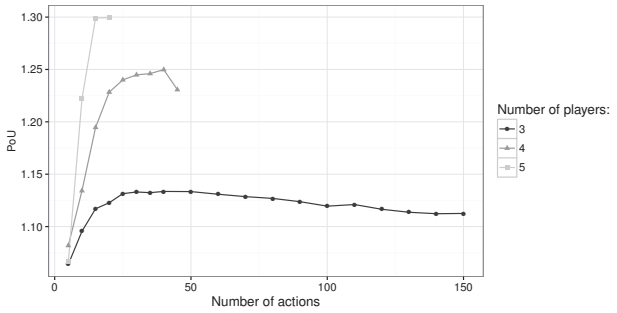


Figure 3: Empiric Price of Uncorrelation.

In (Basilico et al. 2016), we report additional experiments for other specific game classes assessing, analogously to what done in this section with `RandomGames`, empirical approximation performances and Price of Uncorrelation.

Conclusions

The Team–maxmin equilibrium is an important solution concept requiring deep algorithmic studies. In this work, we studied its efficiency w.r.t. Nash equilibrium and the Maxmin equilibrium with correlated team strategies. Moreover, we proposed algorithms to compute/approximate it, deriving theoretical guarantees and empirical evaluations.

In future, we will deal with Team–maxmin equilibrium in specific games like polymatrix games and congestion games.

References

Alpern, S., and Lim, W. S. 1998. The symmetric rendezvous-evasion game. *SIAM J CONTROL OPTIM* 36(3):948–959.

Andersen, G., and Conitzer, V. 2013. Fast equilibrium computation for infinitely repeated games. In *AAAI*, 53–59.

Ashlagi, I.; Monderer, D.; and Tennenholtz, M. 2008. On the value of correlation. *J ARTIF INTELL RES* 33:575–613.

Basilico, N.; Celli, A.; De Nittis, G.; and Gatti, N. 2016. Team-maxmin equilibrium: efficiency bounds and algorithms. *arXiv preprint arXiv:1611.06134*.

Borgs, C.; Chayes, J. T.; Immorlica, N.; Kalai, A. T.; Mirrokni, V. S.; and Papadimitriou, C. H. 2010. The myth of the folk theorem. *GAME ECON BEHAV* 70(1):34–43.

Bornstein, G.; Erev, I.; and Goren, H. 1994. The effect of repeated play in the IPG and IPD team games. *J CONFLICT RESOLUT* 38(4):690–707.

Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *ACM EC*, 82–90.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2009. The complexity of computing a Nash equilibrium. *SIAM J COMPUT* 39(1):195–259.

Deng, X.; Papadimitriou, C.; and Safra, S. 2002. On the complexity of equilibria. In *STOC*, 67–71.

Fourer, R.; Gay, D. M.; and Kernighan, B. 1989. Algorithms and model formulations in mathematical programming. Springer-Verlag New York, Inc. chapter AMPL: A Mathematical Programming Language, 150–151.

Gurobi Optimization, I. 2015. Gurobi optimizer reference manual.

Hansen, K. A.; Hansen, T. D.; Miltersen, P. B.; and Sørensen, T. B. 2008. Approximability and parameterized complexity of minmax values. In *WINE*, 684–695.

Jain, M.; Kardes, E.; Kiekintveld, C.; Ordóñez, F.; and Tambe, M. 2010. Security games with arbitrary schedules: A branch and price approach. In *AAAI*, 792–797.

Jiang, A. X.; Procaccia, A. D.; Qian, Y.; N.; and Tambe, M. 2013. Defender (mis)coordination in security games. In *IJCAI*, 199–206.

Kontogiannis, S. C., and Spirakis, P. G. 2008. Equilibrium points in fear of correlated threats. In *WINE*, 210–221.

Lim, W. S. 1997. A rendezvous-evasion game on discrete locations with joint randomization. *ADV APPL PROBAB* 1004–1017.

Lipton, R. J., and Young, N. E. 1994. Simple strategies for large zero-sum games with applications to complexity theory. In *STOC*, 734–740.

Moon, C., and Conitzer, V. 2016. Role assignment for game-theoretic cooperation. In *AAMAS*, 1413–1414.

Nash, J. 1951. Non-cooperative games. *ANN MATH* 286–295.

Neumaier, A.; Shcherbina, O.; Huyer, W.; and Vinkó, T. 2005. A comparison of complete global optimization solvers. *MATH PROGRAM* 103(2):335–356.

Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. *Algorithmic game theory*. Cambridge University Press Cambridge.

Nudelman, E.; Wortman, J.; Shoham, Y.; and Leyton-Brown, K. 2004. Run the gamut: A comprehensive approach to evaluating game-theoretic algorithms. In *AAMAS*, 880–887.

Palfrey, T. R., and Rosenthal, H. 1983. A strategic calculus of voting. *PUBLIC CHOICE* 41(1):7–53.

Sahinidis, N. V. 2014. *BARON 14.4.0: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual.

Solan, E. 2000. Absorbing team games. *GAME ECON BEHAV* 31(2):245–261.

Tawarmalani, M., and Sahinidis, N. V. 2005. A polyhedral branch-and-cut approach to global optimization. *MATH PROGRAM* 103:225–249.

von Stengel, B., and Koller, D. 1997. Team-maxmin equilibria. *GAME ECON BEHAV* 21(1):309–321.

Wang, X., and Sandholm, T. 2002. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. In *NIPS*, 1571–1578.