Boundedly complex Turing machines play the repeated prisoner's dilemma : some results

Guillaume LACÔTE[1]

Preliminary discussion paper - please do not circulate.

Abstract : Bounding the complexity of strategies in a repeated game tremendously affects the set of Nash equilibria : this has been studied extensively when the complexity of a strategy is defined in terms of finite state automata or of bounded recall. We consider the more general model of computable strategies (those which can be implemented by a Turing machine). To define complexity we depart from structural parameters like number of states and symbols and consider the Kolmogorov-Smirnov complexity of a strategy : it is its shortest constructive description (provided a fixed language).

We consider the finitely repeated Prisoner's Dilemma where each player has to chose a pure strategy of bounded Kolmogorov complexity. We discuss on the length of the game and the respective bounds on complexity to identify wether cooperation is a Nash equilibrium. We show that cooperation is indeed sustainable if and only if the complexity of at least one player is significantly smaller than the length of the game.

Keywords : Repeated games, bounded rationality, Turing macines, Kolmogorov complexity

1 Introduction

A number of arguments suggest that economic agents do not enjoy the unbounded computational power hypothesized by well-known results of game theory. A wide litterature has shown that these results are deeply affected if this assumption is relaxed (see [1], [2] or [7] for surveys). For example in the finitely repeated prisoner's dilemma played by finite state automata of fixed number of states, cooperation is a path of Nash equilibrium provided that at least one automaton has sufficiently few states (see [5] for a detailed study).

In this paper we consider the more sophisticated model of Turing machines and focus on the repeated Prisoner's Dilemma played by to machines of bounded complexity. It is arguable that the class of Turing machines encompasses all effective pure strategies as any effectively constructible function is implemented by at least one Turing machine. Other functions can not possibly be implemented by a human even with the help of any deterministic computational device.

We consider the definition of complexity suggested by Kolmogorov and Smirnov. We discuss on the relative complexity bounds of each player and on the size of the game to determine if some kind of cooperation is a path of Nash equilibrium in pure strategies. We show that it is the case if, and only if the complexity of at least one player is significantly smaller than the length of the game.

[1]ENSAE. 3 av. P. Larousse, F-92240 Malakoff FRANCE. Mail: Games@glacote.com

This paper intends to present preliminary results. The next section recalls standard concepts of game theory and of computational models. Readers familiar with repeated games or computational models may want to skip this section. Section three presents the specific setting and the main result, which is proved in section four.

2 The setting

2.1 The repeated prisoner's dilemma

We consider the two players one-stage Prisoner's Dilemma $G = (\{0,1\} \times \{0,1\}, r)$ where each player plays either 0 (he *defects*) or 1 (he *cooperates*) and with payoff matrix

$$r = \begin{array}{|c|cc|} \hline & 0 & 1 \\ \hline 0 & (1,1) & (4,0) \\ 1 & (0,4) & (3,3) \\ \hline \end{array}$$

For example if player one plays 0 and player two plays 1 then player one gets a payoff of 4 while player two receives 0.

For $T \in \mathbb{N}^*$ we define G_T be the *finitely repeated game* induced by the repetition of G T times, and G_∞ the *infinitely repeated game* . First for $t \in \mathbb{N}^*$ let $H_t = (\{0,1\}^2)^{t-1}$ be the set of all possible *histories* at stage t, with the convention $(\{0,1\}^2)^0 = \{\emptyset\}$ where $\emptyset$ denotes the empty history. Let $\mathcal{H}_T = \bigcup_{t\in[1,T]} (\{0,1\}^2)^{t-1}$ the set of histories *up to* stage T, $H_* = \bigcup_{t\in\mathbb{N}^*} (\{0,1\}^2)^{t-1}$ the set of all finite histories. $H_\infty = (\{0,1\}^2)^{\mathbb{N}^*}$ the set of infinite histories.

A *pure strategy* for player i in G_T is any $\sigma^i : H_T \to \{0,1\}$; let $S_T^i = (\{0,1\})^{H_T}$ the set of (pure) strategies. Similarly a strategy for player i in G_∞ is any $\sigma^i : H_\infty \to \{0,1\}$ and let $S_\infty^i = (\{0,1\})^{H_\infty}$ be the set of infinite strategies.

A profile of strategy $\sigma = (\sigma^1, \sigma^2)$ in G_T (respectively in G_∞) induces a *play* $\omega(\sigma)$ in H_T (respectively in H_∞) defined recursively in the following way : first let $\omega(\sigma)_1 = (\sigma_0^1(\emptyset), \sigma_0^2(\emptyset))$, and then set

$$\omega(\sigma)_t = \left(\sigma_{t-1}^1\left(\omega(\sigma)_{t-1}\right), \sigma_{t-1}^2\left(\omega(\sigma)_{t-1}\right)\right)$$

for any $t \in [2,T]$ (respectively for any $t \geq 2$). Two strategies σ^i and τ^i for player i are *equivalent* if $\forall \sigma^{-i}\, \omega(\sigma^i, \sigma^{-i}) = \omega(\tau^i, \sigma^{-i})$. A strategy σ^i of player i is *compatible* with the play $\omega = ((a_1^1, a_1^2), \ldots, (a_t^1, a_t^2))$ (which will be denoted by $\sigma^i \rightsquigarrow \omega$) if

$$\forall l \in [1,t]\,,\ \sigma^i\left((a_1^1, a_1^2), \ldots, (a_{l-1}^1, a_{l-1}^2)\right) = a_l^i$$

Finally let $r_T(\sigma) = \frac{1}{T}\sum_{t=1}^T r\left(\omega(\sigma)_t\right)$ be the *payoff* in G_T, and $r_\infty(\sigma) = \limsup_{T\to+\infty} \frac{1}{T}\sum_{t=1}^T r\left(\omega(\sigma)_t\right)$ the payoff in G_∞. $r_\infty(\sigma)$ may not be finite.

