PageRank Optimization in Polynomial Time by Stochastic Shortest Path Reformulation

Balázs Csanád Csáji[1,2], Raphaël M. Jungers[3,4], and Vincent D. Blondel[4]

[1] Department of Electrical and Electronic Engineering, School of Engineering,
The University of Melbourne, Australia
bcsaji@unimelb.edu.au
[2] Computer and Automation Research Institute, Hungarian Academy of Sciences
[3] Lab. for Information and Decision Systems, Massachusetts Institute of Technology
[4] Department of Mathematical Engineering,
Université catholique de Louvain, Belgium
vincent.blondel@uclouvain.be, raphael.jungers@uclouvain.be

Abstract. The importance of a node in a directed graph can be measured by its PageRank. The PageRank of a node is used in a number of application contexts – including ranking websites – and can be interpreted as the average portion of time spent at the node by an infinite random walk. We consider the problem of maximizing the PageRank of a node by selecting some of the edges from a set of edges that are under our control. By applying results from Markov decision theory, we show that an optimal solution to this problem can be found in polynomial time. It also indicates that the provided reformulation is well-suited for reinforcement learning algorithms. Finally, we show that, under the slight modification for which we are given mutually exclusive pairs of edges, the problem of PageRank optimization becomes NP-hard.

Keywords: PageRank, graphs, complexity, Markov decision processes.

1 Introduction

The *importance* of a node in a directed graph can be measured by its *PageRank*. The PageRank of a node [4] can be interpreted as the average portion of time spent at the node by an infinite *random walk* [10]. It is traditionally applied for ordering web-search results, but it also has many other applications [2], for example, in bibliometrics, ecosystems, spam detection, web-crawling, semantic networks, relational databases and natural language processing.

It is of natural interest to search for the maximum or minimum PageRank that a node (e.g., a website) can have depending on the presence or absence of some of the edges (e.g., hyperlinks) in the graph. For example, since PageRank is used for ordering web-search results, a web-master could be interested in increasing the PageRank of some of his websites by suitably placing hyperlinks on his own site or by buying advertisements or making alliances with other sites [1, 5]. Another motivation is that of estimating the PageRank of a node in the presence of *missing information* on the graph structure. If some of the links on the internet

M. Hutter et al. (Eds.): ALT 2010, LNAI 6331, pp. 89–103, 2010.

are broken, for example, because the server is down or there are network traffic problems, we may have only partial information on the link structure of the web-graph. However, we may still want to estimate the PageRank of a website by computing the maximum and minimum PageRank that the node may possibly have depending on the presence or absence of the hidden hyperlinks [9]. These hidden edges are often referred to as *fragile links*.

It is known that if we place a new edge in a directed graph, the PageRank of the terminal node of the edge can only increase. Optimal linkage strategies are known for the case in which we want to optimize the PageRank of a node and we only have access to the edges starting from this node [1]. This first result has later been generalized to the case for which we are allowed to configure all of the edges starting from a given set of nodes [5]. The general problem of optimizing the PageRank of a node in the case where we are allowed to decide the absence or presence of the edges in a given arbitrary subset of edges is proposed by Ishii and Tempo [9]. They are motivated by the problem of "fragile links" and mention the lack of efficient, polynomial time algorithms to this problem. Then, using interval matrices, they propose an approximate solution to the problem.

We show that the PageRank optimization problem can be efficiently formulated as a *Markov decision process* (MDP), more precisely, as a *stochastic shortest path* (SSP) problem, and that it can therefore be solved in *polynomial time*. Our proof provides a *linear programming* formulation that can then be solved by standard techniques. Our result and the given reformulation indicate that this problem is well-suited for *reinforcement learning* methods, as well. We also prove that under the slight modification for which we are given mutually exclusive constraints between pairs of edges, the problem becomes *NP-hard*.

2 Definitions and Preliminaries

In this section we define the concept of *PageRank* and the PageRank optimization problem as well as give an introduction to stochastic shortest path problems.

2.1 PageRank

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. First, for simplicity, we assume that $\mathcal{G}$ is *strongly connected*. The adjacency matrix of $\mathcal{G}$ is denoted by A. Since $\mathcal{G}$ is strongly connected, A is *irreducible*. We are going to consider a random walk on the graph defined as follows. If we are in node i, in the next step we will go to node j with probability $1/deg(i)$ if j is an out-neighbor of i, where $deg(\cdot)$ denotes out-degree. This defines a *Markov chain* with transition-matrix

$$P \triangleq \left(D_A^{-1} A\right)^{\mathrm{T}} \qquad \text{with} \qquad D_A \triangleq diag(A\mathbb{1}) \tag{1}$$

where $\mathbb{1} = \langle 1, \ldots, 1 \rangle^{\mathrm{T}}$ is the all-one vector and $diag(\cdot)$ is an operator that creates a diagonal matrix from a vector, more precisely, $(D_A)_{ii} \triangleq (A\mathbb{1})_i = deg(i)$. Note that P is a column (left) stochastic matrix and the chain can be interpreted as an infinite uniform random walk on the graph (e.g., a random surfing).

The PageRank vector, $\boldsymbol{\pi}$, of the graph is defined as the *stationary distribution* of the above described homogeneous Markov chain, more precisely, as $P\,\boldsymbol{\pi} = \boldsymbol{\pi}$, where $\boldsymbol{\pi}$ is non-negative and $\boldsymbol{\pi}^{\mathrm{T}}\mathbb{1} = 1$. Since P is an irreducible stochastic matrix, we know (Perron-Frobenius theorem) that $\boldsymbol{\pi}$ exists and it is unique.

