Policy Iteration is well suited to optimize PageRank

Romain Hollanders[*][†]
Jean-Charles Delvenne[*][‡]
Raphaël Jungers[*] [§]

June 2011

Abstract

The question of knowing whether the policy Iteration algorithm (PI) for solving Markov Decision Processes (MDPs) has exponential or (strongly) polynomial complexity has attracted much attention in the last 50 years. Recently, Fearnley proposed an example on which PI needs an exponential number of iterations to converge. Though, it has been observed that Fearnley's example leaves open the possibility that PI behaves well in many particular cases, such as in problems that involve a fixed discount factor, or that are restricted to deterministic actions. In this paper, we analyze a large class of MDPs and we argue that PI is efficient in that case. The problems in this class are obtained when optimizing the PageRank of a particular node in the Markov chain. They are motivated by several practical applications.

We show that adding natural constraints to this PageRank Optimization problem (PRO) makes it equivalent to the problem of optimizing the length of a stochastic path, which is a widely studied family of MDPs. Finally, we conjecture that PI runs in a polynomial number of iterations when applied to PRO. We give numerical arguments as well as the proof of our conjecture in a number of particular cases of practical importance.

Introduction

In search engines, it is critical to be able to compare webpages according to their relative importance, with as few as possible computational resources. This is done by computing the *PageRank* of every webpage from the web [BP98] : pages with higher PageRank will then appear higher in the list of results. To compute this PageRank, the first step is to model the web as a digraph in which the webpages are represented by nodes and the links between them are represented by directed edges. Then, the PageRank of a node is defined as the average portion of time spent in that node during an infinite and uniform random walk on the graph. This random walk can be seen as the infinite process of a random surfer that, from its current page, picks up any available outgoing link with uniform probability and jumps to the page pointed by that link.

The utility of PageRank is not limited to search engines and it has been proposed in several other applications such as financial market, spam detection, web-crawling, semantic networks, and many others. It can also be used in any application that requires ranking nodes in order of relative importance. See [Ber05] for a survey on PageRank and its applications. The introduction of the concept of PageRank has also generated a large number of questions and challenges.

Among these, the problem of optimizing the PageRank of webpages raises increasing interest, as evidenced by the growing literature on the subject [AL04, MV06, dKND08, IT09, CJB09, FABG10]. It is also of great practical interest and well-studied in the engineering community where good practice methods are developed to ensure a high PageRank [CLF09]. PageRank Optimization (PRO) is also the focus of this paper. Here, we study how to maximize (or minimize) the PageRank of some target node when control is granted on some subset of edges, meaning that some edges (called the *free edges*) may be chosen to be activated or deactivated. A typical example of PRO is the so-called *webmaster problem* in which a webmaster tries to maximize the PageRank of one of his webpage by determining which links under his control (i.e. on his website, or on an allied website for instance) he should activate and which links he should not [AL04, dKND08]. Furthermore, the same tools may be used to find how much the PageRank of some nodes can vary when the presence or absence of some links, called *fragile links*, is uncertain (e.g. because a link is broken, the server is down or because of traffic problems) [IT09].

The main difficulty to solve PRO is to deal with the exponential number of possible free edges configurations : since each edge has two possible states - on or off - the number of possible configurations is 2^f, where f is the number of free edges. To escape this difficulty, Ishii and Tempo first proposed an approximate algorithm that would find an interval containing the minimum and maximum PageRank of the target node [IT09]. One year later, Csáji et al. proposed a way of formulating the problem as a *Stochastic Shortest Path* problem (SSP) - which is a subclass of *Markov Decision Processes* (MDPs) - thereby showing that an exact solution of the problem could be found in weakly polynomial time using linear programming [CJB09]. (For some refinements to PRO, see also [FABG10]. For more on SSPs and MDPs see e.g. [Put94], [BT91] and [Ber07].) Yet in practice, MDPs (and thus also SSPs) are solved much more efficiently using algorithms adapted to their special structure. Among these algorithms, *Policy Iteration* (PI) [How60] performs amazingly well and is guaranteed to converge to the optimal solution in a finite number of iterations. However, even though PI usually converges in few iteration, theoretical upper and lower bounds on its complexity are exponential in many cases. The main goal of this paper is to show that the existing exponential lower bounds, as they are, should not apply to PRO. Instead, we believe that polynomial upper bounds exist in that case.

There is a significant research effort for understanding the complexity of PI. Let us quickly review existing complexity results. For general MDPs, the best upper bound - $O(2^m/m)$ - is due to Mansour and Singh [MS99], where m designates the number of choices to be made in the problem, while the largest lower bound is also exponential and has recently been found by Fearnley through a carefully built example [Fea10]. This was a breakthrough after 50 years of research on the question of the complexity of PI. The story is different for discounted MDPs (i.e. a class of MDPs in which the impact of future costs are progressively reduced by some discount factor) for which a first strongly polynomial upper bound has recently been found by Ye [Ye10] and then further improved by Hansen et al. [HMZ10], yet only for fixed discount factors. Though, even if upper and lower bounds seem to meet in both cases, the story does not end here. Indeed, Fearnley's example is impossible to adapt to some other important particular cases of MDPs. This is for example the case for Deterministic MDPs (DMDPs) for which the best lower bound currently known is quadratic and has been found by Hansen and Zwick [HZ10]. Besides, strongly polynomial time algorithms exist to solve that problem [MTZ10] (including PI when fixed discount is included to the problem [Ye10]), which hints that DMDPs might be easier to solve than general MDPs.

