POLARIS Team, LIG, Grenoble

M2 Internship Report

Random Potential Games with Partial Interactions

Author
Léo Gayral

Supervisors
Federica Garin
Bruno Gaujal

28/01/2019 – 28/06/2019

Undergoing of the Internship

The main topic of my internship was *Potential Games with Partial Interactions*. I spent most of my time working on this topic, particularly during the first two months. The groundwork from which I stemmed is essentially Stéphane Durand's PhD *Analysis of Best Response Dynamics in Potential Games* [Dur18]. The majority of the report will focus on my work on the *Black Box Dynamics*, which is deeply related to this thesis.

In order to explore the aspects of this model less suited to a pure theoretical study, I set up several numerical simulations, which required more than a hundred hours of running time from end to end, too much for my small laptop. This is why I learned about the basic usage of the OAR batch scheduler on the CIMENT clusters.

I also followed a MOOC about reproducible research, on the advice of an Inria researcher from my research team who worked on its creation. I learned a lot of useful methodological tips, and it gave me a reason to learn how to use Markdown and Jupyter Notebooks. This knowledge of Python notebooks already proved itself useful when I had to synthesise the results of the aforementioned simulations into clearly understandable documents.

I also worked on several smaller projects. I read Pólya's classic book *How to Solve It* [Pol04], and attended a weekly reading group on Goodfellow's *Deep learning* [GBC16]. I finally made a proper academic webpage in order to bind together in a nice package all the documents worth sharing I have produced up to now.

I would like to thank Bruno Gaujal for this internship opportunity, as well as Federica Garin. While the main topic of my internship has been interesting in itself, my thanks also go to the rest of the POLARIS/DATAMOVE teams for all the learning opportunities they brought to me along the way.

Contents

Introduction

During this internship, I worked in the context of potential games. These games have been defined in [Ros73], and have been extensively studied since [MS96]. Potential games are known to be equivalent to congestion games, a general paradigm for resource-sharing problems. Thanks to this equivalence, Potential games have been notably applied to routing games [MM56], load balancing in HPC and distributed optimisation [Rou05].

Such games have well-known canonical algorithms, with worst-case behaviours long since studied. However, for practical applications, the average complexity matters as much if not more than the worst cases. Consequently, my work is part of a recent trend in algorithmic game theory, which consists in studying the average behaviour of these so-called canonical algorithms.

In the following sections, we will begin with some prerequisite knowledge, upon which my work was based. Then, we will introduce an algorithm called the Black Box Dynamics (BBD), and a distributed framework, in which this approach seems more natural than the Best Response Dynamics (BRD). Finally, we will show that the minimal convergence time in this setup is a $\tilde{\Theta}\left(A^2N^2\right)$ for N players and A actions. After this, we will have a much briefer look at two variants of BRD, by changing the clocks used by the players or the topology of the action space, and then end the report on a few perspectives for future works.

1 Potential Games and the Best Response Dynamics (BRD)

In this section, I will provide a brief overview of *Potential Games* and the *Best Response Dynamics*, in order to provide the required background to understand my work on the subject in the rest of this report.

The baseline object at the center of my study was this class of games:

Definition 1 (Matrix Games)**.** *We define a game $\mathcal{G}$ as a tuple $\left(\mathcal{N}, \mathcal{X} = \prod_{k\in\mathcal{N}} \mathcal{A}_k, (u_k)_{k\in\mathcal{N}}\right)$. More precisely, in this tuple:*

- $\mathcal{N} = [N]$ *is the (finite) set of* players*,*
- $\mathcal{A}_k$ *is the (finite) set of* actions *available to the player $k \in \mathcal{N}$,*
- $u_k : \mathcal{X} \to \mathbb{R}$ *is the* payoff function*, the reward which player k wants to maximise.*

We may simplify the model by assuming, without loss of generality, that all the players share a common action set $\mathcal{A} = [A]$, hence $\mathcal{X} = \mathcal{A}^{\mathcal{N}}$.

Because each player wants to maximise their own reward, a canonical transition law consists in performing a *best response*:

Definition 2 (Best Response Correspondence)**.** *For a game $\mathcal{G}$ we define the best response of player $k \in \mathcal{N}$ in the state $x \in \mathcal{X}$ as the set of actions $BR_k(x) := \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, u_k\left(x\left[x_k := a\right]\right)$.*

These transitions, used successively, result in a dynamic on the game. We are interested in the fixed-points for this dynamic, historically called *pure Nash equilibria*:

Definition 3 (Pure Nash Equilibrium)**.** *A state $x \in \mathcal{A}$ is a pure Nash equilibrium if, for any player $k \in \mathcal{N}$, we have $x_k \in BR_k(x)$.*

We may call such a state a Nash equilibrium or even an equilibrium from now on, for simplicity's sake.

A Nash equilibrium can be seen as a local maximum of all the functions u_k on the graph $\mathcal{X}$ with edges between states with Hamming distance equal to 1. Note that such a pure equilibrium does not always exist. For example, in the two-players zero-sum game with payoff function $u_1 = \mathbb{1}_{x_1=x_2} - \mathbb{1}_{x_1\neq x_2}$, player 1 wants to agree and player 2 to disagree. u_1 has two strict local maxima, both being strict minima of u_2, hence a game without equilibrium. This game is commonly known as Matching Pennies, and has been studied back and forth. The same argument can be made for rock–paper–scissors, as any action for a player is weak against an action from their adversary. By adding a supplementary structure on matrix games, we can however guarantee the existence of such an equilibrium.

Definition 4 (Potential Games)**.** *We say $\mathcal{G}$ is a potential game if:*

$$\exists \Phi : \mathcal{X} \to \mathbb{R},\ \forall k \in \mathcal{N},\ \forall x, y \in \mathcal{X} \text{ s.t. } x_{-k} = y_{-k},\ u_k(x) - u_k(y) = \Phi(x) - \Phi(y)\,.$$

Note that, in this case, the potential function Φ is unique, up to an additive constant.

Proposition 1 (Existence of a Nash Equilibrium)**.** *Consider a potential game $(\mathcal{G}, \Phi)$. Then the Nash equilibria of $\mathcal{G}$ are exactly the local maxima of the potential Φ.*

By finiteness, as Φ has a global maximum, then $\mathcal{G}$ admits at least one Nash equilibrium.

A Nash equilibrium is not necessarily a desirable course of actions. Typically, consider the prisoner's dilemma with actions a for *ally* and b for *betray*. The rewards are $u_1(a,a) = 1$, $u_1(a,b) = -1$, $u_1(b,a) = 2$, $u_1(b,b) = 0$ and $u_2(x_1,x_2) = u_1(x_2,x_1)$. Then $\Phi(x_1,x_2) = \mathbb{1}_{x_1=b} + \mathbb{1}_{x_2=b}$ is a potential function for this game. Hence, the only Nash equilibrium of $\mathcal{G}$ is the "bad" choice (b,b), whereas the "good" choice (a,a) is not.

Though such equilibriums are not desirable, they are still interesting to study for two reasons. First, a selfish behaviour may be the best outcome an agent with limited information may strive for, so as already stated it is natural to study it. Second, in many cases, we have good inequalities between the worst Nash equilibrium and the optimal "centralised" solution, called *Price of Anarchy* bounds. For example, in discrete congestion games with a finite number of players [CK05], the Price of Anarchy is equal to $\frac{5}{2}$.

In this framework, playing consecutive best responses for a sequence of players $(K_t)_{t>0} \in \mathcal{N}^{\mathbb{N}^*}$, called the *revision* sequence, yields a dynamic on the game described in Algorithm 1.

Algorithm 1: BRD Algorithm

```
Input: Initial state x_0 ∈ X, revision sequence (K_t)_{t>0}.
for t ∈ N:
    k = K_{t+1}
    if u_k(x_t) ∉ BR_k(x_t):
        b ∈ BR_k(x_t)
        x_{t+1} = x_t [x_{t,k} := b]
    else:
        x_{t+1} = x_t
```

Proposition 2. *Using Algorithm 1, a revision sequence on $\mathcal{N}$ reaches a Nash equilibrium for any game* iff *for any player k the set $\{t \in \mathbb{N}^*, K_t = k\}$ is infinite.*

This convergence property is indeed agreeable, but it gives no guarantee on the complexity of the algorithm, the number of steps until an equilibrium is reached. Actually, as seen in [Dur18], one can conceive an example which needs $\Omega\left(NA^{N-1}\right)$ steps to reach the equilibrium for any revision sequence, hence an exponential worst-case scenario. The novelty of [Dur18] is to study randomly generated potential games, in order to compute the average complexity of Algorithm 1 among other things.

Theorem 1 (Average Complexity of BRD)**.** *Consider a uniform random potential game with A actions and N players, such that $\Phi(x) \overset{iid}{\underset{x\in\mathcal{X}}{\sim}} \mathcal{U}([0,1])$.*

Using a round-robin revision sequence, we can bound the average number of steps in $[N, e^{\gamma}N + o(N)]$. Likewise, using a uniformly random revision sequence, the average number of steps is a $\Theta(N\ln(N))$.

2 Bandit Policies and the Black Box Dynamics (BBD)

One of the defaults of the BRD algorithm, in practical situations, is that it uses BR_k as an elementary operation. However, computing the best response in a given state $x \in \mathcal{X}$ requires $A-1$ comparisons. In a distributed framework, where atomic operations require a fixed delay δ, it is more coherent to consider each comparison as an elementary operation.