Now, we turn to the general case, when we do not assume that $\mathcal{G}$ is strongly connected, it can be an arbitrary directed graph. In this case, there may be nodes which do not have any outgoing edges. They are usually referred to as *dangling nodes*. There are many ways to handle them [2], e.g., we can delete them, we can add a self-loop to them, each dangling node can be linked to an ideal node (sink) or we can connect each dangling node to every other node. This last solution can be interpreted as restarting the walk from a random starting state if we reach a dangling node. Henceforth, we will assume that we have already dealt with the dangling nodes and, hence, every node has at least one outgoing edge.

We can define a Markov chain similarly to (1), but this chain may not have a unique stationary distribution. The solve this problem, the PageRank vector, $\boldsymbol{\pi}$, of $\mathcal{G}$ is defined as the stationary distribution of the "Google matrix" [10]

$$G \triangleq (1-c)\,P + c\,\boldsymbol{z}\mathbb{1}^{\mathrm{T}}, \tag{2}$$

where $\boldsymbol{z}$ is a positive *personalization vector* satisfying $\boldsymbol{z}^{\mathrm{T}}\mathbb{1} = 1$, and $c \in (0,1)$ is a *damping constant*. In practice, values between 0.1 and 0.15 are usually applied for c and $\boldsymbol{z} = (1/n)\,\mathbb{1}$ [2]. The Markov chain defined by G is *ergodic* that is *irreducible* and *aperiodic*, hence, its stationary distribution uniquely exists and the Markov chain converges to it from any initial distribution [11].

The idea of PageRank is that $\boldsymbol{\pi}(i)$ can be interpreted as the "importance" of i. Thus, $\boldsymbol{\pi}$ defines a *linear order* on the nodes by treating $i \leq j$ if $\boldsymbol{\pi}(i) \leq \boldsymbol{\pi}(j)$.

The PageRank vector can be approximated by the iteration $x_{n+1} \triangleq G x_n$, where x_0 is an arbitrary stochastic vector. It can also be directly computed [1]

$$\boldsymbol{\pi} = c\,(I - (1-c)P)^{-1}\boldsymbol{z}, \tag{3}$$

where I denotes an $n \times n$ identity matrix. Since $c \in (0,1)$ and P is stochastic, it follows that matrix $I - (1-c)P$ is strictly diagonally dominant, thus invertible.

2.2 PageRank Optimization

We will investigate a problem in which a subset of links are "fragile", we do not know their exact values or we have control over them, and we want to compute the maximum (or minimum) PageRank that a specific node can have [9]. More precisely, we are given a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a node $v \in \mathcal{V}$ and a set $\mathcal{F} \subseteq \mathcal{E}$ corresponding to those edges which are under our control. It means that we can choose which edges in $\mathcal{F}$ are present and which are absent, but the edges in $\mathcal{E} \setminus \mathcal{F}$ are fixed, they must be present in the graph. We will call any $\mathcal{F}_+ \subseteq \mathcal{F}$ a *configuration* of fragile links: $\mathcal{F}_+$ determines those edges that we add to the graph, while $\mathcal{F}_- = \mathcal{F} \setminus \mathcal{F}_+$ denotes those edges which we remove. The PageRank of node v under the $\mathcal{F}_+$ configuration is defined as the PageRank of v with

THE MAX-PAGERANK PROBLEM

Instance: A digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a node $v \in \mathcal{V}$ and a set of controllable edges $\mathcal{F} \subseteq \mathcal{E}$.
Optional: A damping constant $c \in (0, 1)$ and a stochastic personalization vector z.
Task: Compute the maximum possible PageRank of v by changing the edges in $\mathcal{F}$ and provide a configuration of edges in $\mathcal{F}$ for which the maximum is taken.

respect to the graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E} \setminus \mathcal{F}_-)$. The problem is how should we configure the fragile links to maximize (or minimize) the PageRank of a given node v?

There are finitely many configurations, thus, we can try to compute them one-by-one. If we have d fragile links, then there are 2^d possible graphs. The PageRank vector of a graph can be computed in $O(n^3)$ via a matrix inversion[1]. The resulting "exhaustive search" algorithm has $O(n^3 2^d)$ time complexity.

We note that if the graph was *undirected*, the Max-PageRank problem would be easy. We know [13] that a random walk on an undirected graph (viz., a time-reversible Markov chain) has the stationary distribution $\boldsymbol{\pi}(i) = deg(i)/2m$ for all nodes i, where m denotes the number of edges and $deg(i)$ is the degree of node i. Therefore, it is easy to see that, in order to maximize the PageRank of a given node v, we should keep edge $(i, j) \in \mathcal{F}$ if and only if $i = v$ or $j = v$.

2.3 Stochastic Shortest Path Problems

In this section we give an overview on stochastic shortest path problems, since our solutions to PageRank optimization are built upon their theory.

Stochastic shortest path (SSP) problems are generalizations of (deterministic) shortest path problems [3]. In an SSP problem the transitions between the nodes are uncertain, but we have some control over their probability distributions. We aim at finding a policy (a function from nodes to controls) such that minimizes the expected cost of reaching a given target state. SSP problems are undiscounted *Markov decision processes* (MDPs) with an absorbing, cost-free terminal state.

An SSP problem can be stated as follows. We have given a finite set of *states*, $\mathbb{S}$, and a finite set of control *actions*, $\mathbb{U}$. For simplicity, we assume that $\mathbb{S} = \{1, \ldots, n, n+1\}$, where $\tau = n + 1$ is a special state, the *target* or *termination* state. In each state i we can choose an action $u \in \mathcal{U}(i)$, where $\mathcal{U}(i) \subseteq \mathbb{U}$ is the set of allowed actions in state i. After the action was chosen, the system moves to state j with probability $p(j \mid i, u)$ and we incur cost $g(i, u, j)$. The cost function is real valued and the transition-probabilities are, of course, nonnegative as well as they sum to one for each state i and action u. The target state is *absorbing* and *cost-free* that is, if we reach state τ, we remain there forever without incurring any more costs. More precisely, for all $u \in \mathcal{U}(\tau)$, $p(\tau \mid \tau, u) = 1$ and $g(\tau, u, \tau) = 0$.