2.2 Turing machines and Kolmogorov complexity

A Turing machine is a computational device introduced by Alan Turing. It consists on a deterministic finite state automaton with one infinite tape. The tape has cell boundaries

and on each cell symbols from a given finite alphabet $\mathcal{A}$ might be stored ; cells can be read or written one at a time. A finite sequence of symbols is written initially on the tape. The machine then proceeds a succession of computational steps : it transitions from one state to another according to the current state and symbol, and may also write to the tape and shift it to access the adjacent cell. The machine may eventually reach a dedicated state called the "finish" state in which case the sequence of symbols written on the tape is called the "output sequence". Although this description looks ground-level any modern computer could be simulated by a Turing machine. Moreover Turing showed among other results that the set of recursive functions as defined by Church is actually equal to the set of functions that can be computed by Turing machines. This means that any constructive [2] function or program can be implemented by a Turing machine. This justifies the use of Turing machines as an abstract computantional model. A complete study is done in [4].

Let $\mathcal{A}$ be the finite set of tape symbols. Let $|A|$ be its cardinality. Let $\mathcal{A}^* = \bigcup_{t\in\mathbb{N}} \mathcal{A}^t$ the set of words, *i.e.* of finite sequence of symbols. For $\omega \in \mathcal{A}^t \subset \mathcal{A}^*$ we note $|\omega| = t$ its length. The concatenation of two words $x, y \in \mathcal{A}^*$ is denoted simply by $xy = x_1, \ldots, x_{|x|}, y_1, \ldots, y_{|y|}$.

Each Turing machine runs a specific program which may or may not ever terminate its computation. For any Turing machine M and any finite input sequence $\omega \in \mathcal{A}^*$ we note $M(\omega) \in \mathcal{A}^*$ the (possibly empty) finite [3] sequence of symbols written on the output tape if the computation ever ends, or $\perp$ if the computation never ends. Note that the actual time required to compute $M(\omega)$ does not matter provided it is finite.

Any word $\omega \in \Sigma^*$ on any non-empty finite alphabet Σ is uniquely encoded to a number through the following mapping :

$$\langle\cdot\rangle_\Sigma : \begin{pmatrix} \Sigma^* & \rightarrow & \mathbb{N} \\ \omega & \mapsto & |\Sigma|^{|\omega|+1} - 1 + \sum_{t=1}^{|\omega|} |\Sigma|^t * ord(\omega_t) \end{pmatrix}$$

where $ord(l)$ denotes the order of the letter l in Σ. $\langle\cdot\rangle_\Sigma$ is bijective and implements and enumeration of all finite words over Σ. Moreover $\langle 0\rangle_\Sigma = 0$ and $\langle\cdot\rangle_\Sigma$ enumerates words in increasing lengths : $\langle\omega\rangle_\Sigma < \langle\omega'\rangle_\Sigma \Rightarrow |\omega| \leq |\omega'|$. In the case where $\Sigma = \{0, 1\}$ we simply have

$$\langle\cdot\rangle_{\{0,1\}} : \begin{pmatrix} \{0,1\}^* & \rightarrow & \mathbb{N} \\ \omega & \mapsto & 2^{|\omega|+1} - 1 + \sum_{t=1}^{|\omega|} 2^t * \omega_t \end{pmatrix}$$

Thus $\langle\cdot\rangle_{\{0,1\}}$ enumerates the words $\emptyset, 0, 1, 00, 01, 000, \ldots$ in this order.

For the sake of simplicity we now assume that each Turing machine takes a number in $\mathbb{N}$ as input and either computes forver or outputs a number in $\mathbb{N}$:

$$\tilde{M} : \begin{pmatrix} \mathbb{N} & \rightarrow & \mathbb{N}\bigcup\{\infty\} \\ n & \mapsto & \begin{cases} \left\langle M\left([x]_{\{0,1\}}\right)\right\rangle_{\{0,1\}} & \text{if } \phi_i\left([x]_{\{0,1\}}\right) \neq \perp \\ \infty & \text{otherwise} \end{cases} \end{pmatrix}$$

[2] This includes arbitrarly large closed-formulae or any computer program. This specifically excludes any definition which relies on the existential symbol : "find the zero of this 5-th degree polynom" is not Turing-computable. "give an approximation of a zero of this 5-th degree polynom with 3 significant digits" is Turing-computable (through the constructive Newton-Raphson method for example).

[3] Since the initial sequence on the tape is finite, a machine could not write an infinite sequence in a finite number of steps.

Since the description of a Turing machine is finite the set of Turing machines is countable. We call $[i]_{\{0,1\}}$ a *description* of the Turing machine ϕ_i. For any enumeration $(\phi_i)_{i\in\mathbb{N}}$ of Turing machines we define its associated *Kolmogorov-Smirnov* complexity for any $x \in \mathbb{N}$ as the shortest description of a Turing machine which outputs x on empty input :

$$K_\phi : \left(\begin{array}{ccc} \mathbb{N} & \to & \mathbb{N} \\ x & \mapsto & \min\left\{ |[n]_{\mathcal{A}}| \,/ \phi_n(\emptyset) = \langle x \rangle_{\mathcal{A}} \right\} \end{array} \right)$$

It is the length of the "shortest description" or "most-compressed representation" of x ; For any $x \in \mathbb{N}$ we note $\overline{x} = \min\{n \in \mathbb{N} \,/\, \phi_n(\emptyset) = \langle x \rangle_{\mathcal{A}}\}$ the smallest number of a Turing description of x. Note that since $\langle \cdot \rangle_{\mathcal{A}}$ enumerates words in increasing length this ensures that $|[\overline{x}]_{\mathcal{A}}| = K_\phi(x)$: $\overline{x}$ is a shortest description of x.

These definitions depend on the particular enumeration ϕ of Turing machines. However for any other enumeration $(\psi_i)_{i\in\mathbb{N}}$ there exists a fixed $c_{\phi,\psi} \in \mathbb{N}$ such that $\forall x \in \mathbb{N}, |K_\phi(x) - K_\psi(x)| \leq c_{\phi,\psi}$. Thus Kolmogorov complexity is usually referred to an universal quantity "up to a fixed constant" and deemed useful for asymptotic results.

As will be clear later on such asymptotic results are inadequate in a game-theoretic context ; instead we shall fix a particular enumeration of Turing machines which suits our needs well.