For this reason, I formalised and studied a model where the elementary operation of the dynamics is to compare *one* alternative action to the current state, and to keep the best out of these two. The way a player samples only one action at a time corresponds to a typical bandit framework. We named this process the *Black Box Dynamics* (BBD), in the sense that the potential function Φ can be evaluated on a given state $x \in \mathcal{X}$, but cannot be studied *per se*. The way I see it, this means the sampled action has to actually be performed by a player in order to observe its corresponding reward and reach a conclusion. An illustration of the phenomenon would be a car routing problem, where each driver has to try a new path to estimate its length and conclude whether it is faster than their previous one.

The black box dynamics on a game $\mathcal{G}$ is then given by a revision sequence of players and actions $(K_t, B_t)_{t>0}$. Just like for BRD, the fixed-points of BBD are exactly the Nash equilibria of $\mathcal{G}$, and we have the following result:

Proposition 3. *To properly define a revision sequence, we need here to fix N and A, thus the state space $\mathcal{X} = \mathcal{A}^{\mathcal{N}}$, beforehand.*

If a revision sequence $(K, B) \in (\mathcal{A}\times\mathcal{N})^{\mathbb{N}^*}$ visits each player-action pair an infinite amount of times, then the induced dynamics reaches a Nash equilibrium for any potential $\Phi : \mathcal{X} \to \mathbb{R}$.

Unlike for the sequence K in BRD, this condition on (K, B) is sufficient but not necessary, and there exists a "universal" revision sequence with finite length which converges on all the considered games.

Proof. The first implication is as easy as for BRD. Consider a state x_t of a game $\mathcal{G}$. If x_t is already a Nash equilibrium, then we are done. If not, then there is a player p and an action b for which $\Phi(x_t[x_{t,k} := b]) > \Phi(x_t)$. Because of our hypothesis on (K, B), starting at any time $t \in \mathbb{N}^*$, we will encounter a time $t' > t$ such that $(K_t, B_t) = (k, b)$. Hence, $\tau = \min\{s \in [t+1, t'], \Phi(x_s) > \Phi(x_t)\}$ is well-defined, and we jump from x_t to x_τ. By finiteness of $\mathcal{X}$, this increasing sequence (with respect to Φ) is finite, hence a Nash equilibrium is reached in a finite amount of time.

For BRD, only the number of players N was specified, hence an infinite amount of games, including the aforementioned worst cases with minimal complexity $NA^{N-1} \underset{A\to\infty}{\longrightarrow} \infty$. Hence, no sequence with finite length could converge for all games. For BBD, though, we also specify beforehand the number of actions A. If we forget about the actual values taken by the potential function Φ to only look at the resulting total order on $\mathcal{X}$, which encapsulates all the results of the comparisons between two actions, we obtain a finite set of games $\mathcal{Z}$. By using the aforementioned infinite sequence on each game and initial state $(\mathcal{G}, x_0) \in \mathcal{Z}\times\mathcal{X}$, we reach an equilibrium after $\tau(\mathcal{G}, x_0) \in \mathbb{N}$ steps. Thus, after $\tau := \max\{\tau(\mathcal{G}, x_0), (\mathcal{G}, x_0) \in \mathcal{Z}\times\mathcal{X}\} \in \mathbb{N}$ steps, all the games reach a Nash equilibrium. □

Now, the game is played over an infinite time horizon, and we are interested in when an equilibrium is reached. Using the very same example as for BRD [Dur18], we can show that the worst case for BBD is exponential, and requires at least NA^{N-1} steps for any revision sequence to reach a Nash equilibrium. In order to begin the transition to the distributed framework, and to allow for explicit definitions of the random variables we will study, let me introduce a centralised algorithm that behaves just like BBD, but with some added structure.

Algorithm 2: BBD Algorithm

```
Input: Game 𝒢, State x_0, Revision sequence (K_t, B_t)_{t>0}
L = ∅    # List of satisfied players
for k in 𝒩:
    M_k := {x_{0,k}}        # Actions explored by player k
    φ_k := Φ(x_0)           # Potential observed by player k
for t ∈ ℕ:
    k, b = K_{t+1}, B_{t+1}
    if Φ(x_t) ≠ φ_k:    # The state x changed since the last time k played
        φ_k = Φ(x_t)
        M_k = {x_{t,k}}
    if b ∉ M_k:         # The action b was unexplored by k on the current state
        if Φ(x_t[x_{t,k} := b]) > φ_k:
            x_{t+1} = x_t[x_{t,k} := b]
            φ_k = Φ(x_{t+1})
            L = ∅
        else:
            x_{t+1} = x_t
        M_k = M_k ∪ {b}
    if M_k = 𝒜:         # The current player was satisfied
        L = L ∪ {k}
```

The interest of this structure stems from the fact that it is hard to formalise and study the precise moment an equilibrium is reached, as it depends on neighbouring states $x \in \mathcal{X}$ which we have never seen before. Hence, we define here T_{BBD} as the first time step at which $L = N$, meaning that all the players acknowledge they have reached an equilibrium from their perspective. More precisely, L is the list of satisfied players at a given time, for whom the current action is their best response. Each player k has a list M_k of the alternative actions they explored in the current state. The list L is updated every time a player chooses an alternative action, or when M_k is filled. Assuming that Φ is injective, that the order induced on $\mathcal{X}$ is strict, each player k also keeps track of a potential φ_k, which serves as a way for them to know whether the state changed since the last time they played.

During a run of the algorithm, some potential comparisons are skipped because $b \in M_k$. We call the steps for which $b \notin M_k$ *useful*, and denote S_{BBD} the amount of useful steps until T_{BBD}. The operation we are interested in for the computational complexity of the algorithm is the amount of comparisons between two potentials, and the total amount of comparisons performed by the algorithm is precisely S_{BBD}. In other words, S_{BBD} represents the algorithmic complexity required in order to reach an equilibrium, while T_{BBD} represents the time complexity of the process, which includes not only the useful steps but also the idle steps.

Until now, we have only described deterministic games and revision sequences. However, to allow a Markovian approach, we must add some randomness to the setting. From now on, we will consider a uniformly-generated random game:

$$(\Phi(x))_{x\in\mathcal{X}} \overset{\text{iid}}{\sim} \mathcal{U}([0,1])\,. \tag{1}$$

Please note that, by symmetry, as soon as all the potentials $(\Phi(x))$ are *iid* and follow a density law, the order induced on $\mathcal{X}$ itself will be uniform and almost-surely strict, which is the property we are actually interested in. We also assume that the random action sequence $(B_t)_{t\in\mathbb{N}^*} \overset{\text{iid}}{\sim} \mathcal{U}(\mathcal{A})$ and the player sequence $(K_t)_{t\in\mathbb{N}^*} \overset{\text{iid}}{\sim} \mathcal{U}(\mathcal{N})$ are independent of the potentials and of each other.

If we generate each random potential when we read it for the first time *during* a run of the algorithm, then we are almost in a Markovian framework: most of the time, we will follow a random law by reading an unseen potential, but from time to time we may read again an already-seen smaller potential. We call such an event an *intersection*, and we denote I_{BBD} the amount of intersections until T_{BBD}, the number of times the algorithm has a non-Markovian behaviour.

You can find a Python implementation of Algorithm 2 in Appendix A, in the randomised setting described above, with an added *collision* mechanism which we will discuss later on, in Section 4. Assuming `Col_prob=0` so that no collisions can happen, this code simulates exactly a run of the algorithm until an equilibrium is reached, and returns T_{BBD}. In the code sample, I sticked as much as possible to the notations of the algorithm above. One reason for using Python is that it is the programming language I am the most used to, but there also is a good practical reason for it. In the case of BBD or BRD, though we generate the potentials of the game on the fly, we want them to be fixed and stored afterwards, until the end, or we may miss an intersection. The state space $\mathcal{X}$ has a full size A^N, hence even with only $A = N = 10$ actions and players we would need several GB of memory to store an initially empty array, which would still be mostly empty when the algorithm stops. Python has the advantage of having a native implementation of the dictionary data structure, which allows us to store only the data we need in a much more efficient fashion. You can see this idea implemented in the Appendix, through the use of the dictionary `S`, whose structure is maintained by the auxiliary functions `Phi` and `Code`.

3 Intersection-Free Approximation (IFA) of BBD

Following the idea introduced in [Dur18], the so-called *intersection-free approximation* (IFA) is a Markovian simplification of the model above, where each potential is taken at random, even if a player has already seen the corresponding state previously. In this simplification, we also consider that whenever a player k switches for a better alternative (which corresponds to the lines 12 to 15 of Algorithm 2), the list M_k is reset to $\emptyset$. This twisted process induces a new stopping time T_{IFA} and a new amount of useful steps S_{IFA}.

We can establish a coupling between BBD and IFA by using a common random potential sequence $(\Phi_j)_{j\in\mathbb{N}}$. At each useful time step, IFA will read a new potential, while BBD may wait until the next one because of an intersection. Let us then denote V_{BBD} (resp. V_{IFA}) the amount of potentials read from this sequence until the time step T_{BBD} (resp. T_{IFA}) of Algorithm 2. By definition of IFA, one new potential is read at each useful step, so $V_{\mathrm{IFA}} = S_{\mathrm{IFA}}$.