The problem is to find a control *policy* such that it reaches state τ with probability one and minimizes the expected costs, as well. A (stationary, Markov) *deterministic* policy is a function from states to actions, $\mu : \mathbb{S} \to \mathbb{U}$. A *randomized* policy can be formulated as $\mu : \mathbb{S} \to \Delta(\mathbb{U})$, where $\Delta(\mathbb{U})$ denotes the set of all probability distributions over set $\mathbb{U}$. It can be shown that every such policy

[1] It can be done a little faster, in $O(n^{2.376})$, using the Coppersmith-Winograd method.

induces a *Markov chain* on the state space [6]. A policy is called *proper* if, using this policy, the termination state will be reached with probability one, and it is *improper* otherwise. The *value* or *cost-to-go* function of policy μ gives us the expected total costs of starting from a state and following μ thereafter; that is,

$$J^{\mu}(i) \triangleq \lim_{k\to\infty} \mathbb{E}_{\mu}\left[\sum_{t=0}^{k-1} g(i_t, u_t, i_{t+1}) \;\middle|\; i_0 = i\right], \tag{4}$$

for all states i, where i_t and u_t are random variables representing the state and the action taken at time t, respectively. Naturally, i_{t+1} is of distribution $p(\cdot \mid i_t, u_t)$ and u_t is of distribution $\mu(i_t)$; or $u_t = \mu(i_t)$ for deterministic policies.

Applying a proper policy, we arrive at a finite horizon problem, however, the length of the horizon may be random and may depend on the policy, as well.

We say that $\mu_1 \leq \mu_2$ if and only if for all states i, $J^{\mu_1}(i) \leq J^{\mu_2}(i)$. A policy is (uniformly) *optimal* if it is better than or equal to all other policies. There may be many optimal policies, but assuming that (A1) there exists at least one proper policy and (A2) every improper policy yields infinite cost for at least one initial state, they all share the same unique optimal value function, J^*. Then, function J^* is the unique solution of the *Bellman optimality equation*, $TJ^* = J^*$, where T is the *Bellman optimality operator* [3]; defined for all states i as

$$(TJ)(i) \triangleq \min_{u\in\mathcal{U}(i)} \sum_{j=1}^{n+1} p(j \mid i, u)\Big[\, g(i, u, j) + J(j) \,\Big]. \tag{5}$$

Operator T is *monotone* and, assuming that (APP) all allowed policies are proper, T is a *contraction* with respect to a weighted maximum norm [3].

From a given value function J, it is straightforward to get a policy, e.g., by applying a *greedy* policy with respect to J [3]. There are several solution methods for solving MDPs, e.g., in the fields of *reinforcement learning* and [neuro-] *dynamic programming*. Many of these algorithms aim at finding (or approximating) the optimal value function, since good approximations to J^* directly lead to good policies [3]. General solution methods include value iteration, policy iteration, Q-learning, SARSA and TD(λ): temporal difference learning [3, 6, 15].

It is known that finite MDPs are P-complete [14] and SSP problems can be reformulated as *linear programming* (LP) problems [3]. More precisely, the optimal cost-to-go, $J^*(1), \ldots, J^*(n)$, solves the following LP in variables $x_1, \ldots, x_n$:

$$\text{maximize} \qquad \sum_{i=1}^{n} x_i \tag{6a}$$

$$\text{subject to} \qquad x_i \;\leq\; \sum_{j=1}^{n+1} p(j \mid i, u)\Big[\, g(i, u, j) + x_j \,\Big] \tag{6b}$$

for all states i and actions $u \in \mathcal{U}(i)$. Note that x_{n+1} is fixed at zero. This LP has n variables and $O(nm)$ constraints, where m is the maximum number of allowed actions per state. Knowing that an LP can be solved in polynomial time [8], this reformulation provides a *polynomial time* solution to SSP problems.

3 PageRank Optimization as a Markov Decision Process

Before we prove that efficient algorithms to the Max-PageRank problem do exist, first, we recall a basic fact about stationary distributions of Markov chains.

Let $(X_0, X_1, \dots)$ denote a time-homogeneous Markov chain defined on a finite set Ω. The *expected first return time* of a state $i \in \Omega$ is defined as follows

$$\boldsymbol{\varphi}(i) \triangleq \mathbb{E}\left[\inf\{\, t \geq 1 : X_t = i \,\} \mid X_0 = i\right]. \tag{7}$$

If state i is *recurrent*, then $\varphi(i)$ is finite. Moreover, if the chain is irreducible,

$$\boldsymbol{\pi}(i) = \frac{1}{\boldsymbol{\varphi}(i)}, \tag{8}$$

for all states i, where $\boldsymbol{\pi}$ is the stationary distribution of the chain [11]. This naturally generalizes to *unichain* processes, viz., when we have a single *communicating class* of states and possibly some *transient* states. In this case we need the convention that $1/\infty = 0$, since the expected first return time to transient states is ∞. Hence, the stationary distribution of state i can be interpreted as the *average portion of time* spent in i during an infinite random walk. It follows from (8) that maximizing (minimizing) the PageRank of a node is equivalent to minimizing (maximizing) the expected first return time to this node.