In this work, we argue that PRO might be another particular case for which a polynomial reduction of Fearnley's example is not possible. We first show how a natural generalization of PRO makes it equivalent to general SSPs and vice versa, giving a new point of view on SSPs (and MDPs). Then, we identify the exclusive nature of the choice of actions in an SSP as the

main constraint that makes it different and most probably harder to solve than PRO. Based on extensive numerical computations, we then conjecture that PI converges in a polynomial number of iterations in the case of PRO. We also give a number of particular cases in which we show that PI converges in polynomial time. In this work, we try to make a step towards deeper insight on the existing link between the properties of an MDP instance and the resulting efficiency of PI.

The paper is organized as follows. In Section 1, we give formal definitions for SSP and PRO and we generalize PRO in a natural way. In Section 2, we show how that generalization of PRO can be transformed into an SSP and vice versa. In Section 3, we conjecture that PI should perform well when applied to PRO, arguing with numerical evidence. Then, in Section 4, we give a number a particular cases of PRO for which PI behaves well.

1 Definitions

In this section, we give a formal definition for Stochastic Shortest Path and for PageRank Optimization problems. We also formulate a natural generalization of the latter.

Stochastic Shortest Path. An instance of the *Stochastic Shortest Path* problem (SSP) is a tuple $(\mathcal{S}, \mathcal{U}, \mathcal{P}, \mathcal{C})$ where $\mathcal{S}$ is the finite set of *states*, U is the finite set of all *actions* and $\mathcal{U}_s \subseteq \mathcal{U}$ is the set of actions available in state $s \in \mathcal{S}$ (there is at least one action for each state), and $\mathcal{P}_{s,s'}^u$ and $\mathcal{C}_{s,s'}^u$ are respectively the *transition probability* of going from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ when choosing action $u \in \mathcal{U}_s$ and the (real-valued) *cost* incurred by this displacement [BT91]. We also ask for the transition probabilities to be non-negative and to sum to one, namely $\sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^u = 1$, for all starting state $s \in \mathcal{S}$ and action $u \in \mathcal{U}_s$. An action is said *probabilistic* if it includes randomization between several arrival states, whereas it is said *deterministic* otherwise.

In SSP, we consider the random process of an *agent* that starts at some starting state s_0 and then jumps to a new available state at each time step (according to the action taken in its current state and the associated transition probabilities). The main feature of an SSP, compared to general MDPs, is that we assume the existence of an absorbing cost-free state τ (also called *target* state) that is required to be reachable with a non-zero probability path by every other state, whichever actions are chosen. In this context, the goal of the *controller* of the process is to choose the right action in each state in order to minimize the expected sum of costs incurred by the agent before reaching the target state, whatever the starting state. The choice of a unique action to take in each state is called a *policy* (or strategy) $\mu : \mathcal{S} \to \mathcal{U}$. The chosen policy is *proper* if the agent eventually reaches τ for any starting state. It is *improper* otherwise. A policy is *optimal* iff it is better at minimizing the controller's goal than any other policy, for any starting state. One fundamental result about MDPs that can be adapted to SSPs guarantees that there always exists an optimal (not necessarily unique) proper policy, provided that there exists at least one proper policy [Put94, Ber07]. Note that it is always possible to formulate an SSP problem as a linear program whose size is polynomial in the number of states and the maximum number of actions per state of the SSP instance.

PageRank Optimization. To define a *PageRank Optimization* problem (PRO), we first define its *support graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{V}' \cup v$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of directed edges of the graph and v is the target node for which we want to maximize (or minimize) the PageRank. For that task, control is granted on some subset $\mathcal{F} \subseteq \mathcal{E}$ of edges (called the set of *free edges*) in which we may choose to activate or deactivate any edge, whereas the edges in $\mathcal{E} \backslash \mathcal{F}$ are fixed and cannot be removed. The goal in PRO is to choose the right subset of free edges that we activate so that the PageRank of node v is maximal (or minimal). Here we focus on the maximization problem but a straightforward modification of the approach can be used to deal with the minimization problem as well.

PRO can be formulated as an SSP in polynomial time [CJB09] as follows. First observe that maximizing the PageRank of v (i.e. its frequency of visit by the random surfer) is equivalent to minimizing the average time between two visits of v. Let us split v into a starting node v_s and a target node v_t such that v_s has all outgoing links of v and v_t has all its ingoing links, plus a zero-cost self-loop. Maximizing the PageRank of v is then equivalent to minimizing the average distance from v_s to v_t. Observe that v_t is now an absorbing node. In this setting, an action is the choice between activation or deactivation for a given free edge and a policy is a subset of activated free edges. Therefore, PRO can be seen as a particular case of SSP where $\overline{\mathcal{V}} = \mathcal{V}' \cup \{v_s, v_t\}$ is the set of states and $\mathcal{F}$ is the set of $2f$ actions[1], where f is the number of free edges. This SSP instance has a polynomial number of states and actions. Uniform transition probabilities and unit costs are assumed here (except for the target node v_t which is cost-free).