A wide litterature is devoted to exhibiting the properties Kolmogorov complexity ; this is outside the scope of this paper. We recall two important properties though : the first is that Kolmogorov complexity is not computable : there exists no Turing machine which outputs $K_\phi(x)$ for any input $x \in \mathbb{N}$. In particular there exists no closed-formula expressing K_ϕ.

The second property is the the following approximation of K_ϕ. Let $p = |\mathcal{A}|$ be the number of symbols of our model of Turing machines. For $E \subset \mathbb{N}$ let $\underline{\mathrm{d}}\,(E) = \liminf_{N\to+\infty} \frac{1}{N} |E \bigcup [1, N]|$ be its *lower density* in $\mathbb{N}$.

Proposition 1 *(i)*

$$\exists m \in \mathbb{N} \,/\, \forall x \in \mathbb{N}^*, K_u(x) \leq \lceil \log_p(x) \rceil + m$$

(ii)

$$\begin{array}{c} \forall f : \mathbb{N} \to \mathbb{N}, f(\infty) = \infty \,,\, \forall \epsilon > 0 \,,\, \exists E_{f,\epsilon} \subset \mathbb{N}^* \,/\, \underline{d}\,(E_{f,\epsilon}) \geq 1 - \epsilon \\ et\ \forall x \in E_{f,\epsilon},\ K_u(x) \geq \lceil \log_p(x) \rceil - f\left(\lceil \log_p(x) \rceil \right) - 1 \end{array}$$

The complexity of most numbers is roughly equal to the length of their representation. An elementary proof is provided in appendices.

In this paper we face two main issues. Firstly the notion of Kolmogorov complexity is widely distinct from the complexity in speed or space. Define instead the complexity of a machine as the number of processing steps required to produce its result. Then there exists problems whose answer is "easy" to verify but lengthy to compute. [4] This property is widely used in the litterature to elaborate path of plays which are complex for one player but not for the other. [5] However in terms of Kolmogorov complexity there is no "simple" question whose answer

[4] This is the case with any NP-complete problem, under the standard assumption that $P \neq NP$.

[5] Devise a path of play composed of cycles where player one plays the product $p * q$ of two large prime numbers and checks that player two correctly plays p then q in the next cycle.

is computable and "complex" (since the very question is a computable representation of the answer). We thus need to rely on other tools to build an equilibrium path ; it appears that there exists simple such tools.

The second issue deals with the "up to a constant" factor. Most results in the litterature state properties of the Kolmogorov complexity where (in)equalities hold up to a fixed constant independant of all other variables. Proposition 1-(i) is an example of such a result. In a game-theoretic context however this uncertainty is unacceptable : each player needs to ascertain that *no* machine of a given complexity can implement some defecting strategy. Let us consider the $\mathcal{D}_t^1$ strategy which cooperates up to stage $t-1$ after which it always defects. [6] An intuitive result states that its complexity is "roughly" that of $t : \exists m \in \mathbb{N}\,, \forall t \in \mathbb{N}^*\,, |K_\phi\left(\mathcal{D}_t^1\right) - K_\phi(t)| \leq m$. We may want to choose t so as to saturate the complexity of player one - so that player one may not implement $\mathcal{D}_{t-1}^1$ but may still implement $\mathcal{D}_t^1$ for example. However both also have "roughly" the same complexity : $\exists m' \in \mathbb{N}\,, \forall t \in \mathbb{N}^*\,, \left|K_\phi\left(\mathcal{D}_t^1\right) - K_\phi\left(\mathcal{D}_{t-1}^1\right)\right| \leq m'$. Thus a sharper computation of the Kolmogorov complexity of $\mathcal{D}_t^1$ with respect to the complexity of t itself is required. We address this issue by devising a specific enumeration of Turing machines tailored to simplify the link between the complexity of a strategy depending on some parameters and the complexity of the parameters themselves.

3 Main results

3.1 Kolmogorv Complexity of a strategy

We consider an enumeration $(\phi_i)_{i\in\mathbb{N}}$ of Turing machines which will be defined later.

For $i \in \{1,2\}$ and $T \in \mathbb{N}\bigcup\{\infty\}$ we say that the Turing machine ϕ_x *implements* the strategy $s^i \in S_T^i$ for player i in G_T if

$$\forall h \in \mathcal{H}_*, s^i \rightsquigarrow h \;\Rightarrow\; \phi_x\left(\langle h\rangle_{\{0,1\}}\right) = \left\langle s^i(h)\right\rangle_{\{0,1\}}$$

i.e. ϕ_x computes the action played by s^i after any compatible history. This implies in particular that the computation $\phi_x(\langle h\rangle_{\{0,1\}})$ ends in finite time. There exist several Turing machines which implement the same strategy.

We define

$$K_\phi(s^i) = \min\{K_\phi(x) \,/\, \phi_x \text{ implements } s^i\}$$

the Kolmogorov complexity of the strategy s^i. In particular $K_\phi(s^i) = +\infty$ if s^i can be implemented by no Turing machine. For $k \in \mathbb{N}^*$ let $\Sigma_T^i(k) = \{s^i \in S_T^i \,/\, K_\phi(s^i) \leq k\}$ be the set of strategies for player i in G_T which can be implemented by a Turing machine of complexity no more than k. $\Sigma_T^i(k)$ might be empty for small k and we have $k \leq l \Rightarrow \Sigma_T^i(k) \subset \Sigma_T^i(l)$. Let $\Sigma_T^i(\infty) = S_T^i$.

For $(k,l,T) \in \left(\mathbb{N}^*\bigcup\{\infty\}\right)^3$ let $G_T(k,l) = (\Sigma_T^1(k) \times \Sigma_T^2(l), r_T)$ be the (in)finitely repeated prisoner's dilemma where player one choses a strategy which can be implemented by a Turing machine of complexity no more than k and player two no more than l.

[6] It is defined formally in the next section

If $T = k = l = \infty$ a folk theorem ensures that $(1,1),\ldots$ is a path of Nash equilibrium. If $k = l = \infty$ but $T < \infty$ on the other hand constant defection of both player is the only Nash equilibrium. We consider the following question : under what conditions on (T,k,l) is some form of cooperation sustainable in $G_T(k,l)$?