We will show that the Max-PageRank problem can be efficiently formulated as a *stochastic shortest path* (SSP) problem [3], where "efficiently" means that the construction (reduction) takes polynomial time. First, we will consider the PageRank optimization *without damping*, namely $c = 0$, but later, we will extend the model to the case of damping and personalization, as well. We will start with a simple, but intuitive reformulation of the problem. Though, this reformulation will not ensure that Max-PageRank can be solved in polynomial time, it is good to demonstrate the main ideas and to motivate the refined solution.

3.1 Assumptions

First, we will make two assumptions, in order to simplify the presentation of the construction, but later, in the main theorem, they will be relaxed.

(AD) *Dangling Nodes Assumption*: We assume that there is a fixed (not fragile) outgoing edge from each node. This assumption guarantees that there are no dangling nodes and there are no nodes with only fragile links.

(AR) *Reachability Assumption*: We also assume that for at least one configuration of fragile links we have a unichain process and node v is recurrent, namely, we can reach node v with positive probability from all nodes. This assumption is required to have a well-defined PageRank for at least one configuration. In our SSP formulation this will be equivalent to assuming that there is at least one *proper* policy. In case of damping this assumption is automatically satisfied, since then the Markov chain is irreducible, and hence unichain, no matter how we configure the fragile links, thus all policies are proper.

3.2 Simple SSP Formulation

First, let us consider an instance of Max-PageRank. We are going to build an associated SSP problem that solves the original PageRank optimization. The *states* of the MDP are the nodes of the graph, except for v which we "split" into two parts and replace by two new states: v_s and v_t. Intuitively, state v_s will be our "starting" state: it has all the outgoing edges of v (both fixed and fragile), but it does not have any incoming edges. The "target" state will be v_t: it has all the incoming edges of node v and, additionally, it has only one outgoing edge: a self-loop. Note that $\tau = v_t$, namely, v_t is the *absorbing termination state*.

An *action* in state i is to select a subset of fragile links (starting from i) which we "turn on" (activate). All other fragile links from i will be "turned off" (deactivated). Thus, for all states i, the allowed set of actions is $\mathcal{U}(i) \triangleq \mathcal{P}(\mathcal{F}_i)$, where $\mathcal{P}$ denotes the power set and $\mathcal{F}_i$ the set of outgoing fragile edges from i.

Let us assume that we are in state i, where there are $a_i \geq 1$ fixed outgoing edges and we have activated $b_i(u) \geq 0$ fragile links, determined by action $u \in \mathcal{U}(i)$. Then, the *transition-probability* to all states j that can be reached from state i using a fixed or an activated fragile link is $p(j \mid i, u) \triangleq 1/(a_i + b_i(u))$.

We define the *immediate-cost* of all control actions as one, except for the actions taken at the cost-free target state. Thus, the immediate-cost function is

$$g(i, u, j) \triangleq \begin{cases} 0 & \text{if } i = v_t, \\ 1 & \text{otherwise,} \end{cases} \tag{9}$$

for all states i, j and actions u. Note that taking an action can be interpreted as performing a step in the random walk. Therefore, the expected cumulative cost of starting from state v_s until we reach the target state v_t is equal to the expected number of steps until we first return to node v according to our original random walk. It follows, that the above defined SSP formalizes the problem of *minimizing* (via a configuration) the expected first return time to state v. Consequently, its solution is equivalent to *maximizing* the PageRank of node v.

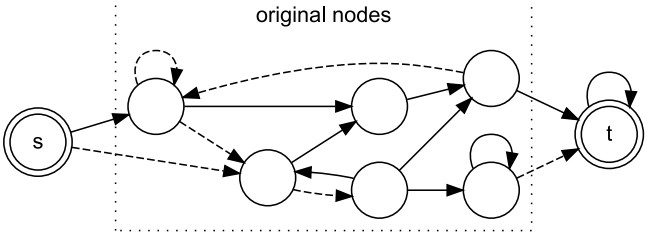

Fig. 1. SSP reformulation: the starting state is $s = v_s$, the target state is $t = v_t$ and the dashed edges denote fragile links. The original nodes in the rectangle exclude v.

Each allowed deterministic *policy* μ defines a potential way to configure the fragile links. Moreover, the v_s component of the cost-to-go function, $J^\mu(v_s)$,

is the expected first return time to v using the fragile link configuration of μ. Therefore, we can compute the PageRank of node v by

$$\boldsymbol{\pi}(v) = \frac{1}{J^{\mu}(v_s)}, \tag{10}$$

where we applied the convention of $1/\infty = 0$, which is needed when v is not recurrent under μ. Thus, the maximal PageRank of v is $1/J^*(v_s)$.

It is known that MDPs can be solved in polynomial time in the number of states, N, and the maximum number of actions per state, M (and the maximum number of bits required to represent the components, L), e.g., by linear programming [12, 14]. The size of the state space of the current formulation is $N = n+1$, where n is the number of vertices of the original graph, but, unfortunately, its action space does not have a polynomial size. For example, if we have maximum m fragile links leaving a node, we had 2^m possible actions to take, namely, we could switch each fragile link independently on or off, consequently, $M = 2^m$. Since $m = O(n)$, from the current reformulation of problem, we have that there is a solution which is polynomial but in 2^n, which is obviously not good enough. However, we can notice that if we restrict the maximum number of fragile links per node to a constant, k, then we could have a solution which is polynomial in n (since the maximum number of actions per state becomes constant: 2^k). This motivates our refined solution, in which we reduce the maximum number of actions per state to two while only slightly increasing the number of states.