Whether using BBD or IFA, because we check if the alternative action has already been explored in the current state (line 12 of Algorithm 2) *before* exploring it if needed, we will read at most $m := (A-1)N$ potentials until a player switches their action for a better one, or else a Nash equilibrium is reached. Hence, the following properties always hold:

Proposition 4. *For any given potential and revision sequences, we have* $S_{BBD} = V_{BBD} + I_{BBD}$ *and* $V_{BBD} \leq V_{IFA}$.

Proof. The first point holds by definition of the variables, as BBD reads a potential at each useful timestep, which may or may not be an intersection.

For the inequality, notice that IFA stops when we reach a potential Φ_j which is maximal so far, the equivalent of a Nash equilibrium in this framework, and then read exactly m smaller potentials. Because of this, assuming BBD kept reading potentials until Φ_j to begin with, it will read *at most* m alternatives

(perhaps less because of intersections, which will also result in potentials smaller than Φ_j) and then stop, hence $V_{\mathrm{BBD}} \leq V_{\mathrm{IFA}}$. □

In particular, we deduce that $\mathbb{E}[S_{\mathrm{BBD}}] \leq \mathbb{E}[S_{\mathrm{IFA}}] + \mathbb{E}[I_{\mathrm{BBD}}]$. Because of its entirely Markovian behaviour, the process IFA can be studied quite precisely. Without any effort, because L must be filled, hence because each S_k must be filled, we need at the very least m useful steps until the algorithm stops whether in BBD or IFA.

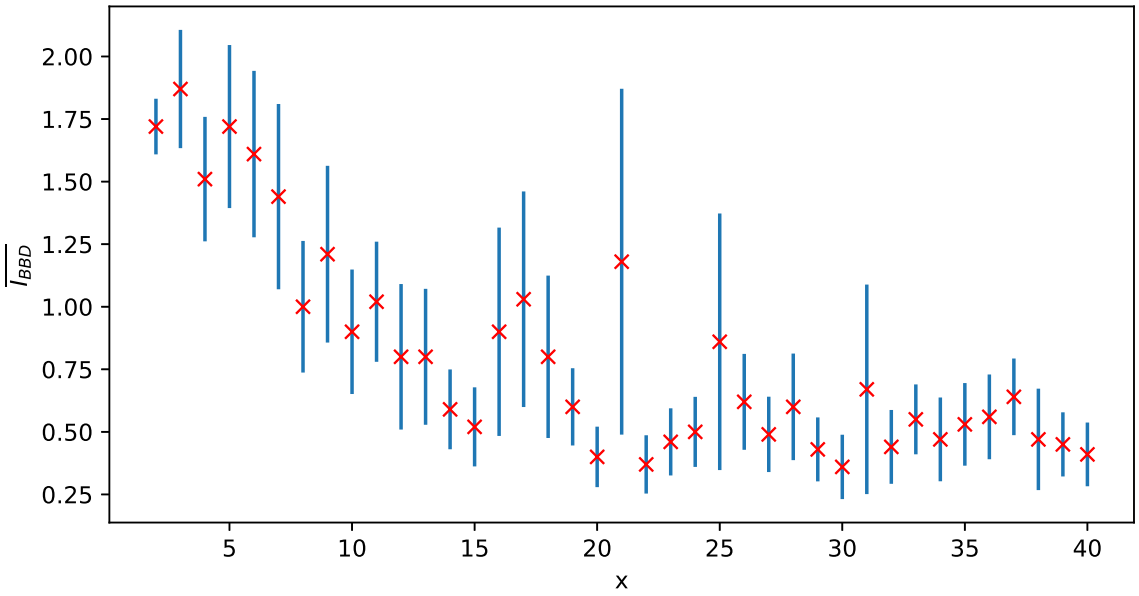

Figure 1: 95% confidence intervals for $\mathbb{E}[I_{\mathrm{BBD}}]$, using samples of 100 simulations for each interval, by making both parameters $A = x$ and $N = x$ vary.

I_{BBD}, however, is quite hard to grasp precisely. Its worst-case value may be catastrophic, but we are only looking at its average value here, which is really small as seen in Figure 1. More interestingly, the bigger the game, the lower the mean value gets, probably because, as the dimension of $\mathcal{X}$ gets higher, the trajectories leading to intersections get less and less probable. This means that the typical behaviours of BBD and IFA get closer for bigger amounts of players and actions.

Theorem 2. *In this framework, we have* $\mathbb{E}[S_{BBD}] = \tilde{\Theta}(m)$.

Proof. First of all, remember the stochastic lower bound $S_{\mathrm{BBD}} \geq m$, as each of the N players must read $A - 1$ actions at the very least to be satisfied.

For the upper bound, let us first study I_{BBD}. To do so, let us remark that each intersection occurs because, at two points in time during the execution of BBD, two players compared the same state to their respective positions, meaning that these were at a Hamming distance 1 or 2. Each such pair of configurations can be associated to *at most* $A - 2$ intersections during the whole game (if they are at distance 1) or 2 (at distance 2). Consequently, instead of counting the amount of *intersections* in BBD, we just have to estimate M_{BBD} the amount of *moves* from one state to another actually performed until T_{BBD}, and then use the stochastic upper bound $I_{\mathrm{BBD}} \leq \max(A-2, 2) \times \frac{M_{\mathrm{BBD}}(M_{\mathrm{BBD}}-1)}{2} \leq A \times M_{\mathrm{BBD}}^2$ for any $A \geq 2$.

Now, notice that when a move occurs in BBD, it means that the new potential seen in the sequence

is bigger than all the previous ones. What's more, for the same reason as for $V_{\mathrm{BBD}} \leq V_{\mathrm{IFA}}$, when m consecutive potentials are smaller than their predecessor, then necessarily the game must have reached an equilibrium at most now. Hence, we define M the new variable that counts the number of maxima encountered in the potential sequence before a window of size m without new maximum, such that $M_{\mathrm{BBD}} \leq M$. Now, using the computations from [Dur18] (p.45) for 2 actions and m players, we obtain $\mathbb{E}\left[M^2\right] = O\left(\ln(m)^2\right)$. Hence, we have $\mathbb{E}\left[I_{\mathrm{BBD}}\right] \leq O\left(A \ln(m)^2\right)$.

For the other term, let us remember that $S_{\mathrm{IFA}} \leq T_{\mathrm{IFA}}$ stochastically. Now, let us use the upper bound $\mathbb{E}\left[T_{\mathrm{IFA}}\right] = O\left(m \ln(m)\right)$ from Theorem 4 shown later on. From this, it naturally follows that $\mathbb{E}\left[S_{\mathrm{BBD}}\right] = O\left(m \ln(m)^2\right)$, hence the announced result. □

4 Distributed Framework

In the previously described models, the explicit use of a revision sequence required a centralised or at least synchronised behaviour. To get closer to a distributed framework, we want each player to follow their own internal clock in order to decide when they play, to choose an action uniformly at random, and then to follow the behaviour described by the lines 8-20 of Algorithm 2. We will here describe this new framework for BBD and IFA, make a full study of T_{IFA}, and then conclude by comparing it with BBD.

In this framework, we add a new parameter, δ, to denote the length of the turn of a player, *i.e.* the time they need to explore an alternative action. Because of this, if a new player starts playing before the previous one ended their turn, a *collision* occurs. In our framework, where we have to actively perform the alternative action during the delay δ, it is quite hard to define what happens to the measure of the potential, both for the previous and the new player. We will here consider that whenever a collision occurs between two players, this causes the first player to notice something is off during their turn, hence they will keep playing the new action until their next play. In the case of IFA, because of the Markovian behaviour, it is equivalent to describe collisions as *full-restart* scenarios, where the game starts from scratch, as the lists are reset and the new potential is *always* uniform in $[0, 1]$.

This will actually have a huge impact on the amount of time the algorithm needs to come to a halt. Indeed, in [Dur18], as δ corresponds to the time the player spends computing the best course of action, without doing anything, collisions can only occur when this player is unsatisfied and decides to move, which yields a pseudo-linear complexity $\tilde{O}(m)$. The alternative way of dealing with collisions, as we will see later on, yields a pseudo-quadratic complexity $\tilde{O}\left(m^2\right)$, which is indeed worse but still polynomial with a low degree, much lower than the exponential worst case.

The last thing left to define is the clocks used by the players. In a purely centralised and sequential framework, the global order of the operations is fixed, and there is no need to consider their precise timing. In a distributed framework, though, the order of the operations performed by each player may be fixed but their actual outcome relies on how they intertwine with the other players' operations. By using clocks, which assign a timestamp to each operation, we are now able to deduce a global order for the operations, hence to study the induced dynamics.

While not the most efficient in practical cases, we will here use independent Poisson clocks with a common parameter $\frac{\lambda}{N}$. For a Poisson clock with parameter θ, the time interval between two consecutive ticks follows an exponential distribution $\mathcal{E}(\theta)$, and the intervals are independent of each other. Such clocks are a canonical tool in a lot of practical cases, like for studying arrival times in distributed communication protocols. These Poisson clocks also allow for an elegant theoretical study, as it is now equivalent to study a game with a centralised clock of parameter λ and to pick a player uniformly at random at each tick. Hence, the collision probability between two consecutive turns is $p = 1 - e^{-\lambda\delta}$, and we will denote $q = 1 - p$.