If we do not assume that the support graph $\mathcal{G}$ is strongly connected, extra care must be taken. Indeed, in that case, there may be nodes (or connected components) that do not have any outgoing edge. To deal with such *dangling* nodes (or components), many techniques exist [Ber05], such as connecting them to every other node. We may choose any of the existing solution and assume that we have already dealt with dangling nodes. Another case for which we must be careful is when all outgoing edges from a node are free. Indeed, such a node would become a dangling node is every free edge was deactivated. However, in that case, Csáji et al. have shown that the optimal policy would always activate exactly one of these free edges, i.e. the one that points towards the node that is closest to the target node [CJB09]. Furthermore, an algorithm like PI would only consider policies in which exactly one of these free edges are active. Therefore, in this case, we may always consider that there are as many actions as free edges, where each action consists in activating exactly one of the free edges. As a consequence of the above, we may always consider that all nodes are able to reach the target node with positive probability, whatever the chosen policy (so that every policy is proper).

Generalized PageRank Optimization. A natural way of extending PRO is to allow arbitrary transition probabilities and costs. We call such a relaxation a *Generalized PageRank Optimization* problem (GPRO), which we now formally define. A convenient way to deal with arbitrary transition probabilities in the context of GPRO, where the out-degree of the nodes may vary, is to assign a weight to each possible transition and to compute the transition probabilities in proportion to these weights. More precisely, we define the weight set $\mathcal{W}$ such that $\mathcal{W}_{i,j} > 0$ if $(i,j) \in \mathcal{E}$ and $\mathcal{W}_{i,j} = 0$ otherwise. Given a policy μ (i.e. a configuration of free edges), we define the corresponding weight set $\mathcal{W}^\mu$ as

$$\mathcal{W}^\mu_{i,j} = \begin{cases} \mathcal{W}_{i,j} & \text{if } (i,j) \in \mathcal{E}^\mu, \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathcal{E}^\mu$ is the set of activated edges when policy μ is chosen. Transition probabilities $\mathcal{Q}^\mu$ are then defined according to the weights by :

$$\mathcal{Q}^\mu_{i,j} = \frac{\mathcal{W}^\mu_{i,j}}{\sum_{k \in \overline{\mathcal{V}}} \mathcal{W}^\mu_{i,k}}.$$

In a node i, weights enable to distribute the probabilities among the activated edges that leave i, and this in proportion to their mutual importance. Note that for $\mathcal{Q}$ to be well defined, there must always exist at least one edge with positive weight going out of any starting node i, for any policy μ. For any nodes i, j, a cost matrix $\mathcal{K}$ is also defined such that $\mathcal{K}_{i,j}$ is the cost of going from i to j.

---

[1]For precision, taking an action in a state in which k outgoing edges are free should be seen as choosing the subset of these k edges that should be activated. See Section 2 for a construction to deal with the size of these subsets in polynomial time.

Finally, in the GPRO framework, we also allow some exclusivity constraints in the following case : in a node in which there are only two free edges and no fixed edges going out, we may assume that exactly one of these edges must be activated while the other must be deactivated. We will see that these exclusivity constraints will enable us to make the link with SSP. We believe that such constraints are the key difference with PRO that makes the latter problem easier to solve.

Putting everything together, we define a GPRO instance by the tuple $(\overline{\mathcal{V}}, \mathcal{F}, \mathcal{W}, \mathcal{K})$. The original edge set $\mathcal{E}$ can be obtained from $\mathcal{W}$. Let us now make some comments about the introduced concepts.

Remarks.

1. A PRO problem can be formulated as a GPRO problem in which $\mathcal{W}_{i,j} = 1$ if $(i,j) \in \mathcal{E}$ and $\mathcal{W}_{i,j} = 0$ otherwise, and in which $\mathcal{K}_{i,j} = 1$ for all $(i,j) \in \mathcal{E}$ (except when $i = v_t$).

2. Exclusivity constraints can be modeled using small weights. Indeed, in a node where there is a free and a fixed edge, if the free edge has a weight significantly higher than the fixed edge, it means that this edge will be chosen with high probability if activated, while the fixed edge will always be chosen with probability one if the free edge is deactivated : so depending on the activation state of the free edge, one edge or the other will be chosen, which imitates the exclusive behavior of SSP actions, as illustrated in figure 1. However at that point, we have not been able to adapt such a model to any instance with the guarantee that the optimal solution would not change, at least not with weights that have polynomial value.
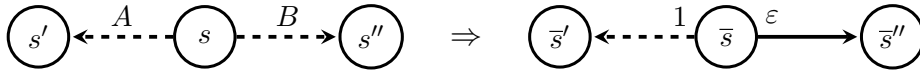

Figure 1: Left : exclusive actions in the GPRO framework. The controller is asked to choose either A or B. Right : the equivalent action modeled with weights in the GPRO framework. The controller is asked to activate or deactivate the free (dashed) edge. Here, ε represents a "small enough" weight.