3.3 Refined SSP Formulation

We are going to modify our previous SSP formulation. The key idea will be to introduce an *auxiliary state* for each fragile link. Therefore, if we have a fragile link from node i to node j in the original graph, we place an artificial state, f_{ij}, "between" them in the refined reformulation. The refined transition-probabilities are as follows. Let us assume that in node i there were $a_i \geq 1$ fixed outgoing edges and $b_i \geq 0$ fragile links. Now, in the refined formulation, in state i we have only one available action which brings us uniformly, with $1/(a_i + b_i)$ probability, to state j or to state f_{ij} depending respectively on whether there was a fixed or a fragile link between i and j. Notice that this probability is *independent* of how many fragile links are turned on, it is always the same. In each auxiliary state f_{ij} we have two possible actions: we could either turn the fragile link on or off. If our action is "on" (activation), we go with *probability one* to state j, however, if our action is "off" (deactivation), we go back with *probability one* to state i.

We should check whether the transition-probabilities between the original nodes of graph are not affected by this reformulation. Suppose, we are in node i, where there are a fixed and b fragile links[2], and we have turned k of the fragile links on. Then, the transition-probability to each node j, which can be reached via a fixed or an activated fragile link, should be $1/(a + k)$. In our refined reformulation, the immediate transition-probability from state i to state

[2] For simplicity, now we do not denote their dependence on node i.

j is $1/(a+b)$, however, we should not forget about those $b-k$ auxiliary nodes in which the fragile links are deactivated and which lead back to state i with probability one, since, after we returned to state i we have again $1/(a+b)$ probability to go to state j an so on. Now, we will compute the probability of eventually arriving at j if we start in i and only visit auxiliary states meantime.

To simplify the calculations, let us temporarily replace each edge leading to an auxiliary sate corresponding to a *deactivated* fragile link with a self-loop. We can safely do so, since these states lead back to state i with probability one, therefore, the probability of eventually arriving at node j does not change by this modification. Then, the probability of arriving at j can be written as

$$\mathbb{P}\left(\exists t : X_t = j \,|\, \forall s < t : X_s = i\right) = \tag{11a}$$

$$= \sum_{t=1}^{\infty} \mathbb{P}\left(X_t = j \,|\, X_{t-1} = i\right) \prod_{s=1}^{t-1} \mathbb{P}\left(X_s = i \,|\, X_{s-1} = i\right) = \tag{11b}$$

$$= \sum_{t=1}^{\infty} \frac{1}{a+b} \left(\frac{b-k}{a+b}\right)^{t-1} = \frac{1}{a+b} \sum_{t=0}^{\infty} \left(\frac{b-k}{a+b}\right)^{t} = \frac{1}{a+k}\,. \tag{11c}$$

With this, we have proved that the probability of eventually arriving at state j if we start in state i, before arriving at any (non-auxiliary) state l that was reachable via a fixed or a fragile link from state i in the original graph, is the same as the one-step transition-probability was from state i to state j according to the original random walk. This partially justifies the construction.

However, we should be careful, since we might have performed several steps in the auxiliary nodes before we finally arrived at state j. Fortunately, this phenomenon does not ruin our ability to optimize the expected first return time to state v in the original graph, since we count the steps with the help of the cost function, which can be refined according to our needs. All we have to do is to allocate zero cost to those actions which lead us to auxiliary states:

$$g(i,u,j) \triangleq \begin{cases} 0 & \text{if } i = v_t \text{ or } j = f_{il} \text{ or } u = \text{“off”}, \\ 1 & \text{otherwise,} \end{cases} \tag{12}$$

for all states i, j, l and action u. Consequently, we only incur cost if we directly go from state i to state j, without visiting an auxiliary node (viz., it was a fixed link), or if we go to state j via an activated fragile link, since we have $g(f_{ij}, u, j) = 1$ if $u =$ “on”. It is easy to see that in this way we only count the steps of the original random walk and, for example, it does not matter how many times we visit auxiliary nodes, since these visits do not have any cost.

This reformulation also has the nice property that $J^{\mu}(v_s)$ is the expected first return time to node v in the original random walk, in case we have configured the fragile links according to policy μ. The minimum expected first return time that can be achieved with suitably setting the fragile links is $J^*(v_s)$, where J^* is the optimal cost-to-go function of the above constructed SSP problem. Consequently, the *maximum* PageRank node v can have is $1/J^*(v_s)$.

It is also easy to see that if we want to compute the *minimum* possible PageRank of node v, we should simply define a new immediate-cost function as $\hat{g} = -g$, where g is defined by equation (12). If the optimal cost-to-go function of this modified SSP problem is $\hat{J}^*$, then the *minimum* PageRank v can have is $1/|\hat{J}^*(v_s)|$. Thus, Min-PageRank can be handled with the same construction.

The number of states of this formulation is $N = n + d + 1$, where n is the number of nodes of the original graph and d is the number of fragile links. Moreover, the maximum number of allowed actions per state is $M = 2$, therefore, this SSP formulation provides a proof that, assuming (AD) and (AR), Max-PageRank can be solved in *polynomial time* in the size of the problem.

The resulted SSP problem can be reformulated as a linear program, namely, the optimal cost-to-go function solves the following LP in variables x_i and x_{ij},

$$\text{maximize} \qquad \sum_{i \in \mathcal{V}} x_i + \sum_{(i,j) \in \mathcal{F}} x_{ij} \tag{13a}$$

$$\text{subject to} \qquad x_{ij} \leq x_i\,, \qquad \text{and} \qquad x_{ij} \leq x_j + 1\,, \qquad \text{and} \tag{13b}$$

$$x_i \leq \frac{1}{deg(i)} \left[\sum_{(i,j) \in \mathcal{E} \setminus \mathcal{F}} (x_j + 1) + \sum_{(i,j) \in \mathcal{F}} x_{ij} \right], \tag{13c}$$

for all $i \in \mathcal{V} \setminus \{v_t\}$ and $(i,j) \in \mathcal{F}$, where x_i is the cost-to-go of state i, x_{ij} relates to the auxiliary states of the fragile edges, and $deg(\cdot)$ denotes out-degree including both fixed and fragile links (independently of the configuration). Note that we can only apply this LP after state v was "splitted" into a starting and a target state. Also note that the value of the target state, x_{v_t}, is fixed at zero.