5 Lower Bound with the Clumsy Coupon Collector (CCC)

Using a randomised revision sequence, a simple lower bound on the amount of steps of BRD can be obtained through the coupon collector problem. In order to find a lower bound on the convergence time for the black box dynamics, let us define a variant called the *clumsy coupon collector* (CCC). In this model, the player wants to collect m coupons to complete their collection. Each acquired coupon is picked uniformly at random among the m possible ones. Each time they obtain a *new* coupon, when they add it to their existing collection, they have a probability p of losing their whole deck in the process.

This model can be found in BBD if we only look at the amount of steps which occur between a move and a collision, or after the last move. The total amount of alternative actions explored by the players corresponds here to the number of coupons. This amount of steps will be lower than the overall amount of time-steps T_{BBD}.

Let us denote C_k the average amount of coupons the player will acquire until they complete their collection, starting with k coupons initially. We have $C_m = 0$, and the recursion scheme $C_k = 1 + \frac{k}{m} C_k + \frac{m-k}{m}(pC_0 + qC_{k+1})$.

Proposition 5. *Let us define* $v_k(q) = \sum_{j=1}^{k} \frac{q^{k-j}}{j}$, $g_k(q) = \sum_{j=0}^{k-1} q^j = \frac{1-q^k}{1-q}$ *and* $\theta = \frac{1}{q^m}$. *Using the previous recursion scheme, we obtain* $C_{m-k} = m \times v_k(q) + C_0 \times \theta \times g_k(q)$.

In particular, for m coupons, $C_0 = \frac{m \times v_m(q)}{1 - p \times g_m(q)} = m \times \sum_{j=1}^{m} \frac{1}{jq^j}$. Note that, if there are no collisions, if $q = 1$, then $C_0 = mH_m$ – where H_m is the m-th term of the harmonic series – which corresponds to the usual coupon collector problem.

Theorem 3. *Consider* $\mathcal{T}_{BBD}$ *the time elapsed during a run of the algorithm, until the time step* T_{BBD}. *Then* $\mathbb{E}[\mathcal{T}_{BBD}] \geq \delta m^2 \frac{\ln(m)-1}{\ln(\ln(m))}$ *for any choice of* λ.

Proof. Using Wald's lemma, $\mathbb{E}[\mathcal{T}_{\mathrm{BBD}}] = \frac{1}{\lambda} \times \mathbb{E}[T_{\mathrm{BBD}}] = \frac{\delta}{\ln\left(\frac{1}{q}\right)} \mathbb{E}[T_{\mathrm{BBD}}]$, with the factor on the left of the product corresponding to the average length of a time-step.

Hence, $\mathbb{E}[\mathcal{T}_{\mathrm{BBD}}] \geq \delta m^2 \times \frac{1}{\ln\left(\frac{1}{q^m}\right)} \sum_{j=1}^{m} \frac{1}{jq^j}$. Let us denote $\theta = \frac{1}{q^m} \in]1, \infty[$ so that:

$$\frac{\mathbb{E}[\mathcal{T}_{\mathrm{BBD}}]}{\delta m^2} \geq \frac{1}{\ln(\theta)} \times \sum_{j=1}^{m} \frac{1}{m} \frac{\theta^{\frac{j}{m}}}{\frac{j}{m}} = \frac{1}{\ln(\theta)} \left(H_m + \sum_{j=1}^{m} \frac{1}{m} \times \frac{\theta^{\frac{j}{m}} - 1}{\frac{j}{m}} \right). \tag{2}$$

As the rightmost sum is the right Riemann sum of a positive increasing function, we now have the lower-bound:

$$\frac{\mathbb{E}[\mathcal{T}_{\mathrm{BBD}}]}{\delta m^2} \geq \frac{1}{\ln(\theta)} \left(H_m + \int_0^1 \frac{\theta^x - 1}{x} dx \right). \tag{3}$$

Using the explicit threshold $\theta \leq \ln(m)^2$, then $\frac{\mathbb{E}[\mathcal{T}_{\mathrm{BBD}}]}{\delta m^2} \geq \frac{H_m}{\ln(\theta)} \geq \frac{1}{2} \frac{H_m}{\ln(\ln(m))}$. On the other hand, assume now $\theta \geq \ln(m)^2$. Consider the substitution $y = x \ln(\theta)$, so that $\frac{\mathbb{E}[\mathcal{T}_{\mathrm{BBD}}]}{\delta m^2} \geq \frac{1}{\ln(\theta)} \int_0^{\ln(\theta)} \frac{e^y - 1}{y} dy$. This lower bound corresponds to the mean value on the interval $[0, \ln(\theta)]$ of the slope of the line segments of exp between 0 and y, which is a positive increasing convex function on $\mathbb{R}^+$. Hence, using Jensen's

inequality, we obtain:

$$\begin{aligned} \frac{1}{\ln(\theta)} \int_0^{\ln(\theta)} \frac{e^y-1}{y} dy &= \mathbb{E}_{\mathcal{U}([0,\ln(\theta)])} \left[\frac{e^X-1}{X} \right] \\ &\geq \frac{e^{\frac{\ln(\theta)}{2}}-1}{\frac{\ln(\theta)}{2}} \\ &\geq \frac{\sqrt{\theta}-1}{\ln(\sqrt{\theta})} \end{aligned} \tag{4}$$

which is an increasing function of θ, here lower bounded by $\frac{\ln(m)-1}{\ln(\ln(m))}$, hence the wanted result. □

Note that using the same coupling, the same lower bound applies to $\mathbb{E}[\mathcal{T}_{\text{IFA}}]$.

6 Upper Bound on IFA

In the IFA framework, we can entirely describe the current state of the game at a given time by the amount of alternative actions explored (an integer $k \in [\![0, m]\!]$) and the current potential (a number in $y \in [0, 1]$). Let us denote $s_k(y)$ the average number of *actual* steps (not to be mistaken for the number of *useful* steps) starting from such a state. We also define the integral $I(y) = \int_y^1 s_0(u)du$, so that $I(0) = \mathbb{E}[T_{\text{IFA}}]$ is the overall average number of steps starting with a random potential.

In this section, we will obtain an exact expression for $I(0)$, break it down into several factors which we will study separately, and finally combine them into an upper bound on $\mathbb{E}[\mathcal{T}_{\text{IFA}}]$.

6.1 Exact Expression for $\mathbb{E}[\mathcal{T}_{\text{IFA}}]$

In this subsection, we will follow the tracks of [Dur18], and use analogous names to highlight the similarities. The exact computations here are actually really close to what happens in the case of BRD for 2 actions, as it involves precisely one comparison between the current potential, given beforehand, and a random one. This is why it may help to compare the black box setup with N players and A actions with the best response dynamics with m players and 2 actions.

Using the dynamics described above, we obtain the following equation when $k < m$:

$$s_k(y) = 1 + \frac{n+k}{AN} s_k(y) + \frac{m-k}{AN} \left(pI(0) + q\left[y s_{k+1}(y) + I(y) \right] \right) \tag{5}$$

with the border condition $s_m = 0$. In this equation, $s_k(y)$ stands for the case where the action has already been explored by the player. The $I(0)$ term corresponds to the occurrence of a collision. The $s_{k+1}(y)$ corresponds to the case where the new potential is smaller than y. Finally, the $I(y)$ term corresponds to an actual move, which increases the potential but resets the list of explored actions.

As $\frac{N+k}{AN} \neq 1$, we can simplify the equation above into:

$$s_k(y) = \frac{AN}{m-k} + pI(0) + q\left[y s_{k+1}(y) + I(y) \right] . \tag{6}$$

Proposition 6. *By induction over $k \in [\![0, m]\!]$, we obtain the following expression:*

$$s_{m-k} = AN \times v_k(yq) + \left[pI(0) + qI(y) \right] g_k(yq) . \tag{7}$$

In particular, for $k = m$, as $s_0 = -I'$, we obtain the following differential equation on I:

$$I'(y) + q \times g_m(yq) I(y) + AN \times v_m(yq) + p \times g_m(yq) I(0) = 0 . \tag{8}$$

Hence, as $I(1) = 0$ by definition of the integral, we have a boundary condition so we can explicitly solve this system:

$$I(y) = \int_y^1 \exp\left[q \int_y^u g_m(vq)dv\right] \times (pg_m(uq)I(0) + AN \times v_m(u))\, du\,. \tag{9}$$

Let us now introduce the function $h_k(y) = \sum_{j=1}^{k} \frac{y^j}{j} = q\int_0^y g_k(y)$. By taking $y = 0$ in the previous equation and making the substitution $z = qu$, we finally obtain the following equality:

$$\begin{aligned} I(0) &= \frac{AN}{1-\frac{p}{q}\left(e^{h_m(q)}-1\right)} \times \int_0^1 e^{h_m(qu)} v_m(qu)du\,, \\ &= \frac{AN}{1-p\times e^{h_m(q)}} \times \int_0^q e^{h_m(z)} v_m(z)dz\,. \end{aligned} \tag{10}$$

Finally, using Wald's lemma, $\mathbb{E}\left[\mathcal{T}_{\text{IFA}}\right] = \frac{\delta}{\ln\left(\frac{1}{q}\right)} I(0)$. This function as a whole is not simple to study.