3. In SSP and GPRO, exclusivity constraints concern edges that leave the same node. Csáji et al. have shown in [CJB09] that if one adds exclusivity constraints between free edges that leave different nodes, PRO becomes NP-hard to solve. This fact is another clue towards the fact that these constraints do make a difference in the efficient solvability of these problems.

4. Solving a GPRO problem can be seen as the search of the best subgraph of a given graph (the support graph) such that some edges cannot be removed, and such that it satisfies additional exclusivity constraints.

5. Because of the context of PRO, we focused here on an SSP-like criterion in which an absorbing state has to be reached as quickly as possible. However, the GPRO formulation could have been adapted to match any MDPs' classical optimization criteria, like the average-cost and the discounted-cost criteria for instance.

2 Comparison between PRO and SSP

In this section, we show that from any instance of SSP, we can build a GPRO instance that has the same optimal solution, and vice versa.

Theorem 1. *Given any instance of an SSP problem with n states and a total of m available actions, it can be reduced in polynomial time to a GPRO problem with $O(m)$ nodes and $O(m)$ free edges that has the same optimal solution. Similarly, given any instance of a GPRO problem with n nodes and f free edges, it can be transformed in polynomial time into an SSP problem with $O(n)$ single-action states and f 2-actions states that has the same optimal solution.*

Proof. We first show how to reduce an SSP to a GPRO and then a GRPO to an SSP.

GOING FROM SSP TO GPRO. Let us create a GPRO instance with a set of nodes $\overline{\mathcal{V}}$ that corresponds to the set of states $\mathcal{S}$ of the SSP. Then, we first claim that any SSP can be expressed as another SSP problem in which there are at most two actions per state, without changing the optimal solution. Then we show how probabilistic 2-actions states can be split into one deterministic 2-actions state and two single-action states. We conclude by showing how single-action states and deterministic 2-actions states can be reduced in the GPRO framework.

Claim 1 : Given any state s of an SSP instance with $k \geq 2$ available actions, s can be split into $(k-1)$ 2-actions states without changing the optimal solution of the original SSP. We show this by induction on k. The base case for $k = 2$ is trivial. Then, if it is true for $k - 1$, it is still true for k. Indeed, let us split s into two states s' and s'' and suppose that s'' has $(k-1)$ actions that correspond to the last $(k-1)$ actions of s while s' has two actions : one that corresponds to the first action of s and one that goes to state s'' deterministically with probability 1 and cost 0. Actions that were previously pointing towards s are now pointing towards s'. Hence state s' corresponds to state s but it has a restricted decision to take : either the first action of s or some of the other actions. The optimal action to take in s does not change in this construction since if the first action of s was optimal, it will also be taken in s' and if not, it means that some of the other actions of s would be preferable so the decision is postponed by choosing the second action of s' that goes to s''. Since s'' has $(k - 1)$ available actions, it can be split into $(k - 2)$ 2-actions states without changing the optimal solution by induction hypothesis, which makes a total of $(k - 1)$ 2-actions states.

Claim 2 : A probabilistic 2-actions state s of an SSP instance can be split into one deterministic 2-actions state u and two (probabilistic) single-action states u' and u''. In u, the choice of one of the two available actions is done, allowing the process to move deterministically to either u' or u'' with probability 1 and cost 0. Now the only available action in u' (resp. u'') performs the randomization relative to the first (resp. second) action of s.

Using Claims 1 and 2, we transform the original SSP problem with n states and a total of m actions into an equivalent SSP problem with $O(m)$ states, all of them being either deterministic 2-actions states or probabilistic single-action states. Indeed, we first create an equivalent SSP with only 2-actions states using Claim 1 and then we transform every probabilistic 2-actions state into one deterministic 2-actions states and two probabilistic single-action states using Claim 2. These transformations can be done in polynomial time and the resulting SSP problem is equivalent to the first SSP in the sense that it still has the same optimal solution. We now give the tools to transform this new SSP problem into a GPRO.

Claim 3 : In an SSP instance, a deterministic action in state s in which one must choose either action A or action B can be reproduced using exclusivity constraint in the GPRO setting. We saw in Section 1 that this is done by giving two free edges to the node $\overline{s}$ that corresponds to state s, and assuming that these edges are linked with an exclusivity constraint. So, 2-actions states in the SSP setting can be modeled using two free edges in the GPRO setting.

Claim 4 : A single-action state s that randomizes between a set of states $\mathcal{S}'$ can be reduced to the GPRO framework by adding an edge from the node $\overline{s}$ corresponding to s to every node $\overline{s}'$ that correspond to the states in $\mathcal{S}'$. To every such edge $(\overline{s}, \overline{s}')$, we give a weight $\mathcal{W}_{\overline{s},\overline{s}'} = \mathcal{P}_{s,s'}$ and a cost $\mathcal{K}_{\overline{s},\overline{s}'} = \mathcal{C}_{s,s'}$. It is not hard to see that the obtained randomization effect in GPRO is

equivalent to that of the SSP (same transition probabilities and same costs). Hence, single-action states can be modeled without any free edge.