3.2 Specific strategies and programs

We first define some standard strategies. For $t \in \mathbb{N}$ let

$$\mathcal{D}_t^i : \left(\begin{array}{ccc} \mathcal{H}_T & \to & \{0,1\} \\ h & \mapsto & \left\{ \begin{array}{ll} 1 & \text{if } h = (1,1),\ldots,(1,1) \text{ and } |h| \neq t \\ 0 & \text{otherwise} \end{array} \right. \end{array} \right)$$

$\mathcal{D}_1^i$ is the strategy of perpetual defection for player i. $\mathcal{D}_0^i$ is the strategy which cooperates as long as the other player does and defects forever if he ever defects once. For $t \geq 2$, $\mathcal{D}_t^i$ is the strategy which cooperates up to stage $t-1$ inclusive as long as the other player does and defects forever afterwards.

For $t \leq T$ and $x \in \mathcal{H}_*$ let

$$\mathcal{ED}_{x,:}^i \left(\begin{array}{ccc} \mathcal{H}_T & \to & \{0,1\} \\ h & \mapsto & \left\{ \begin{array}{ll} x_{n+1} & \text{if } n = |h| < |x| \text{ and } h = (x_1,x_1),\ldots,(x_n,x_n) \\ 1 & \text{if } |x| \leq |h| < t \text{ and } h = x,(1,1),\ldots,(1,1) \\ 0 & \text{otherwise} \end{array} \right. \end{array} \right)$$

Let $\mathcal{E}_x^i = \mathcal{ED}_{x,\infty}^i$.

$\mathcal{E}_x^i$ is the strategy for player i which plays $x\,1\cdots1$ as long as the other player does so (and defects forever otherwise). $\mathcal{ED}_{x,t}^i$ does the same except that it always defects starting at stage $|x|+t$.

We now define programs which will be useful later. For reasons apparent in hte next paragraph on we consider octal symbols [7] and set $\mathcal{A} = \{0,\ldots,7\}$. Let $(\psi_i)_{i\in\mathbb{N}}$ be some arbitrary enumeration of Turing machines. Let ψ_u be some universal Turing machine. [8] Now for $p \in \mathbb{N}$ we consider the following program :

$CountDefection_p$:
  $h \leftarrow \emptyset$
  $t \leftarrow 0$
  while $\psi_u(p,h) = 1$ do
    $h \leftarrow h,(1,1)$
    $t \leftarrow t+1$
  return t

[7] See the enumeration ϕ of Turing machines below. A side effect is that any length or Kolmogorov complexity translates easily to its binary counterpart through a division by three.

[8] That is $\exists u \in \mathbb{N} \,/\, \forall i \in \mathbb{N}, \forall x \in \mathbb{N}, \psi_u(i,x) = \psi_i(x)$.

This program takes no input and looks for the first stage at which ψ_p would defect from $(1,1),\dots,(1,1)$, if any such stage exists. Moreover it never ends its computation if ψ_p is not an implementation of a strategy compatible with $x,(1,1),\dots,(1,1)$, or if it never defects from it before the end of the game. On the other hand for $t \geq 1$ if ψ_k implements $\mathcal{D}^i_t$ then $CountDefection_k$ computes in finite time and returns t.

We now consider the following program :

$GetPathAndDefection_{p,T}$:
(First build the whole history up to stage T)
$h \leftarrow \emptyset$
$t \leftarrow 0$
while $t <= T$ do
$a^1 \leftarrow \phi_u(p,h)$ $\quad$ $h \leftarrow h,(a^1,a^1)$
$t \leftarrow t+1$
(Here $h = \left(\begin{array}{c|c|c|c} x & 0 & 1\cdots1 & 0\cdots0 \\ x & 0 & 1\cdots1 & 0\cdots0 \end{array}\right)$ provided ϕ_p is an impemention of some $\mathcal{ED}^i_{x,t}$)
(Now extract t)
$b \leftarrow T$
while $h^1_b = 0$ do
$b \leftarrow b-1$
$a \leftarrow b$
while $h^1_a = 1$ do
$a \leftarrow a-1$
$t \leftarrow b-a$
(Eventually extract x)
$x \leftarrow (h_1,\dots,h_{a-1})$
if $t=0$ then return x otherwise return (x,t)

This program (which takes no input) simulates ψ_p and builds the whole history up to stage T from which it extracts t and x. [9] If ψ_l implements $\mathcal{ED}^i_{x,t}$ for some $t \geq 1$ and $x \in \mathcal{H}_*$ then $GetPathAndDefection_{l,T}$ computes in finite time and returns (x,t). Similarly if ψ_l implements $\mathcal{E}^i_x$ for some $x \in \mathcal{H}_*$ then $GetPathAndDefection_{l,T}$ computes in finite time and returns x. It may not ever terminate its computation for some other ψ_l.

We now fix a specific enumeration of Turing machines :
- $\langle 0\rangle_{\mathcal{A}}$ is the number of some implementation of the constant defection $\mathcal{D}^1_1$.
- $\langle 1\rangle_{\mathcal{A}}$ is the number of some implementation of the cooperative strategy $\mathcal{D}^1_0$.
- For $t \in \mathbb{N}^*$ let $\langle 2\overline{t}\rangle_{\mathcal{A}}$ be the number of some implementation of $\mathcal{D}^1_t$.
- For $x \in \mathcal{H}_*$ let $\langle 3\overline{x}\rangle_{\mathcal{A}}$ be the number of some implementation of $\mathcal{E}^1_x$.
- For $x \in \mathcal{H}_*$ and $t \in \mathbb{N}^*$ let $\left\langle 4\overline{(x,t)}\right\rangle_{\mathcal{A}}$ be the number of some implementation of $\mathcal{ED}^1_{x,t}$.

Note that the implementation of any of the aforementioned strategies for player one is also a valid implementation for player two.

[9] This works because $(x,t) \mapsto \left(x\,0\,\underbrace{1\cdots1}_{t}0\cdots0\right)$ is injective. This is why we need to insert a 0 between x and $1\cdots1$.