6.2 Numerical Simulations

Now that we have a relatively simple formula (computationally at least), let us try to perform a few calculations to estimate the actual minimum of the function with respect to λ for a given value of m. For this task, we will make use of Python's `scipy.optimize` library in order to minimise the function described above. This yields the results shown in Figure 2.

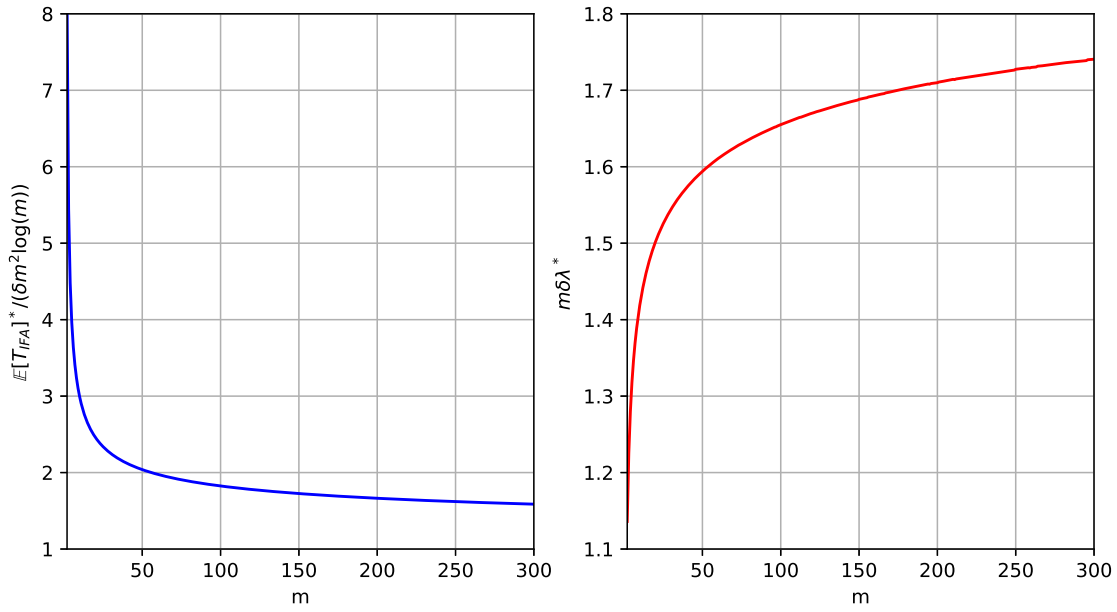

Figure 2: Minimum of $\frac{\mathbb{E}[\mathcal{T}_{\text{IFA}}]}{\delta \times An \times m\ln(m)}$ (left) and optimal parameter $m\delta\lambda^*$ (right) as functions of m.

Let us first have a look at the left part of the figure. Here is plotted $\frac{\mathbb{E}[\mathcal{T}_{\text{IFA}}]}{\delta \times AN \times m\ln(m)}$ as a function of m. This function seems to be decreasing, which strongly suggests $\mathbb{E}\left[\mathcal{T}_{\text{IFA}}\right] = O\left(\delta m^2 \ln(m)\right)$. Considering the allure of the plot, we may even have a $\frac{1}{\ln(\ln(m))}$ factor, converging to 0 really slowly, which is not worth the struggle for now.

If we look at the right part, we can see $m \times \delta\lambda^*$ as a function of m. This time, the function is increasing, a bit faster, but still at a sub-polynomial speed. For the rest of this section, we will instead replace this increasing function by a constant $C > 0$, so that our parameter becomes $\lambda^* \cong \frac{C}{\delta m}$. Consequently, we have $q^* = \exp\left(-\frac{C}{m}\right) \leq 1$ which converges to 1 really fast as $m \to \infty$.

6.3 Upper Bound of the Integral

By positivity, computing an upper bound on the integral $\int_0^1 e^{h_m(u)} v_m(u) du$ will also upper bound the integral in Equation 10 for any choice of $q \leq 1$. Using q^* as defined above to numerically estimate the integral results in something really close to what is obtained using $q = 1$, which means that in practice this upper bound using $q = 1$ will be pretty tight.

Now, to find an upper bound on this integral, let us split the integration domain into two parts using a threshold $(1-\theta) \in]0,1[$. On the right side, as the functions are increasing and $h_m(1) = v_m(1) = H_m$, we have $\int_{1-\theta}^1 e^{h_m(u)} v_m(u) du \leq \theta e^{H_m} H_m$. On the left side, we need now to obtain good bounds on v_m and h_m close to 0.

For h_m, notice that $m \mapsto h_m$ is increasing, and $h_\infty(y) = -\ln(1-y)$ on $[0,1[$. Hence, $e^{h_m(u)} \leq \frac{1}{1-u}$. For v_m, it is a tad harder to obtain a bound. Notice that, for a given value of $y \in]0,1[$, $j \mapsto y^{m-j}$ is increasing and $j \mapsto \frac{1}{j}$ is decreasing. Finally, using Chebyshev's sum inequality, we obtain the following bound:

$$v_m(y) \leq \frac{1}{m}\left(\sum_{j=1}^{m} \frac{1}{j}\right) \times \left(\sum_{j=1}^{m} y^{m-j}\right) = \frac{H_m}{m} g_m(y) \leq \frac{H_m}{m(1-y)} . \tag{11}$$

These upper bounds are clearly not tight close to 1, as they tend to $+\infty$. However, this bound is good enough under the threshold:

$$\int_0^{1-\theta} e^{h_m(u)} v_m(u) du \leq \frac{H_m}{m} \int_0^{1-\theta} \frac{1}{(1-u)^2} du = \frac{H_m}{m}\left[-\frac{1}{u}\right]_\theta^1 \leq \frac{H_m}{m\theta} . \tag{12}$$

As $e^{H_m} = e^\gamma m\left(1 + O\left(\frac{1}{m}\right)\right)$, by putting together the previous bounds, we obtain $\int_0^1 e^{h_m(u)} v_m(u) du \leq H_m\left(m\theta \times e^\gamma\left(1 + O\left(\frac{1}{m}\right)\right) + \frac{1}{m\theta}\right)$.

Proposition 7. *By taking the optimal value $\theta^* = \frac{1}{me^{\frac{\gamma}{2}}}$ in the expression above, we obtain the following upper bound on the integral:*

$$\int_0^1 e^{h_m(u)} v_m(u) du \leq 2e^{\frac{\gamma}{2}} H_m\left(1 + O\left(\frac{1}{m}\right)\right) = O\left(\ln(m)\right) . \tag{13}$$

Theorem 4. *In the case without collision, when $p \to 0$, we obtain in particular:*

$$E\left[T_{IFA}\right] = AN \int_0^1 e^{h_m(u)} v_m(u) du \leq 4e^{\frac{\gamma}{2}} m H_m\left(1 + O\left(\frac{1}{m}\right)\right) . \tag{14}$$

6.4 Upper Bound of the Fraction

Our current upper bound on the convergence time of IFA is:

$$\mathbb{E}\left[\mathcal{T}_{\text{IFA}}\right] \leq 2e^{\frac{\gamma}{2}} \delta AN H_m \times \frac{1}{\ln\left(\frac{1}{q}\right)\left(1 - (1-q) \times e^{h_m(q)}\right)} \times \left(1 + O\left(\frac{1}{m}\right)\right) . \tag{15}$$

We won't be able to find a uniformly good upper bound which does not depend on q, as too long time steps ($q \to 1$) or too many collisions ($q \to 0$) will result on an infinitely long game on average. What is left to do is to optimise this central fraction with respect to q, to find a choice of λ which guarantees a close to optimal outcome on average.

Once again, a precise study of this function for any $q \in]0,1[$ seems like an unreachable ideal. Graphically, it appears to be convex, but after a lot of trial and error I learned that proving it is an altogether different story. This is why we will instead obtain a good upper bound on the minimum using the parameter $\lambda^* = \frac{C}{\delta m}$ and thus $q^* = e^{-\frac{C}{m}}$.

Naturally, we have $\frac{1}{\ln\left(\frac{1}{q}\right)} = \frac{m}{C}$. The other half of the fraction is a bit harder to deal with, and we will in fact prove that $(1-q)e^{h_m(q)}$ converges to a constant strictly smaller than one.

The main idea is to make a Riemann sum appear:

$$\begin{aligned} h_m(q) &= \sum_{j=1}^{m} \frac{e^{-j\times\frac{C}{m}}}{j} = \sum_{j=1}^{m} \frac{1}{m} \times \frac{e^{-C\times\frac{j}{m}}-1}{\frac{j}{m}} + H_m \,, \\ &= -\int_0^1 \frac{1-e^{-Cx}}{x} dx + \ln(m) + \gamma + O\left(\tfrac{1}{m}\right) \,. \end{aligned} \tag{16}$$

We know that $\int_0^1 \frac{1-e^{-Cx}}{x} dx = \ln(C) + \gamma + \Gamma(0, C)$, where $\Gamma(a,z) = \int_z^\infty t^{a-1}e^{-t}dt$ is the incomplete Gamma function. Thus, we have:

$$\begin{aligned} (1-q)e^{h_m(q)} &= \frac{C}{m}\left(1+O\left(\tfrac{1}{m}\right)\right) \times \exp\left(-\Gamma(0,C) + \ln\left(\tfrac{m}{C}\right) + O\left(\tfrac{1}{m}\right)\right) \,, \\ &= e^{-\Gamma(0,C)} \times \left(1+O\left(\tfrac{1}{m}\right)\right) \,. \end{aligned} \tag{17}$$

As $C > 0$, we have $\Gamma(0,C) \in \mathbb{R}^{+*}$ and thus $e^{-\Gamma(0,C)} \in]0,1[$.