We may now reduce the new SSP problem obtained from Claims 1 and 2 into a GPRO, using Claims 3 and 4. Indeed, Claim 3 tells us how to reduce deterministic 2-actions states in a GPRO setting, whereas Claim 4 gives us the argument to reduce probabilistic single-action states, both in polynomial time. The resulting GPRO problem has $O(m)$ nodes (as many as the number of states of the transformed SSP) and $O(m)$ free edges (2 free edges for every 2-actions state). In the resulting GPRO, all the nodes correspond to some state from the transformed SSP and they all have the same probability distribution and transition costs, so the optimal solution of both problems is identical.

GOING FROM GPRO TO SSP. A GPRO is already an instance of SSP with however a small distinction concerning the way the action set is described : in GPRO, actions are taken in the free edges while in SSP, actions are taken in the states. However, we may assume that the actions in GPRO are also taken in the states but then, extra care must be taken. Indeed, if several free edges go out from one single node (say k free edges), then every possible configuration of these free edges (so 2^k configurations) must be considered as an available action in that node and therefore, the number of actions per node can grow exponentially (in the worst case, we may end up with one node that has 2^f available actions, where f is the number of free edges).

Before going further, we must consider the case of a node in which all outgoing edges are free. It that case, we saw in section 1 that we may consider as many actions as there are free edges such that each action corresponds to a situation in which exactly one free edge is activated. Therefore, such nodes with k outgoing free edges may be transformed into a state with k actions.

Now let us consider a node i with $k > 1$ outgoing free edges in addition to some fixed edges. We show that such nodes can be transformed into a substructure in which every node has at most two outgoing free edges. The main idea of the construction, illustrated at figure 2, is to create a new artificial node for every outgoing free edge, which is designed to act exactly as the original free edge. In node i the choice of any edge is taken with respect to their weight but independently from the activation state of the free edges. If a free edge is chosen, the process jumps to the corresponding auxiliary node with cost 0. If the edge was activated, the path that leaves the structure is taken with probability 1 and the cost that corresponds to the original edge while if it is not, the process returns to node i with probability 1 and cost 0. This procedure is then repeated until an activated edge is chosen. Thus, since there is always at least one fixed outgoing edge, we are always able to leave the structure. Observe that the auxiliary nodes exactly match the nodes with exclusive constraints described in the definition of GPRO and they can thus be transformed into states with two deterministic actions in the SSP setting.
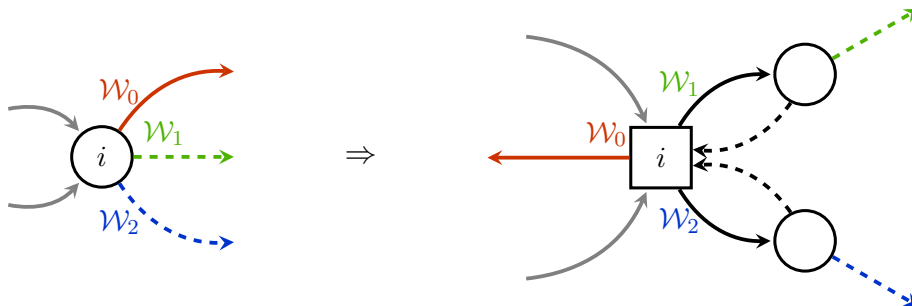

Figure 2: In the right figure, there are maximum two free edges (i.e. dashed edge) per node, even though the two substructures have the same dynamics. All the costs of the black edges are zero, while the costs of the colored edges are the same as the costs of the edges of corresponding color in the left figure.

The whole process needs a polynomial number of transformations (add one node and two edges for some free edges). □

As a final remark for this section, observe that all the arguments we have been using are not specific to the SSP optimization criterion. Here, we focused on SSP because the GPRO formulation originally comes from an SSP-like problem. However, it is easy to generalize GPRO to make it also equivalent to any MDP, whatever the chosen optimization criterion. The same arguments that we used here may be used to make the requested link. As a consequence, an MDP can always be formulated as the search of the best subgraph in a support graph (with some constraints on the edges that are allowed to be removed). This may be useful to enrich the way MDPs are usually viewed and enhance the associated intuition.

3 Applying Policy Iteration to PRO

An adaptation of Policy Iteration (PI) to PRO has been proposed by Csáji et al. in [CJB09]. When writing the algorithm, we represent a configuration of free edges by the set of activated free edges that we denote by policy μ. We also define the first hitting time φ_i^μ of node i under policy μ as the average time needed to reach the target node v_t when starting the process at node i and following policy μ afterwards. Of course, $\varphi_{v_t}^\mu = 0$. First hitting times can be computed in polynomial time by solving a linear system.