– For $p \in \mathbb{N}^*$ let $\langle 5\overline{p} \rangle_{\mathcal{A}}$ be the number of some implementation of $CountDefection_p$.
– For $p \in \mathbb{N}^*$ let $\langle 6\overline{p} \rangle_{\mathcal{A}}$ be the number of some implementation of $GetPathAndDefection_{p,T}$. Note that this numbering implicitely depends on the length T of the game.
– For $n \in \mathbb{N}$ let $\langle 7\overline{n} \rangle_{\mathcal{A}}$ be the number of the n-th Turing machine ψ_n in the enumeration ψ.

This ensures that $(\psi_n)_{n\in\mathbb{N}}$ is a (non-injective) exhaustive enumeration of Turing machines (since $\forall n, \psi_n = \phi_{\langle 7\overline{n} \rangle_{\mathcal{A}}}$). We have explicitely reordered some of them so as to ensure that we know their Kolmogorov complexity. [10] In particular we have $K_\phi(\mathcal{D}_0^i) = K_\phi(\mathcal{D}_1^i) = 1$.

3.3 Main proposition

Proposition 2 *For most $T \in \mathbb{N}^*$ in $G_T(k,l)$ some form of cooperation is sustainable if, and only if the complexity of at least one player is smaller than the length of the game. In particular there exists a path of Nash equilibrium with at least $(T-k)^+$ consecutive stages of cooperation.*

That is, if the complexity of at least one player (assume the first) is smaller than the length T of the game then $\left(\begin{array}{c|c|c} x & 1\cdots 1 & y \\ u & 1\cdots 1 & v \end{array}\right)$ is a path of play of Nash equilibrium for some x, y, u, v. Conversely if the complexity of both players is larger than the length of the game then constant defection is the only path of Nash equilibrium. Note that the enumeration ϕ of Turing machines implicitely depends on T (since the code $\langle 6\overline{p} \rangle_{\mathcal{A}}$ doesn't include T) ; of course it does not depend on k nor on l.

4 Proofs

We discuss on T first, and then on k, l.

If $T = +\infty$ we show that $(\mathcal{D}_0^1, \mathcal{D}_0^2)$ is a Nash equilibrium. First note that $K_\phi(\mathcal{D}_0^1) = K_\phi(\mathcal{D}_0^2) = 1 \leq \min k, l$ such that both strategies can be implemented. Moreover the very structure of r ensures that there is no profitable single-stage deviation from $(1,1), \ldots, (1,1)$. Hence no player has any profitable deviation (whatever the complexity to implement it) and $(\mathcal{D}_0^1, \mathcal{D}_0^2)$ is a Nash equilibrium.

We now assume that $T < +\infty$.

Lemma 1 *For $t \geq 2$, $K_\phi(\mathcal{D}_t^i) = K_\phi(t)$.*

Lemma 1 states that to cooperate but to defect at stage t is of the same complexity as to write down the number t itself. This would be intuitively true up to a uniform constant for any enumeration of Turing machines ; our enumeration ensures that the constant is zero.

We now discuss on k, l.

If $k < K_\phi(T)$ and $l < K_\phi(T)$ then both players are so restricted that they can not even "count" up to the length T of the game. We show that the cooperative profile $(\mathcal{D}_0^1, \mathcal{D}_0^2)$ is a Nash equilibrium. The payoff matrix ensures that assuming that player two plays $\mathcal{D}_0^2$ player one has no profitable deviation before the last stage T of the game (whatever the complexity

[10] $\overline{x}$ is the smallest description of x among the enumeration $(\psi_i)_{i\in\mathbb{N}}$. It is straightforward that the smallest description of x among $(\phi_i)_{i\in\mathbb{N}}$ is equal or at most one bit shorter.

of implementing such deviation). This is a fundamental property of the prisoner's dilemma. Now consider the strategy $\mathcal{D}_T^1$ for player one which implements this profitable deviation. By lemma 1 it is of complexity $K_\phi(T) > k$. Thus player one can not implement it. Hence player one has no profitable deviation he could implement with a complexity no higher than k. The conclusion follows by symetry.

If $k < K_\phi(T-1)$ and $l \geq K_\phi(T)$ then player one is so restricted that he can not even "count" up to the length T of the game but player two can do it. It follows that $(\mathcal{D}_0^1, \mathcal{D}_T^2)$ is a Nash equilibrium : both can be implemented by hypothesis. Moreover player two has no profitable deviation while player one can not implement his only profitable deviation $\mathcal{D}_{T-1}^1$.

We do not elaborate during the discussion on wether $K_\phi(T-1) \leq K_\phi(T)$ or not. [11]

If $T < k \leq l$ then both players are basically unrestricted. This is because for any full sequence of actions $x \in \mathcal{H}_T$ we have $K_\phi(\mathcal{E}_x^i) = K_\phi(x) \leq |x| = T < k \leq l$. Thus each player can implement any (oblivious) sequence of actions. It is a basic result however that $\underbrace{(0,0),\ldots,(0,0)}_{T}$ is the only path without a profitable deviation. Since any profitable deviation can be implemented in $G_T(k,l)$ it is the only path of Nash equilibrium.

If $K_\phi(T) < k < T < l$ then player one is effectively restricted in the set of sequences of actions that he can implement whereas player two is not. We rely on the following lemmas :

Lemma 2 *For any $x \in \mathcal{H}_*$, $K_\phi\left(\mathcal{E}_{x0}^i\right) = K_\phi(x0)$.*

This lemma states that playing so as to ensure the path $\left(\begin{array}{c|cc} x & 0 & 1\cdots1 \\ x & 0 & 1\cdots1 \end{array}\right)$ is of the same complexity as writing down x itself.

Lemma 3 *For any $x \in \mathcal{H}_*$, $K_\phi\left(\mathcal{E}\mathcal{D}_{x0,t}^i\right) = K_\phi(x0,t)$.*

This lemma extends lemma 2 in the case of constant defection from stage $|x|+t$ onwards : this is of the same complexity as writing down x and t themselves. Both lemmas are once again straitghforward up to a fixed constant for any enumeration of Turing machines. The specific enumeration ϕ ensures that the constant can be set to zero.