3.4 Handling Dangling Nodes

Now, we are going to show that assumption (AD) can be omitted and our complexity result is independent of how dangling nodes are particularly handled.

Suppose that we have chosen a rule according to which the dangling nodes are handled, e.g., we take one of the rules discussed by Berkhin [2]. Then, in case (AD) is not satisfied, we can simply apply this rule to the dangling nodes before the optimization. However, we may still have problems with the nodes which only have fragile links, since they are latent dangling nodes, namely, they become dangling nodes if we deactivate all of their outgoing edges. We call them "fragile nodes". Notice that in each fragile node we can safely restrict the optimization in a way that maximum one of the fragile links can be activated. This does not affect the optimal PageRank of v, since the only one allowed link should point to a node that has the smallest expected hitting time to v. Even if there are several nodes with the same value, we can select one of them arbitrarily.

It may also be the case that deactivating all of the edges is the optimal solution, e.g., if the fragile links lead to nodes that have very large hitting times to v. In this case, we should have an action that has the same effect as the

dangling node handling rule. Consequently, in case we have a fragile node that has m fragile links, we will have $m + 1$ available actions: $u_1, \ldots, u_{m+1}$. If u_j is selected, where $1 \leq j \leq m$, it means that only the j-th fragile link is activated and all other links are deactivated, while if u_{m+1} is selected, it means that all of the fragile links are deactivated and auxiliary links are introduced according to the selected dangling node handling rule. If we treat the fragile nodes this way, we still arrive at an MDP which has states and actions polynomial in n and d, therefore, Max-PageRank can be solved in polynomial time even if (AD) is not satisfied and independently of the applied rule. The modification of the LP formulation if fragile nodes are allowed is straightforward.

3.5 Damping and Personalization

Now, we are going to extend our refined SSP formulation, in order to handle *damping*, as well. For the sake of simplicity, we will assume (AD), but it is easy to remove it in a similar way as it was presented in Section 3.4. Note that assumption (AR) is always satisfied in case of damping (cf. Section 3.1).

Damping can be interpreted as follows: in each step we continue the random walk with probability $1 - c$ and we restart it ("zapping") with probability c, where $c \in (0, 1)$ is a given damping constant. In this latter case, we choose the new starting state of the random walk according to the probability distribution of a given positive and stochastic personalization vector $\boldsymbol{z}$. In order to model this, we introduce a new global auxiliary state, q, which we will call the *teleportation state*, since random restarting is sometimes referred to as "teleportation" [10].

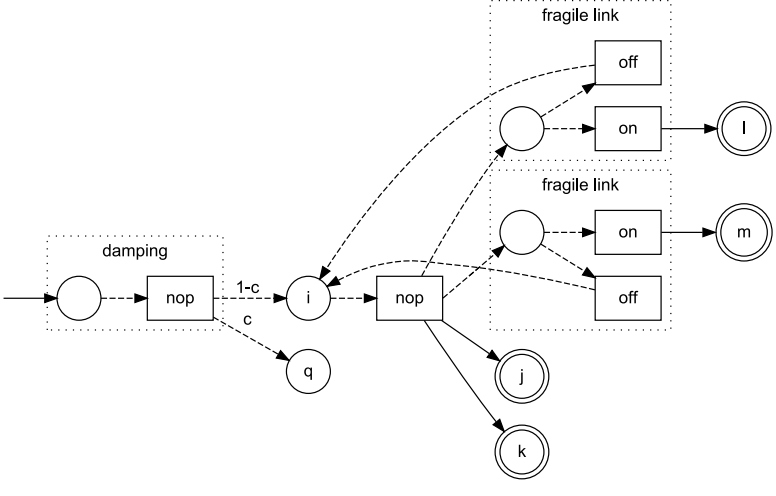

Fig. 2. An illustration of damping: the substructure of a node of the original digraph. Circles represent states and boxes represent actions. State q denotes the global "teleportation" state. Dashed edges help determining zero cost events: if a state-action-state path has only dashed edges, then this triple has zero cost, otherwise, its cost is one.

In order to take the effect of damping into account, we place a new auxiliary state h_i "before" each (non-auxiliary) state i (see Figure 2). Each action that leads to i in the previous formulation now leads to h_i. In h_i we have only one available action ("nop" abbreviating "no operation") which brings us to node i with probability $1-c$ and to q with probability c, except for the target state, v_t, for which h_{v_t} leads with probability one to v_t. In q, we have one available action which brings us with $\boldsymbol{z}$ distribution to the newly defined nodes,

$$p(\,h_i \,|\, q\,) \triangleq p(\,h_i \,|\, q, u\,) \triangleq \begin{cases} \boldsymbol{z}(i) & \text{if } i \neq v_s \text{ and } i \neq v_t \\ \boldsymbol{z}(v) & \text{if } i = v_t \\ 0 & \text{if } i = v_s. \end{cases} \tag{14}$$

All other transition-probabilities from q are zero. Regarding the cost function: it is easy to see that we should not count the steps when we move through h_i, therefore, $g(h_i, u, i) = 0$ and $g(h_i, u, q) = 0$. However, we should count when we move out from the teleportation state, i.e., $g(q, u, i) = 1$ for all i and u.