We call the resulting adaptation of PI : *PageRank Iteration* (PRI). The different steps are formalized in Algorithm 1.

Algorithm 1 PAGERANK ITERATION

Require: An arbitrary policy μ_0, $k = 0$.
Ensure: The optimal policy μ^*.

1: **while** $\mu_k \neq \mu_{k-1}$ **do**
2:     Evaluation step : compute φ^{μ_k}.
3:     Greedy Improvement step : $\mu_{k+1} = \{(i,j) \in \mathcal{F} : \varphi_i^{\mu_k} \geq \varphi_j^{\mu_k} + 1\}$.
4:     $k \leftarrow k + 1$.
5: **end while**
6: **return** μ_k.

To summarize the operating mode of PRI, we start with an arbitrary policy and then proceed iteratively. At each iteration we determine the set of free edges that are such that, if they were independently switched (i.e. switched *on* if edge is *off* and vice versa), the resulting policy would improve on the preceding one. Then, PRI being a greedy version of PI, we make all the improving switches simultaneously, assuming that this would be even better than single switches. This procedure improves the policy iteratively until no more improvements are possible, meaning that it has converged to the optimal policy. Observe that one does not have to transform the PRO problem in an SSP problem, as the modifications of the policies are handled implicitly directly in the PRO problem.

Since each iteration needs polynomial time to compute, the only condition for PRI to run in polynomial time is to run in a polynomial number of iterations. Unfortunately, determining bounds on the number of iterations of PRI is an open question : the best known upper bound $O(2^f / f)$ is adapted from Mansour and Singh [MS99] whereas Fearnley's exponential lower bound seems unlikely to apply to PRI for the reasons exposed in the previous sections.

We formulate the following conjecture, based on extensive computations.

Conjecture 1. *The number of iterations of PRI is polynomial in the number of free edges.*

To test Conjecture 1, we have first generated random instances of increasing size of PRO and have recorded the number of iterations. Figure 3 (left) shows that the number of iterations

of PRI seems to grow at most linearly with the number of free edges. On the figure, random instances have been generated using a *power-law* distribution [ACL01] but identical simulations have also been performed on *Erdös-Rényi* random graphs [ER60] or on portions of the real web, with about the same tendency each time.

In a second time, we have tried to generate instances that would perform more than f iterations. Therefore, we have generated more than 200 million Erdös-Rényi and Power-law random instances, with parameters ranging from 3 to 10 free edges, 5 to 15 nodes and a highly variable number of edges, without ever being able to find such an example. Figure 3 (right) shows how the number of iterations of PRI are distributed when generating many instances with $n = 8$ and $f = 4$. Note that we have also been exploring bigger value for f and n but since PRI behaves so well in practice, we have only been able to obtain a few iterations w.r.t. the problem size for these instances (always less than 8 iterations). By concentrating on small instances, we were able to generate some examples that were close to cross that f-iterations bound. Even if crossing this bound was possible, our simulations give a good indication about the scarcity of such examples when considering random graphs. Showing that random instances for which PRI takes more than f iterations are unlikely to be observed is in our plans for further research.
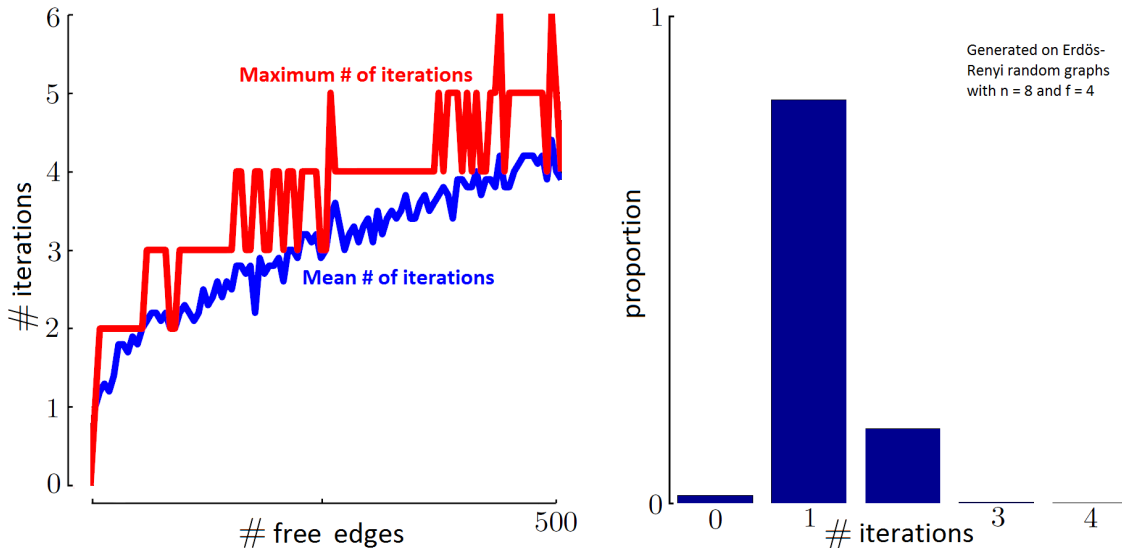

Figure 3: Left : the evolution of the number of iterations when the number of free edges grows. For each value of f, 5 tests have been performed (here on Power Law random graphs) and the average (in blue) and the maximum (in red) number of iterations have been recorded. Right : the distribution of the number of iterations of PRI after over 3 million tests on small Erdös-Renyi random graphs with 8 nodes and 4 free edges each. Among all, 5 tests (so 1.5E-4%) have produced 4 iterations - hitting the barrier of f iterations.