Now let $t \in [k, T-2]$ and $x \in \mathcal{H}_{t-1}$ such that $K_\phi(x0) = k$. [12] We show that $(\mathcal{E}_{x0}^1, \mathcal{E}\mathcal{D}_{x0,T-t}^2)$ is a Nash equilibrium. The corresponding path of play is $\left(\begin{array}{c|cc|c} x & 0 & 1\cdots1 & 1 \\ x & 0 & 1\cdots1 & 0 \end{array}\right)$. First note that $K_\phi(\mathcal{E}_{x0}^1) = K_\phi(x0) = k$ and $K_\phi(\mathcal{E}\mathcal{D}_{x0,T-t}^2) \leq K_\psi(x\,0\,1\cdots1\,0) + 1 \leq T+1 \leq l$ so that both strategies can effectively be implemented. First note that player two has no profitable deviation (whatever the complexity to implement it). On the other hand the only profitable

[11] It may happen that $K_\phi(T-1) > K_\phi(T)$ in which case our discussion is incomplete. However this occurs for a "negligable proportion" of numbers $T : m : x \mapsto \inf\{K_\phi(y)/y \geq x\}$ goes to infinity slower than any other computable function which goes to infinity.

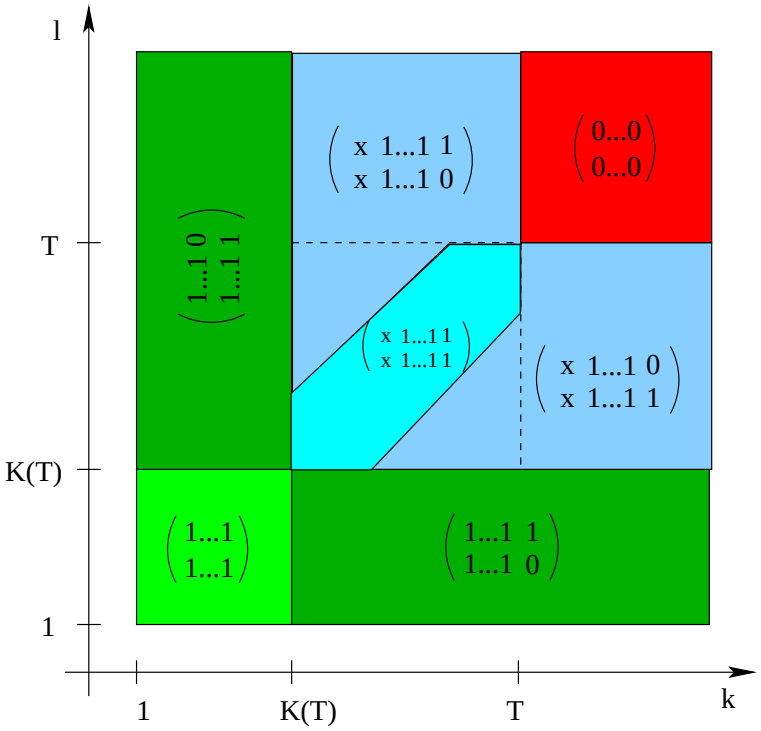
[12] We do not elaborate on the existence of such an x. Basically a counting argument ensures that from any $k' \in \mathbb{N}^*$ there exists $y \in \mathcal{H}_{k'}$ such that $K_\phi(y) = k'$.

deviation for player one is to defect at stages T and/or $T-1$. [13] However by lemma 3 the complexity for him to do so is at least $K_\phi(x)+1 > k$. Hence neither player can implement any profitable deviation. The fact that player one still cooperates at the last stage (while player two doesn't) comes from the fact that he is limited in complexity (while player two isn't).

Note that x still remains to be defined; by a simple couting argument there exists at least one x of length $k-1$ such that $x0$ is of complexity k. Longer choices for x of complexity k may be chosen; note however that since the ratio of cooperations and defections inside x is directly linked to k [14] all different choices will lead to more-or-less the same average payoff. [15]

If $K_\phi(T) < k \leq l < T$ then both players are effectively restricted in the set of sequences of actions that they can implement. Now let $t \in [k, T-2]$ and $x \in \mathcal{H}_t$ such that $K_\phi(x) = k$. If $l \geq k + K_\phi(T-t)$ the same arguments as above show that $(\mathcal{E}^1_x, \mathcal{ED}^2_{x,T-t})$ is a Nash equilibrium and the path of play is again $\left(\begin{array}{c|c|c} x & 1\cdots1 & 1 \\ x & 1\cdots1 & 0 \end{array}\right)$. Otherwise $(\mathcal{E}^1_x, \mathcal{E}^2_x)$ is a Nash equilibrium. and the path of play is $\left(\begin{array}{c|c|c} x & 1\cdots1 & 1 \\ x & 1\cdots1 & 1 \end{array}\right)$.

We summarize the path of plays in the following way :



[13] Note that $|x| \leq T-2$ by construction so that there is at least one cooperation pair $(1,1)$ after x in the play.

[14] This is because for any binary sequence $x \in \{0,1\}^t$ we have $K_\phi(x) \approx tH(\widehat{\pi})$ where $\widehat{\pi}$ is the empirical distribution of zeros and H denotes the entropy.

[15] Let n_x be the number of zeros in $x \in \mathcal{H}_t$ for $t \geq 2$. We have $K_\phi(x) \approx tH\left(\frac{n_x}{t}\right) = -n_x \ln \frac{n_x}{t}$. This comes from the fact that for each block of length $\left\lceil \frac{t}{n_x} \right\rceil$ you have approximately one zero which requires about $\lceil \log_2 \left(\left\lceil \frac{t}{n_x} \right\rceil \right) \rceil$ bits to be describe. A more detailed proof is provided by Theorem 2.8.1, p180 in [4].
Now fixing $K_\phi(x) = k$ this means that $n_x (\ln t - \ln n_x) = k$ is constant.