In this variant the size of the state space is $N = 2n + d + 2$ and we still have maximum 2 actions per state, therefore, it can also be solved in *polynomial time*.

In this case, the LP formulation of finding the optimal cost-to-go is

$$\text{maximize} \qquad \sum_{i\in\mathcal{V}} (x_i + \hat{x}_i) \;+\; \sum_{(i,j)\in\mathcal{F}} x_{ij} \;+\; x_q \tag{15a}$$

$$\text{subject to} \qquad x_{ij} \leq \hat{x}_j + 1\,, \qquad \text{and} \qquad \hat{x}_i \leq (1-c)\,x_i + c\,x_q\,, \tag{15b}$$

$$x_{ij} \leq x_i\,, \qquad \text{and} \qquad x_q \;\leq\; \sum_{i\in\mathcal{V}} \hat{z}_i\,(\hat{x}_i + 1)\,, \tag{15c}$$

$$x_i \;\leq\; \frac{1}{deg(i)} \left[\sum_{(i,j)\in\mathcal{E}\setminus\mathcal{F}} (\hat{x}_j + 1) \;+\; \sum_{(i,j)\in\mathcal{F}} x_{ij} \right], \tag{15d}$$

for all $i \in \mathcal{V} \setminus \{v_t\}$ and $(i,j) \in \mathcal{F}$, where $\hat{z}_i = p(\,h_i \,|\, q)$, $\hat{x}_i$ denotes the cost-to-go of state h_i and x_q is the value of the teleportation state, q. All other notations are the same as in LP (13) and we also have that x_{v_t} and $\hat{x}_{v_t}$ are fixed at zero.

We arrived at an LP problem with $O(n+d)$ variables and $O(n+d)$ constraints. Given an LP with k variables and $O(k)$ constraints, it can be solved in $O(k^3 L)$, where L is the binary input size (for rational coefficients) or in $O(k^3 \log \frac{1}{\varepsilon})$, where ε is the desired precision [8]. Therefore, Max-PageRank can be solved using $O((n+d)^3 L)$ operations under the *Turing model of computation*. Thus:

Theorem 1. *The* Max-PageRank Problem *can be solved in polynomial time even if the damping constant and the personalization vector are part of the input.*

Note that assumptions (AD) and (AR) are not needed for this theorem, since dangling and fragile nodes can be treated as discussed in Section 3.4 (without increasing the complexity) and, in case of damping, all policies are proper.

4 PageRank Optimization with Constraints

In this section we are going to investigate a variant of the PageRank optimization problem in which there are mutually exclusive constraints between the fragile links. More precisely, we will consider the case in which we are given a set of fragile link pairs, $\mathcal{C} \subseteq \mathcal{F} \times \mathcal{F}$, that cannot be activated simultaneously.

The Max-PageRank Problem under Exclusive Constraints	
Instance:	A digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a node $v \in \mathcal{V}$, a set of controllable edges $\mathcal{F} \subseteq \mathcal{E}$ and a set $\mathcal{C} \subseteq \mathcal{F} \times \mathcal{F}$ of those edge-pairs that cannot be activated together. A damping constant $c \in (0, 1)$ and a stochastic personalization vector z.
Task:	Compute the maximum possible PageRank of v by activating edges in $\mathcal{F}$ and provide a configuration of edges in $\mathcal{F}$ for which the maximum is taken.

We will show that the Max-PageRank problem under exclusive constraints is already *NP-hard*, more precisely, that the decision version of it is NP-complete. In the decision version, one is given a number $p \in (0, 1)$ and is asked whether there is a configuration such that the PageRank is larger or equal to p.

Theorem 2. *The decision version of the* Max-PageRank Problem under Exclusive Constraints *is NP-complete.*

Proof. The problem is in NP because given a solution (viz., a configuration), it is easy to *verify* in polynomial time, e.g., via a simple matrix inversion, cf. equation (3), whether the corresponding PageRank is larger or equal than p.

We now reduce the 3SAT problem, whose NP-completeness is well known [7], to this problem. In an instance of the 3SAT problem, we are given a Boolean formula containing m disjunctive *clauses* of three *literals* that can be a variable or its negation, and one is asked whether there is a truth assignment to the variables so that the formula (or equivalently: each clause) is satisfied. Suppose now we are given an instance of 3SAT. We will construct an instance of Max-PageRank under exclusive constraints that solves this particular instance of 3SAT.

We construct a graph having $m + 2$ nodes in the following way: we first put a node s and a node t. Figure it as a source node and a sink node respectively. Each clause in the given 3SAT instance can be written as $y_{j,1} \vee y_{j,2} \vee y_{j,3}$, $1 \leq j \leq m$, where $y_{j,l}$ is a variable or its negation. For each such clause, we add a node v_j between s and t, we put an edge from v_j to itself (a self-loop), we put an edge from s to v_j, and we put three edges between v_j and t, labeled respectively with $y_{j,1}, y_{j,2}$, and $y_{j,3}$. We finally add an edge from t to s. We now define the set of exclusive constrains, $\mathcal{C}$, which concludes the reduction. For all pairs $(y_{j,l}, y_{j',l'})$ such that $y_{j,l} = \bar{y}_{j',l'}$ (i.e., $y_{j,l}$ is a variable and $\bar{y}_{j',l'}$ is its negation, or conversely), we forbid the corresponding pair of edges. Also, for all pairs of edges $(y_{j,l}, y_{j,l'})$ corresponding to a same clause node, we forbid the corresponding pair. This reduction is suitable, since the sizes of the graph and $\mathcal{C}$ are *polynomial* in the size of the 3SAT instance.