4   Particular cases

In this section, we formulate some particular cases of PRO on which it can be shown that PRI behaves well.

In many applications, it is assumed that the random walk used to compute PageRank can be interrupted at any time with some fixed probability c and start again from an arbitrary node of the graph [Ber05]. This restarting probability is called *zapping*. It can be seen as if the random surfer could get bored of performing its search with probability c and decide to start a new search from a new randomly chosen node. We show below that in such cases, PRI converges in weakly polynomial time.

Theorem 2. *PRO with fixed non-zero zapping probability c can be solved in weakly polynomial time using PRI.*

Proof. Our proof relies on results from Tseng and Puterman [Tse90, Put94]. Puterman shows that PI converges always in less iterations than the other well-known algorithm Value Iteration (VI). Furthermore, Tseng shows that VI converges in at most $O(n \log(n\delta) \eta^{-r})$, where n is the number of states, δ is the binary input size, η is the minimum non-zero transition probability and r is the minimum number of steps needed to join two arbitrary nodes. In case of zapping, there is always a non-zero probability for any node to reach any other node in only one step, i.e. when zapping happens. Hence $r = 1$. Besides, because of the uniform random walk, η is always at least c/n. Regrouping all the arguments, we show that PI must converge in at most $O(n^2 \log(n\delta)/c)$ steps, which is weakly[2] polynomial in n for a fixed value of c. □

In the next case, we show that PRI converges in at most f iterations when all free edges come out of the same arbitrary node w. Note that a particular case of this result was one of the main contribution of [dKND08] : they were able to formulate an explicit optimal strategy when all edges come out of the starting node v_s.

Theorem 3. *PRI takes less than f iterations when all the free edges go out of the same node w and/or out of the starting node v_s.*

Proof. We are going to show that in all considered cases, PRI always makes at least one final decision in each step (final in the sense that it will never be undone in a subsequent iteration). If this is true, then of course PRI takes at most f iterations since we may consider at least one less free edge at each iteration. Furthermore, observe that the nodes may always be sorted w.r.t. their first hitting time at each iteration of PRI : this will be the key to derive our result. The proof goes in three steps : first we suppose that all free edges leave node v_s, then that they all leave some other node w and we finally unify these two results to prove the claim.

- **Case 1 :** *all free edges leave node v_s.* Since no edge enters v_s, switching a free edge that leaves v_s does not influence the first hitting times of the other nodes (it does not shorten or lengthen their path). Hence, their first hitting times is fixed from the beginning and only φ_{v_s} decreases in the iterative process. Suppose that every free edge is initially activated. Then, since φ_{v_s} can only decrease, $\varphi_{v_s} - (\varphi_u + 1)$ can also only decrease for any node u such that $(v_s, u) \in \mathcal{F}$, and therefore free edges can only be deactivated by PRI (see line 3 of the algorithm). So PRI never undoes any of its choices and it converges in at most f iterations. If the initial policy was different, the argument is the same except at the first step where free edges (v_s, u) such that $\varphi_{v_s} \geq (\varphi_u + 1)$ are activated and the other free edges are deactivated. Then again, free edges can only be deactivated since φ_{v_s} is the only one to decrease.

- **Case 2 :** *all free edges leave some node $w \neq v_s$.* Here, the key is to see that when switching a free edge, φ_w decreases more than the other nodes' first hitting times. If this is true, then it means that for any node $u \neq w$, $\varphi_w - (\varphi_u + 1)$ can only decrease and that free edges can only be deactivated at each step, so the argument used in case 1 is still valid. Hence PRI would again take at most f iterations. It only remains to prove that φ_w indeed decrease faster than any other first hitting time, which we do next.

  Let us consider any node $u \neq w$. Among all the paths starting from u that lead to the target node v_t, only those that go through w will be shortened when switching free edges,

---

[2]The bound is only *weakly* polynomial because it depends on the number of bits that are necessary to represent the numbers in a problem instance.