Recalling that Kolmogorov complexity is not computable the practical positioning on this graph may sound impracticable. The aim of proposition 1 is precisely to tackle with this issue : the boundary between $k < K_\phi(T)$ and $k > K_\phi(T)$ may be substituted with $k < \lceil \log_p(T) \rceil - \left\lceil f(\lceil \log_p(T) \rceil) \right\rceil$ and $k > \lceil \log_p(T) \rceil$ for some adequate f (like $f : x \mapsto \log_p(x)$). The gray area inbetween can be made arbitrarily sparse.

5 Discussion and future work

We hope that these preliminary results pave the way for a more complete study. There are at least two additionnal directions that we believe may be of interest.

Firstly we would like to address the issue of learning and coordination. Consider any two player repeated game and assume that player one is limited to a fixed subset of strategies. We would like to know under which conditions player two can achieve perfect coordination with him.

Assume that both players are limited to computable strategies - but without any bound on their complexity. We conjecture that the halting problem prevents player two from having a strategy which would eventually coordinate itself with any strategy of player one. On the other hand is player two is enhanced with a halting Oracle[16] then by exhaustively enumerating all Turing strategies and progressively discarding those which are not compatible with past history player two eventually coordinates itself with any strategy of player one. Now assume that player one has a bound on the complexity of his strategies. This implies that the set of his strategies is finite - which ensures following [6] that player two has a strategy which eventually coordinates itself with player one. However the halting problem again prevents this strategy from being implemented by a Turing machine.

The second direction we are interested in concerns mixed strategies. Consider thet set of rational distributions of probabilities over actions and let $\Sigma = (\mu_n)_{n\in\mathbb{N}}$ be the subset of those that can be implemented by a Turing machine. There exists a distribution (namely $\sum_{n\in\mathbb{N}} p^{-K_\mu(n)}\mu_n$) which is absolutely continuous with respect to any μ_n. Hence the "grain of salt" hypothesis is true and the result by [3] ensures that the path of play may converge to a repetition of single-stage Nash equilibrium (in mixed strategies). However there is no *computable* distribution which is a.c. with respect to any computable distribution. On the other hand if player one is bounded in complexity by some $k \in \mathbb{N}^*$ then $\overline{\mu}_k = \sum_{\substack{n\in\mathbb{N} \\ K_\mu(n)\leq k}} p^{-K_\mu(n)}\mu_n$ satisfies grain of salt again. Moreover it is of complexity no more than p^k (although its actual complexity could be smaller). The problem is that $k \mapsto \overline{\mu}_k$ is not computable : there exists a strategy for player two but noone could ever effectively find it.

The last orientation that could appear fruitful concerns the link between Kolmogorov complexity and entropy : it follows from theorem 8.1.2 p524 of [4] that $H(X_1, \ldots, X_n) \approx_{n\to+\infty}$

[16] A Turing machine with a halting Oracle is a standard Turing machine augmented by a device which at each steps computes in finite time wether a given program would eventually halt on a given input. Such an Oracle is not Turing-computable.

$\mathbb{E}_{\mathbb{P}}\left(K_\phi(X_1,\ldots,X_n)\right)$ for any finite random sequence X drawn from a computable rational probability distribution. This property possibly opens a new field of application for many results on the entropy of a mixed strategy in a repeated game. We look forward to materialize them.

Références

[1] R.J. Aumann. Survey of repeated games. In R.J. Aumann, editor, *Essays in game theory and mathematical economics in honor of Oskar Morgenstern*, pages 11–42. Wissenschaftsverlag, Bibliographisches Institut, Mannheim, Wien, Zurich, 1981.

[2] E. Kalai. Bounded rationality and strategic complexity in repeated games. *Game Theory and Applications*, pages 131–157, 1993.

[3] E. Kalai and E. Lehrer. Rational learning leads to Nash equilibrium. *Econometrica*, 51 :1019–1045, 1993.

[4] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, 1997. Second Edition.

[5] A. Neyman. Finitely repeated games with finite automata. *Mathematics of Operations Research*, 23 :513–551, 1998.

[6] A. Neyman and D. Okada. Two-person repeated games with finite automata. *International Journal of Game Theory*, 29 :309–325, 2000.

[7] A. Rubinstein. *Modelling Bounded Rationality*. MIT Press, 1998.

6 Appendices

6.1 Proof of proposition 1

Proof (i) Let $k \in \mathbb{N}$ the number of an implementation of identity : $\phi_k : x \mapsto x$.
Then $K_\phi(x) \leq |x| + \underbrace{K_\phi(k) + d}_{m} = \lceil \log_p(x) \rceil + m$.

(ii) Let $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \xrightarrow[x\to+\infty]{} +\infty$. Let $\epsilon > 0$.
For $n \in \mathbb{N}^*$ and $c \in \mathbb{N}$ let

$$A_n(c) = \left\{x \in \left[p^{n-1}, p^n - 1\right] \ , \ K_\phi(x) \leq n - c - 1\right\}$$

Then $n \in \mathbb{N}^*$ and $c \in \mathbb{N}$ we have

$$\begin{aligned}
|A_n(c)| &= \left|\left\{x \in \left[p^{n-1}, p^n - 1\right] \ , \ \exists k \in \mathbb{N}, |\langle k \rangle_{\mathcal{A}}| \leq n - c - 1, \phi_k(\emptyset) = x\right\}\right| \\
&\leq \left|\left\{k \in \mathbb{N}, |\langle k \rangle_{\mathcal{A}}| \leq n - c - 1\right\}\right| \\
&= \left|\bigcup_{l=0}^{n-c-1} \left\{k \in \mathbb{N}, |\langle k \rangle_{\mathcal{A}}| = l\right\}\right| \\
&= \sum_{l=0}^{n-c-1} p^l \\
&= p^{n-c} - 1
\end{aligned}$$

For $I \subset \mathbb{N}$ let

$$B_I = \{x \in I\,,\ \lceil \log_p(x) \rceil - f\left(\lceil \log_p(x) \rceil\right) - 1\}$$