Finally, putting everything together, for a given value of C, we obtain the bound:

$$\mathbb{E}\left[\mathcal{T}_{\text{IFA}}\right] \leq \frac{2e^{\frac{\gamma}{2}}\delta}{C \times \left(1-e^{-\Gamma(0,C)}\right)} \times AN \times mH_m \times \left(1+O\left(\frac{1}{m}\right)\right) \,. \tag{18}$$

Using numerical estimations, the best choice for C, which minimises its contribution, is roughly $C = 0.63$. This yields $\frac{1}{C\times\left(1-e^{-\Gamma(0,C)}\right)} \leq 4.56$, hence the following approximate asymptotic bound:

$$\mathbb{E}\left[\mathcal{T}_{\text{IFA}}\right] \leq 12.18 \times \delta m^2 H_m \,. \tag{19}$$

7 Numerical Comparison Between BBD and IFA

Putting the exact value of the multiplicative factor aside, we have obtained a choice for the clock parameter λ^* which guarantees $\mathbb{E}\left[\mathcal{T}_{\text{IFA}}\right] = O\left(\delta m^2 \ln(m)\right)$. By keeping the lower bound $\mathbb{E}\left[\mathcal{T}_{\text{IFA}}\right] = \Omega\left(\frac{\delta m^2 \ln(m)}{\ln(\ln(m))}\right)$ in mind, we conclude this choice of λ^* is close to optimal, and that in any case the optimal will satisfy $\mathbb{E}\left[\mathcal{T}_{\text{IFA}}\right] = \tilde{\Theta}\left(\delta m^2\right)$.

Currently, only the lower bound $\mathbb{E}\left[\mathcal{T}_{\text{BBD}}\right] = \Omega\left(\frac{\delta m^2 \ln(m)}{\ln(\ln(m))}\right)$ has been properly proved for BBD. I initially hoped I could follow again the tracks of [Dur18] and their coupling with BRD with collisions, but by trying to do so I encountered some critical gaps which invalidate it in its current state. Unfortunately, whether in the case of BRD or my study of BBD, we have not managed yet to repair the coupling.

If we are to explicitly obtain a coupling useful for our theoretical purpose, the two processes should exhibit a somewhat similar behaviour stochastically. This is why I used the `numpy.random.RandomState()`

container to use a common pseudo-random sequence of potentials (resp. players, actions, collisions) on a run of BBD and IFA, by effectively implementing the coupling described in Section 3. The Python code for BBD can be found in Appendix A.

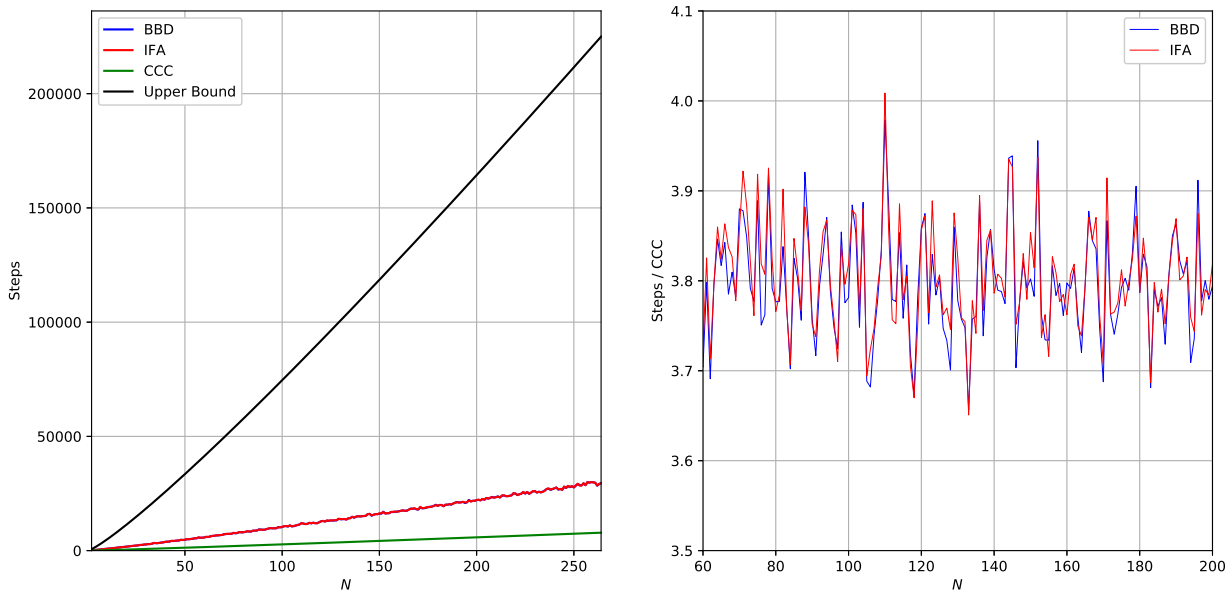

Figure 3: On the left, average values of T_{BBD} and T_{IFA}, over 400 samples, compared to our lower bound on CCC and upper bound on IFA, as functions of N, using $A = 10$ and $q^* = e^{-\frac{0.63}{m}}$. On the right, comparison of T_{BBD} and T_{IFA} normalised by the lower bound.

Let us now have a look at a few numerical simulations on Figure 3, to convince ourselves that the current coupling is already quite good. In the dataset used here, the processes T_{BBD} and T_{IFA} are coupled as described in Section 3. As expected, the lower bound on Clumsy Coupon Collector indeed appears to be a lower bound on the number of steps for the other processes as well. The empirical means of T_{BBD} and T_{IFA} appear to be undistinguishable on the left figure. On the right figure, which puts the focus on the normalised ratios between these quantities and CCC, we can see that there is a strong correlation between the two processes up to the random noise – the difference between these empirical averages and their actual mean values – but that none appears to be consistently bigger than the other one. Random noise aside, the average values of these two ratios seems to be relatively constant, meaning that the actual values of $\mathbb{E}[T_{\mathrm{BBD}}]$ and $\mathbb{E}[T_{\mathrm{IFA}}]$ are proportional to $\frac{m \ln(m)}{\ln(\ln(m))}$. We can observe a likewise behaviour in Figure 4 by making A vary instead of N. Naturally, unlike for IFA and BBD, the ratio for our upper bound is slowly increasing at a $\ln(\ln(m))$ speed.

Now, the previous comparison between BBD and IFA was only on average. As we can see in Figure 5, there is a distinctly noticeable subset of simulations for which BBD and IFA use the same amount of steps. This subset represents about 85% of the samples, and is much bigger that the subset of samples for which $1 \leq |T_{\mathrm{IFA}} - T_{\mathrm{BBD}}| \leq 1000$, which represents less than 0.1% of the samples. In the absence of any collision or intersection, IFA resets the players' knowledge a bit more often than BBD, which should yield a consistently greater number of steps. At the same time, intersections in the absence of collisions slow down BBD relatively to IFA. These two phenomenons are the reason why this strong observed incentive to have the exact same number of steps seems quite curious and unexpected to me. These considerations aside, we obtain a relatively symmetric distribution with respect to the diagonal. If the

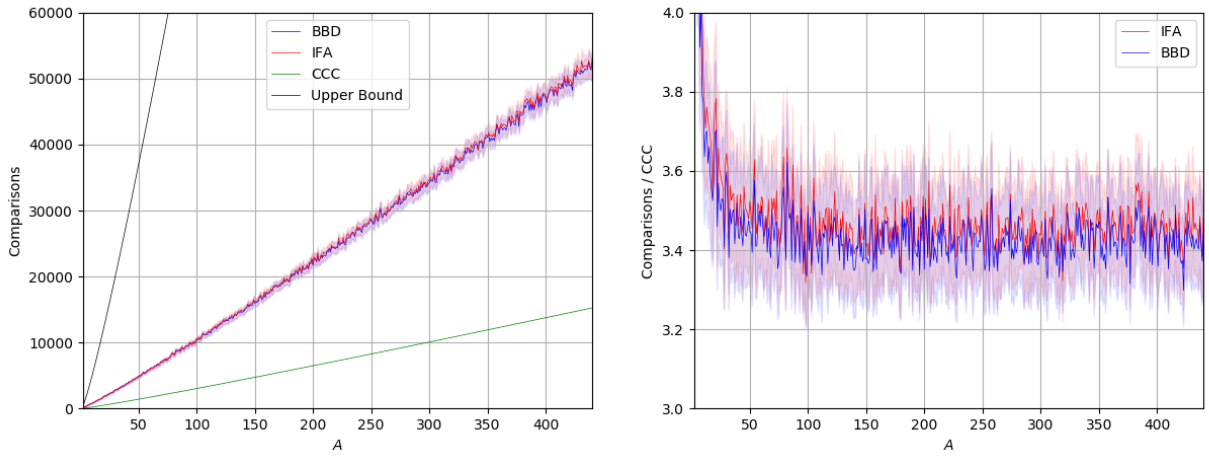

Figure 4: On the left, average values of T_{BBD} and T_{IFA}, over 500 samples, compared to our lower bound on CCC and upper bound on IFA, as functions of A, using $N = 10$ and $q^* = e^{-\frac{0.63}{m}}$. On the right, comparison of T_{BBD} and T_{IFA} normalised by the lower bound. In both cases, a 99% confidence interval is displayed for IFA and BBD.

two samples were purely independent, we would obtain a somewhat "square" pattern. Here, however, the dots are relatively concentrated around the diagonal line, with a roundish shape, which corresponds to the fact the coupled behaviours are pretty similar at first, and only the latter part of their trajectories differs in a chaotic way. The symmetry of the pattern tends to indicate that there is no huge bias for IFA nor BBD, which in turns seems to indicate that the current coupling between the two processes will be insufficient to establish a proper theoretical bound on $\mathbb{E}[\mathcal{T}_{\mathrm{BBD}}]$.