since all free edges leave w. Let us thus partition the set of all paths from u to v_t into the ones that go through w that we denote by P_{uwv}, and the ones that do not go through w that we denote by P_{uv}. We also denote the average weighted length of the paths in P_{uv} by φ_{uv} and the probability to take such a path by p_{uv}, all w.r.t the probability for the considered paths to be chosen. If a path goes through w, it means that u reaches w before reaching v_t (since v_t is absorbing). Therefore, the probability of hitting w before hitting v_t is given by $p_{uw} = 1 - p_{uv}$, and we denote the average weighted length of paths between u and the first visit of w by φ_{uw}. Using these notations, we can write the first hitting time of u as follows :

$$\varphi_u^{\mu_k} = p_{uv}\varphi_{uv} + (1 - p_{uv})(\varphi_{uw} + \varphi_w^{\mu_k}) \tag{1}$$

where $(\varphi_{uw} + \varphi_w^{\mu_k})$ is the average weighted length of a path that goes through node w before reaching v_t. In this equation, observe that only $\varphi_w^{\mu_k}$ can change during the iterative process of PRI since the changes to the probability distributions and to the average lengths of paths can only happen when travelling through w. Let us now suppose that in some step k of PRI, $\varphi_w^{\mu_k}$ decreases from $\Delta\varphi^k$, so $\varphi_w^{\mu_{k+1}} = \varphi_w^{\mu_k} - \Delta\varphi^k$. Using equation (1), the influence of this decrease on $\varphi_u^{\mu_k}$ is thus :

$$\varphi_u^{\mu_{k+1}} = \varphi_u^{\mu_k} - (1 - p_{uv})\Delta\varphi^k.$$

Hence, $\varphi_u^{\mu_k}$ decreases of at most $\Delta\varphi^k$, but only if all its paths to v_t pass through w. Therefore, the first hitting time of w decreases more than the first hitting time of any other node. This concludes the proof for this case.

- **Case 3 :** *all free edges leave either v_s or w.* Since node w is not influenced by node v_s, we can consider the PRI process in w independently from the process in v_s. Thus, in node w, applying case 2, PRI makes at least one switch that is final in each step until every free edge leaving w reaches its optimal activation state. At that point, the first hitting times of every node is fixed for the rest of the process except maybe in node v_s. If φ_{v_s} has not reached its optimal value yet, we let PRI run as if we were in case 1. So, we first focus on w and observe that at least one final switch is made there at each step until the optimal configuration of the free edges leaving w is reached. And then we focus on node v_s where the same observation can be made. Combining both subprocesses, we conclude that one final decision is made at each iteration and so, again, PRI takes at most f iterations to converge. □

References

[ACL01]   W. Aiello, F. Chung, and L. Lu. A random graph model for power law graphs. *Experimental Mathematics*, 10(1):53–66, 2001.

[AL04]   K. Avrachenkov and N. Litvak. Decomposition of the google pagerank and optimal linking strategy. *Technical report, INRIA*, 2004.

[Ber05]   P. Berkhin. A survey on pagerank computing. *Internet Mathematics*, 2(1):73-120, 2005.

[Ber07]   D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, Massachusetts, 3rd edition, 2007.

[BP98]   S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[BT91]   D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580-595, 1991.

[CJB09]   B. C. Csáji, R. M. Jungers, and V. D. Blondel. Pagerank optimization by edge selection. *CoRR*, abs/0911.2280, 2009.

[CLF09]   D. Chaffey, C. Lake, and A. Friedlein. Search engine optimization - best practice guide. *Econsultancy.com Ltd*, 2009.

[dKND08]   C. de Kerchove, L. Ninove, and P. Van Dooren. Maximizing pagerank via outlinks. *Linear Algebra and its Applications*, 429:1254-1276, 2008.

[ER60]   P. Erdős and A. Rényi. *On the evolution of random graphs*. Citeseer, 1960.

[FABG10]   O. Fercoq, M. Akian, M. Bouhtou, and S. Gaubert. Ergodic control and polyhedral approaches to pagerank optimization. *Arxiv preprint arXiv:1011.2348*, 2010.

[Fea10]   J. Fearnley. Exponential lower bounds for policy iteration. *CoRR, abs/1003.3418*, 2010.

[HMZ10]   T. D. Hansen, P. B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. 2010.

[How60]   R.A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[HZ10]   T. Hansen and U. Zwick. Lower bounds for Howard's algorithm for finding minimum mean-cost cycles. *Algorithms and Computation*, pages 415–426, 2010.

[IT09]   H. Ishii and R. Tempo. Computing the pagerank variation for fragile web data. *SICE J. of Control, Measurement, and System Integration*, 2(1):1-9, 2009.

[MS99]   Y. Mansour and S. Singh. On the complexity of policy iteration. *Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[MTZ10]   O. Madani, M. Thorup, and U. Zwick. Discounted deterministic Markov decision processes and discounted all-pairs shortest paths. *ACM Transactions on Algorithms (TALG)*, 6(2):33, 2010.

[MV06]   F. Mathieu and L. Viennot. Local aspects of the global ranking of web pages. In *Proceedings of the 6th International Workshop on Innovative Internet Community Systems*, volume 44. Citeseer, 2006.

[Put94]   M. L. Puterman. *Markov decision processes*. John Wiley & Sons, 1994.

[Tse90]   P. Tseng. Solving h-horizon, stationary Markov decision problems in time proportional to log(h). *Operations Research Letters*, 9(4):287-297, 1990.

[Ye10]   Y. Ye. The simplex method is strongly polynomial for the Markov decision problem with a fixed discount rate. *(Submitted)*, 2010.