Let $n \in \mathbb{N}^*$; then $\forall x \in [p^{n-1}, p^n - 1]\ ,\ \lceil \log_p(x) \rceil = n$.
Thus $B_{[p^{n-1},p^n-1]} = A_{[p^{n-1},p^n-1]}\left(f(n)\right)$ hence

$$\left|B_{[p^{n-1},p^n-1]}\right| \leq p^{n-f(n)}$$

It follows that for $N \in \mathbb{N}^*$

$$\begin{aligned}
\left|B_{[1,N]}\right| &\leq \left|B_{\left[1,p^{\lceil \log_p(N) \rceil}\right]}\right| \\
&= \left|B_{[1,p-1]} \bigcup B_{[p,p^2-1]} \bigcup \cdots \bigcup B_{\left[p^{\lceil \log_p(N) \rceil - 1}, p^{\lceil \log_p(N) \rceil} - 1\right]} \bigcup B_{\left\{p^{\lceil \log_p(N) \rceil}\right\}}\right| \\
&= (p-1) + \left(\sum_{n=1}^{\lceil \log_p(N) \rceil - 1} \left|B_{[p^{n-1},p^n-1]}\right| \right) + 1 \\
&\leq p + \sum_{n=1}^{\lceil \log_p(N) \rceil - 1} p^{n-f(n)}
\end{aligned}$$

Since $f(n) \xrightarrow[n \to +\infty]{} +\infty$ let $n_0 \in \mathbb{N}^*$ be such that $\forall n \geq n_0$, $f(n) > \log_p\left(\frac{2p}{\epsilon}\right)$. Then for $N \geq p^{n_0}$ we have

$$\begin{aligned}
\left|B_{[1,N]}\right| &\leq \underbrace{p + \sum_{n=1}^{n_0-1} p^{n-f(n)}}_{M_0} + \sum_{n=n_0}^{\lceil \log_p(N) \rceil - 1} \frac{p^n}{p^{f(n)}} \\
&\leq M_0 + \frac{\epsilon}{2p} \sum_{n=n_0}^{\lceil \log_p(N) \rceil - 1} p^n \\
&\leq M_0 + \frac{\epsilon}{2p} \left(p^{\lceil \log_p(N) \rceil} - p^{n_0}\right) \\
&\leq M_0 + \frac{\epsilon}{2p} \left(p^{\log_p(N)+1} - 0\right) \\
&= M_0 + \frac{\epsilon}{2} N
\end{aligned}$$

Let $N_0 \geq p^{n_0}$ such that $\forall N \geq N_0$, $\frac{M_0}{N} < \frac{\epsilon}{2}$; then for $N \geq N_0$ we have

$$\frac{1}{N}\left|B_{[1,N]}\right| < \epsilon$$

thus $\frac{1}{N}\left|[1,N] - B_{[1,N]}\right| > 1 - \epsilon$ that is

$$\frac{1}{N}\left|\left\{x \in [1,N]\,,\ K_\phi(x) \geq \lceil \log_p(x) \rceil - f\left(\lceil \log_p(x) \rceil\right) - 1\right\}\right| > 1 - \epsilon$$

This holds for any $N \geq N_0$ which completes the proof with

$$E_{f,\epsilon} = \bigcup_{N \geq N_0} \left([1, N] - B_{[1,N]} \right)$$

□

6.2 Proof of lemma 1

Let $t \geq 2$. $\phi_{\langle 2\overline{t} \rangle_{\mathcal{A}}}$ is an implementation of $\mathcal{D}^i_t$ by definition. Thus $K_\phi(\mathcal{D}^i_t) \leq \left|2\,\overline{t}\right| = 1 + K_\psi(t)$ since $\overline{t}$ is a shortest description of $t \geq 2$ in the enumeration ψ.

On the other hand $CountDefection_k$ computes t for any k such that ϕ_k implements $\mathcal{D}^i_t$ (and in particular for $k = \langle 2\,\overline{t} \rangle_{\mathcal{A}}$). Since $\langle 5\overline{k} \rangle_{\mathcal{A}}$ implements $CountDefection_k$ it follows that $K_\phi(t) \leq K_\phi(\mathcal{D}^i_t)$.

To conclude note that $\forall n, \psi_n = \phi_{\langle 7\overline{n} \rangle_{\mathcal{A}}}$ so that $K_\psi(t) \leq K_\phi(t) + 1$. And by construction of the enumeration ψ there is no shorter description of t than $7\overline{n}$ in the enumeration ψ. Thus $K_\phi(t) = K_\psi(t) + 1$ so that $K_\phi(\mathcal{D}^i_t) \leq K_\phi(t) \leq K_\phi(\mathcal{D}^i_t)$ which completes the proof.

6.3 Proof of lemmas 2 and 3

Let $x \in \mathcal{H}_*$. $\phi_{\langle 3\overline{x} \rangle_{\mathcal{A}}}$ is an implementation of $\mathcal{E}^i_x$ by definition. Thus $K_\phi(\mathcal{E}^i_x) \leq |3\,\overline{x}| = 1 + K_\psi(x) = K_\phi(x)$.

On the other hand $GetPathAndDefection_{k,T}$ returns x for any k such that ϕ_k implements $\mathcal{E}^i_x$. Since $\langle 6\overline{k} \rangle_{\mathcal{A}}$ implements $GetPathAndDefection_{k,T}$ it follows that $K_\phi(x) \leq K_\phi(\mathcal{E}^i_x)$ which proves lemma 2.

Now let $t \in [2, T - |x|]$. $\phi_{\langle 4\overline{x,t} \rangle_{\mathcal{A}}}$ is an implementation of $\mathcal{ED}^i_{x,t}$ by definition ; thus $K_\phi(\mathcal{ED}^i_{x,t}) \leq \left|3\,\overline{x,t}\right| = K_\phi(x,t)$.

Since $GetPathAndDefection_{k,T}$ returns (x,t) for any k such that ϕ_k implements $\mathcal{ED}^i_{x,t}$ and since $\langle 6\overline{k} \rangle_{\mathcal{A}}$ implements $GetPathAndDefection_{k,T}$ it follows that $K_\phi(x,t) \leq K_\phi(\mathcal{ED}^i_{x,t})$ which completes the proof of lemma 3.