We claim that for c small enough, say $c = 1/(100m)$, it is possible to obtain an expected return time from t to itself which is smaller than 77 if and only if the instance of 3SAT is satisfiable. The reason for that is easy to understand with $c = 0$: if the instance is not satisfiable, there is a node v_j with no edge from it to t. In that case, the graph is not strongly connected, and the expected return time from t to itself is infinite. Now, if the instance is satisfiable, let us consider a particular satisfiable assignment. We activate all edges which correspond to a literal which is true and, if necessary, we deactivate some edges so that for all clause nodes, there is exactly one leaving edge to t. This graph, which is clearly satisfiable, is strongly connected, and so the expected return time to t is finite.

Now if $c \neq 0$ is small enough, one can still show by continuity that the expected return time is much larger if some clause node does not have an outgoing edge to t. To see this, let us first suppose that the instance is not satisfiable, and thus that a clause node (say, v_1), has no leaving edge. Then, for all $l \geq 3$, we describe a path of length l from t to itself: this path passes through s, and then remains during $l-2$ steps in v_1, and then jumps to t (with a zapping). This path has probability $(1-c)\frac{1}{m}(1-c)^{l-2}c$. Thus, the expected return time

$$E_1 \;\geq\; \sum_{l=3}^{\infty} lp(l) \;\geq\; \frac{c}{m}\sum_{l=3}^{\infty} l(1-c)^{l-1} \;\geq\; \frac{c}{m}\left[c^{-2}-3\right] \;\geq\; 99, \tag{16}$$

where we assumed that $c = 1/(100m)$ and the personalization vector is $z = (1/n)\,\mathbb{1}$. Note that c and z are part of the input, thus they can be determined.

Consider now a satisfiable instance, and build a corresponding graph so that for all clause nodes, there is exactly one leaving edge. It appears that the expected return time from t to itself satisfies $E_2 \leq 77$. To see this, one can aggregate all the clause nodes in one macro-node, and then define a Markov chain on three nodes that allows to derive a bound on the expected return time from v_t to itself. This bound does not depend on m because one can approximate the probabilities $m/(m+2)$ and $1/(m+2)$ that occur in the auxiliary Markov chain by one so that the bound remains true. Then, by bounding c with $1/8 > 1/(100m)$, one gets an upper bound on the expected return time. For the sake of conciseness, we skip the details of the calculations. To conclude the proof, it is possible to find an edge assignment in the graph so that the PageRank is greater than $p = 1/77$ if and only if the instance is satisfiable. □

5 Conclusions

The task of ordering the nodes of a directed graph according to their *importance* arises in many applications. A promising and popular way to define such an ordering is to use the *PageRank* method [4]. The problem of optimizing the PageRank of a given node by changing some of the edges caused a lot of recent interest [1, 5, 9]. We considered the general problem of finding the extremal values of the PageRank a node can have in the case we are allowed to control (activate or deactivate) some of the edges, which we referred to as *fragile links*.

Our main contribution was that we proved that these problems could be efficiently formulated as stochastic shortest path problems (special Markov decision

processes). This does not only imply that they can be solved in *polynomial time*, but also show that they are well-suited for *reinforcement learning* methods.

We note that we do not need to assume that the graph is simple, namely, it can have multiple edges (and self-loops). This allows the generalization of our results to *weighted graphs*, in case the weights are positive integers or rationals.

We also showed that slight modifications of the problem, as for instance adding mutual exclusive constraints between the activation of several fragile links, may turn the problem *NP-hard*. We conjecture that several other modified variants of the problem are also NP-hard, e.g., the Max-PageRank problem with restrictions on the number of fragile links that can be simultaneously activated.

Acknowledgments

The authors are grateful for the valuable discussions to Paul Van Dooren, Yurii Nesterov, Cristobald de Kerchove, Vincent Traag and Tzvetan Ivanov.

References

[1] Avrachenkov, K., Litvak, N.: The effect of new links on Google PageRank. Stochastic Models 22, 319–331 (2006)
[2] Berkhin, P.: A survey on PageRank computing. Internet Math. pp. 73–120 (2005)
[3] Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific, Belmont (1996)
[4] Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Proc. of the 7th Intern. Conf. on World Wide Web, pp. 107–117 (1998)
[5] De Kerchove, C., Ninove, L., Van Dooren, P.: Maximizing PageRank via outlinks. Linear Algebra and its Applications 429, 1254–1276 (2008)
[6] Feinberg, E.A., Shwartz, A. (eds.): Handbook of Markov Decision Processes: Methods and Applications. Kluwer Academic Publishers, Dordrecht (2002)
[7] Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
[8] Gonzaga, C.C.: An Algorithm for Solving Linear Programming Problems in $O(n^3L)$ operations. In: Progress in Mathematical Programming: Interior-Point and Related Methods, pp. 1–28. Springer, Heidelberg (1988)
[9] Ishii, H., Tempo, R.: Computing the PageRank variation for fragile web data. SICE J. of Control, Measurement, and System Integration 2(1), 1–9 (2009)
[10] Langville, A.N., Meyer, C.D.: Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press, Princeton (2006)
[11] Levin, D.A., Peres, Y., Wilmer, E.L.: Markov Chains and Mixing Times. American Mathematical Society, Providence (2009)
[12] Littman, M.L., Dean, T.L., Kaelbling, L.P.: On the complexity of solving Markov decision problems. In: Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence, pp. 394–402 (1995)
[13] Lovász, L.: Random walks on graphs: A survey. In: Combinatorics: Paul Erdős is Eighty, vol. 2, pp. 348–353. Bolyai Society Mathematical Studies, Budapest (1996)
[14] Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Mathematics of Operations Research 12(3), 441–450 (1987)
[15] Sutton, R.S., Barto, A.G.: Reinforcement learning. MIT Press, Cambridge (1998)