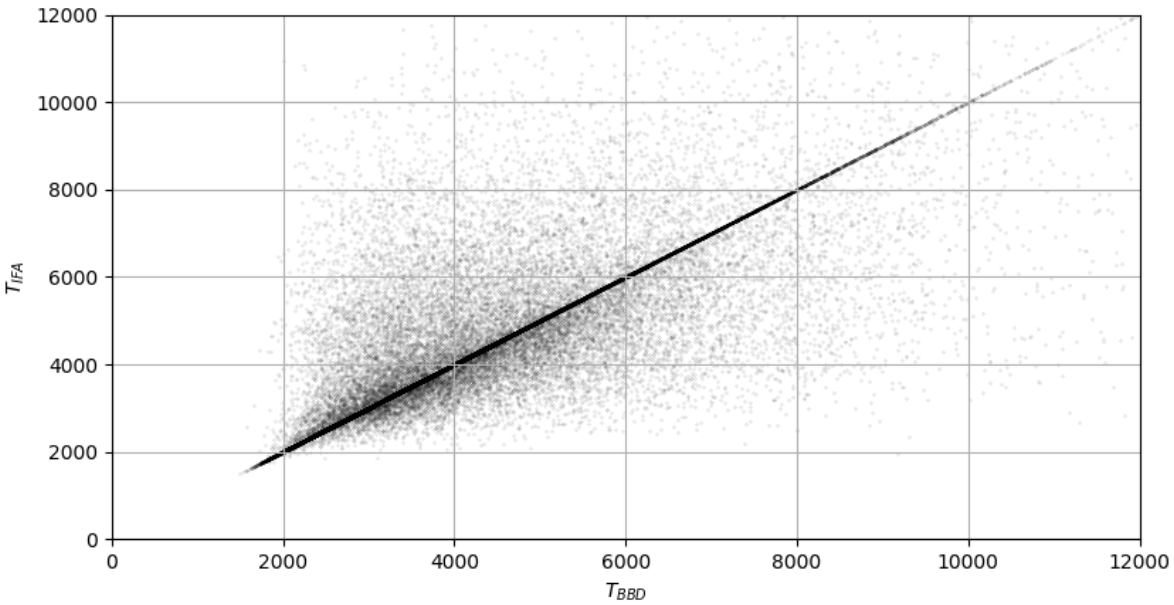

Figure 5: Scatter plot of 144000 samples of the coupled process $(T_{\mathrm{BBD}}, T_{\mathrm{IFA}})$, using $N = A = 20$ players and actions and and $q^* = e^{-\frac{0.63}{m}}$.

8 A Better Clock for BRD

Let us now shift the topic a bit from the black box algorithm, back to the best response dynamics. Up until now, because of the overwhelmingly theoretical study, we used Poisson clocks. This typically embeds a coupon collector problem into the dynamics, which in turn adds a $\ln(N)$ multiplicative factor to the complexity of BRD, with respect to a round-robin centralised setting, as seen in Theorem 1.

Out of curiosity, we tried using some alternative clocks for BRD. The clocks that performed best in practical situations were the most naive ones: uniform noisy clocks. More precisely, each player follows a clock for which the time between two ticks follows a law $\tau \times (1 + \theta \times \mathcal{U}([-1, 1]))$, where τ is the average waiting time and θ the amount of noise. In the noiseless case $\theta = 0$, we obtain a purely periodic clock, hence a distributed round-robin setting, which poses a simple problem: if two player use the same frequency and play too close to each other, they will always be in collision, at each and every tick of their clocks. To avoid this problem, it is natural to add some noise to allow players to avoid eternal collisions.

Intuitively, τ must be big enough so that there aren't too much collisions, but small enough so that we still have to wait at most a a linear amount of time once the Nash equilibrium has been found. Likewise, θ should be small enough to get closer to a round-robin behaviour, but big enough to disperse the eventual collisions. In this framework, for a given value of A, the ratio $\frac{\mathbb{E}[\mathcal{T}]}{N}$ seems to have roughly the same value independently of N. Empirically, it appears that too small values of θ have a nefarious effect on the mean number of steps, but not really $\theta = 1$, as we can see in Figure 6.

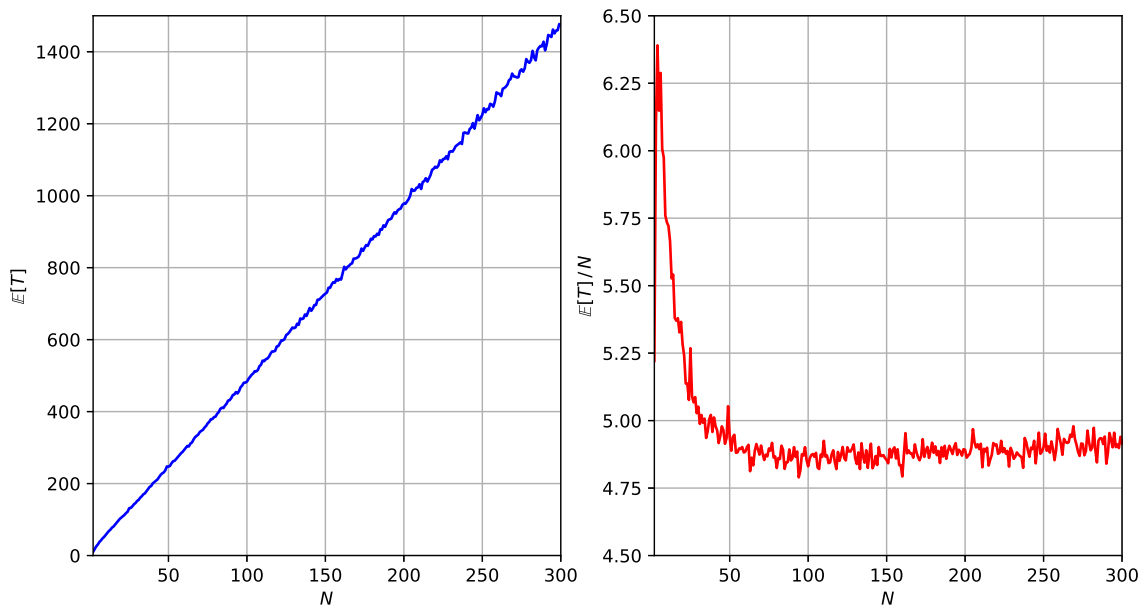

Figure 6: Average value of the convergence time $\mathcal{T}$ (left) and the ratio $\frac{T}{N}$ (right) over 1000 samples using parameters $\tau = N$ and $\theta = 1$, with $\delta = 1$.

Let us now estimate the somewhat worst-case probability of collision, to get a least a rough idea of why big values of θ still work relatively well. The first thing we would like to avoid is for another player j to be in collision twice with the same tick of k. This means that we want at most one tick of j in a given interval of length 2δ. Considering the minimal time between two ticks, this yields the condition $\tau(1-\theta) \geq 2\delta$. As we will reasonably have $\theta = o(\tau)$, this is not a harsh constraint when the amount of players gets large.

Now, what is the worst possible case for player k in terms of collisions? For each player, only one tick of their respective clock could collide with k, if it happens in the interval $[-\delta, \delta]$. Hence, the worst case would for each player j to have the collision window of size 2δ fully included in their target interval of size $2\theta\tau$, hence a collision probability $\frac{\delta}{\theta\tau}$ (this requires $\theta\tau \geq \delta$). As each player has a clock independent of the others, this means that, conditionally on the value of player k's tick time, the worst case collision probability is $1 - \left(1 - \frac{\delta}{\theta\tau}\right)^{N-1}$.

Now, consider $\tau = \frac{\delta N}{C}$ for some constant $C > 0$. This choice of τ needs a linear time to stop once a Nash equilibrium has been reached, which is what we desire. Using this value of τ, and assuming θ does not go to 0:

$$1 - \left(1 - \frac{\delta}{\theta\tau}\right)^{N-1} = 1 - \exp\left((N-1)\ln\left(1 - \frac{C}{\theta N}\right)\right) = 1 - \exp\left(-\frac{C}{\theta} + O\left(\frac{1}{N}\right)\right). \tag{20}$$

Taking a bigger value for C lowers the worst-case collision probability, but at the expense of a longer time to stop once Nash is reached. However, taking a bigger value for θ not only helps to lower this worst-case collision probability, the probability to *stay* in a bad case, but also keeps the probability to *enter* in a collision under control.

As seen in [Dur18], using the collision rule of BRD with Poisson clocks, a constant collision probability at each tick is sufficient to reach a reasonable $O(N\ln(N))$ value for $\mathbb{E}[\mathcal{T}_{\text{IFA}}]$. Thus, seeking the same property for these clocks seems reasonable. Be aware however that, while this worst-case study makes sense heuristically, it is only a heuristic. The study done with Poisson clock fully exploits the memoryless behaviour of the ticks, which induces a natural way to analyse the process. Here, say players j and k both previously played at time 0, and the current time is $\tau(1+\theta) - \delta$. Then we are certain that both players will play again in less than δ, hence we are certain that another collision will occur. From a centralised viewpoint, a collision may be a certain event in the close future at some points in time. To overcome this obstacle, other analysis methods are needed.

With a bit more time, I would have adapted my simulations from BRD to BBD, to see if the convergence time becomes a $O(m^2)$. With much more time, a more in-depth theoretical study of these clocks would be an interesting topic, perhaps to see how they relate to the round-robin framework they aim to approximate.

9 Local Best Response (LBR)

I also worked on another variation of BRD, which we called Local Best Response (LBR). In this new framework, the actions of a player are the vertices of a graph, and the player can only see the neighbouring actions, and will choose the best response out of these neighbours until they reach a local maximum of the potential on their action graph. This model can be justified by a form of laziness from the players: in a car routing problem, a lazy agent may consider a path slightly different from their current route, but not an altogether different one. Best Response corresponds to the special case where every player has a complete graph over their actions.

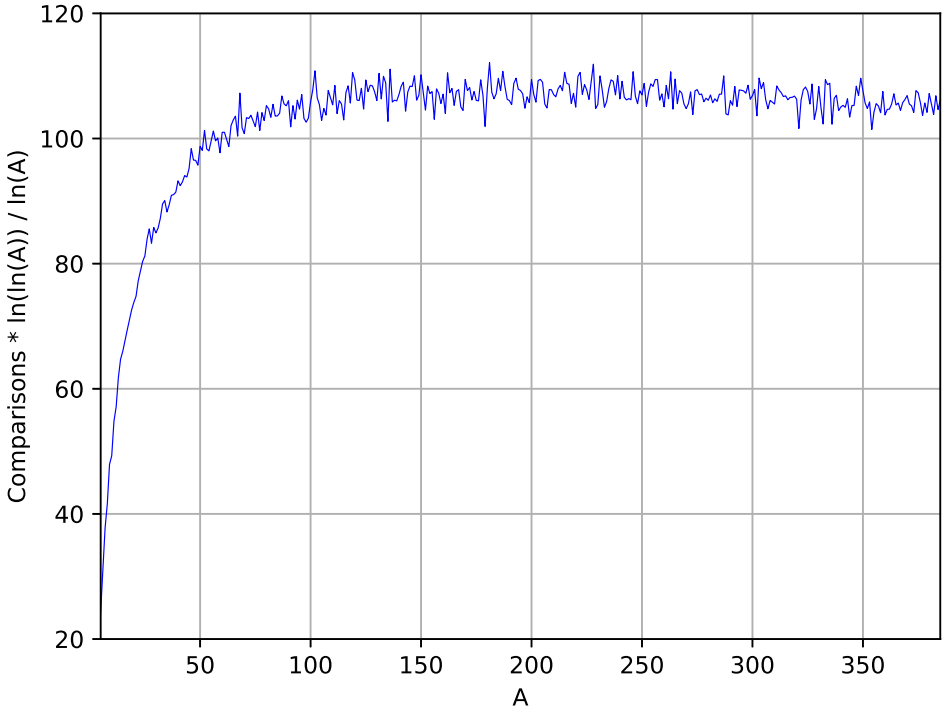

Figure 7: Average number of comparisons for LBR over 1000 samples, as a function of A, using $N = 10$ players and an average degree $D = 10$.

To simulate the LBR algorithm, we generated a random graph for each player at the beginning of the game using a variant of the configuration model with independent random clustering coefficients following an exponential distribution, so that the average degree is fixed to D even as $A \to \infty$. In my current implementation, the players checks their surrounding actions at each one of their turns, even if they should know they are satisfied.

Using a complete graph, we perform $D = A - 1$ comparisons at each steps, and $N \ln(N)$ steps on average as stated in Theorem 1, thus $O(AN \ln(N))$ comparisons overall. If the graphs are generated independently at each step, for $A \gg D$, we have an average number of comparisons by step proportional to D. Hence, the best we may hope for is a $O(DN \ln(N))$.

However, we can see in Figure 7 that the number of comparisons seems to grow logarithmically with A, thus our best guess for the complexity would be $O(D \ln(A) N \ln(N))$. This may come from the fact that each jump from one action to a neighbouring one corresponds to a transition for the uniform random walk, which induces a bias towards higher degrees if we look at its invariant distribution.

This model is still quite fresh, so there is a lot of things to study and it would greatly benefit from exchanges with people specialised on (random) graphs, at the very least to choose a graph model representative of the kind of networks used as a motivation for LBR.

10 Perspectives

The last two sections correspond to much less explored landscapes, and each of them would deserve a proper in-depth theoretical study, as we can observe a noticeable decrease of the complexity in comparison to the typical best response algorithm with Poisson clocks.

More importantly, we have seen a full study of the average complexity of the intersection-free approximation of BBD. The most crucial missing step now would be to establish a proper coupling between BBD (resp. BRD) and its approximation, to properly conclude the theoretical study of BBD. As we have seen in Section 7, the coupling used for the centralised case still yields a positive correlation between IFA and BBD in a distributed framework, but its theoretical properties are insufficient to deduce proper relations between T_{BBD} and T_{IFA}. The next milestone would now be to find a new coupling, with properties specifically adapted to the collisions and intersections of the distributed framework.

References

[BDML06] Tom Britton, Maria Deijfen, and Anders Martin-Löf. Generating simple random graphs with prescribed degree distribution. *Journal of Statistical Physics*, 124(6):1377–1397, Sep 2006.

[CK05] George Christodoulou and Elias Koutsoupias. The price of anarchy of finite congestion games. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 67–73, New York, NY, USA, 2005. ACM.

[Dur18] Stéphane Durand. *Analysis of Best Response Dynamics in Potential Games*. PhD thesis, École doctorale Mathématiques, Sciences et technologies de l'information, Informatique (MSTII), 2018.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[MM56] Christopher B. Beckmann Martin; McGuire, C. B.; Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[MS96] Dov Monderer and Lloyd S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124 – 143, 1996.

[Pol04] G. Polya. *How to Solve It: A New Aspect of Mathematical Method*. Princeton Science Library. Princeton University Press, 2ed. edition, 2004.

[Ros73] Robert W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67, Dec 1973.

[Rou05] Tim Roughgarden. *Selfish routing and the price of anarchy*, volume 174. MIT press Cambridge, 2005.

A Implementation of BBD with Collisions in Python

```
import numpy as np

def Code(x,A): # Dictionaries cannot use arrays as indices, hence the conversion to a codeword
    c=0
    for b in x :
        c = b+A*c
    return c

def Phi(x,A,S,POTENTIAL):
    c = Code(x,A)
    if c not in S :
        S[c] = POTENTIAL.uniform()
    return S[c]

def BBD(A,A_seed,N,N_seed,Phi_seed,Col_prob,Col_seed):
    # Here we define the pseudo-rng used by the algorithm
    # so that IFA and BBD can use the same random sequence
    POTENTIAL = np.random.RandomState()
    ACTION    = np.random.RandomState()
    PLAYER    = np.random.RandomState()
    COLLISION = np.random.RandomState()
    POTENTIAL.seed(Phi_seed)
    ACTION.seed(A_seed)
    PLAYER.seed(N_seed)
    COLLISION.seed(Col_seed)
    S   = {}                                      # Dictionary of explored states
    x   = [ 0 for k in range(N) ]                 # Initial state
    L   = set()                                   # Set of satisfied players
    M   = [ set([0]) for k in range(N) ]          # Set of explored actions for each player
    phi = [ Phi(x,A,S,POTENTIAL) for k in range(N) ] # Last potential for each player
    T   = 0                                       # Amount of steps

    while len(L) < N :
        k = PLAYER.randint(0,N) # Random player in this turn
        b = ACTION.randint(0,A) # Random action in this turn
        collide = ( COLLISION.uniform() < Col_prob ) # Boolean event of a collision in this turn

        if Phi(x,A,S,POTENTIAL) != phi[k] : # If the state changed, we reset the player's knowledge
            phi[k] = Phi(x,A,S,POTENTIAL)
            M[k] = set([x[k]])
        if b not in M[k] : # If b was previously explored by k, the turn stops, no collision can occur
            y = x.copy()
            y[k] = b
            if collide :
                x[k] = b
                phi[k] = Phi(x,A,S,POTENTIAL)
                L = set()
                M[k] = set()
            elif Phi(y,A,S,POTENTIAL) > phi[k] : # If there is no collision then turn plays is it should
                x = y
                phi[k] = Phi(x,A,S,POTENTIAL)
                L = set()

        M[k].add(b)
        if len( M[k] ) == A :
            L.add(k)
        T += 1
    return T
```