Discrete Applied Mathematics 169 (2014) 73–87

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

PageRank optimization by edge selection

Balázs Csanád Csáji [a,b,*], Raphaël M. Jungers [c,d], Vincent D. Blondel [d]

[a] Department of Electrical and Electronic Engineering, School of Engineering, The University of Melbourne, Australia
[b] Computer and Automation Research Institute, Hungarian Academy of Sciences, Hungary
[c] Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, United States
[d] Department of Mathematical Engineering, Université catholique de Louvain, Belgium

ARTICLE INFO

Article history:
Received 16 August 2010
Received in revised form 3 December 2012
Accepted 9 January 2014
Available online 2 February 2014

Keywords:
PageRank
Computational complexity
Stochastic shortest path

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. Our core solution results in a linear programming formulation, but we also provide an alternative greedy algorithm, a variant of policy iteration, which runs in polynomial time, as well. 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.

1. Introduction

The *importance* of a node in a directed graph can be measured by its *PageRank*. The PageRank of a node [6] can be interpreted as the average portion of time spent at the node by an infinite *random walk* [16], or in other words, the weight of the node with respect to the *stationary distribution* of an associated homogeneous Markov chain. PageRank 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 [20]. 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 [1,9]. 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 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 unknown, hidden hyperlinks [14]. These hidden edges are often referred to as *fragile links*.

* Corresponding author at: Department of Electrical and Electronic Engineering, School of Engineering, The University of Melbourne, Australia. Tel.: +61 383440498; fax: +61 383446678.
E-mail addresses: bcsaji@unimelb.edu.au, balazs.csaji@sztaki.mta.hu (B.Cs. Csáji), raphael.jungers@uclouvain.be (R.M. Jungers), vincent.blondel@uclouvain.be (V.D. Blondel).

0166-218X/$ – see front matter
http://dx.doi.org/10.1016/j.dam.2014.01.007

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 [9].

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 [14]. 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.

Here, we aim at showing that this general problem can be solved in polynomial time. To the best of our knowledge, an earlier account of our method [8] was the first polynomial-time solution to this problem. Later, it was followed by another polynomial-time approach given by Fercoq et al. [11].

In this paper 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, but we propose a greedy algorithm, as well, which is a variant of the *policy iteration* algorithm. This latter method also runs in polynomial time, under some assumptions. Our main result on polynomial-time computability remains valid even if the *damping constant* and the *personalization vector* are part of the input and it does not depend on the particular way the *dangling nodes* are handled. 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 a brief introduction to stochastic shortest path problems, a special class of Markov decision processes (MDPs).

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 define a *random walk* on the graph. 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

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

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 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 Markov chain, more precisely, as $P\,\boldsymbol{\pi} = \boldsymbol{\pi}$, where $\boldsymbol{\pi} \geq 0$ and $\boldsymbol{\pi}^{\mathrm{T}}\mathbb{1} = 1$. Since P is an irreducible stochastic matrix, we know, e.g., from the Perron–Frobenius theorem, that $\boldsymbol{\pi}$ exists and 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], for example, we can delete them, we can add a self-loop to them, each dangling node can be linked to an artificial node (sink) or we can connect each dangling node to every other node. This last solution can be interpreted as restarting the random 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, therefore, every node has at least one outgoing edge.

We can then define a Markov chain similarly to (1), but this chain may not have a unique stationary distribution. To solve this problem, the PageRank vector is defined as the stationary distribution of the "Google matrix" [16]

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

where $\boldsymbol{z} > 0$ is a *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 *irreducible* and *aperiodic*, consequently, its stationary distribution uniquely exists and the Markov chain converges to it from any initial distribution [17].

An application of PageRank is that $\boldsymbol{\pi}(i)$ can be interpreted as the "importance" of node i. Therefore, we can use $\boldsymbol{\pi}$ to define a *total pre-order* on the nodes of the graph by treating $i \lesssim j$ if and only 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, or it can be directly computed [1]

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

where I denotes the $n \times n$ identity matrix. Since $c \in (0, 1)$ and P is stochastic, 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", i.e., we do not know whether they are present in the graph or we have control over them, and we want to compute the maximum (or minimum) PageRank that a specific node can have [14]. 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 exist 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 v under the $\mathcal{F}_+$ configuration is defined as the PageRank of v w.r.t. the graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E} \setminus \mathcal{F}_-)$. The problem is the following: how should we configure the fragile links to maximize (or minimize) the PageRank of a given node v?

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 $\boldsymbol{z}$.
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.

The Min-PageRank problem, which can be used, e.g., to obtain a sharp lower bound on the PageRank of a node in case the link structure is only partially known, can be stated similarly. We will concentrate on Max-PageRank, but a straightforward modification of our method can deal with the Min problem, as well. We will show that Max-PageRank can be solved in polynomial time, under the Turing model of computation, even if the damping constant and the personalization vector are part of the input, i.e., not fixed.

Of course, in a particular instance of the Max-PageRank problem, there are finitely many configurations, thus, we can try to compute them one-by-one. If we have d fragile links, 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.

Note that if the graph was *undirected*, the Max-PageRank problem would be easy. We know [19] that a random walk on an undirected graph, 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. Hence, 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 [5]. 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 control policy (a function from nodes to controls) that minimizes the expected (cumulative) cost of reaching a given target state. SSP problems are finite, undiscounted *Markov decision processes* (MDPs) with an absorbing, cost-free termination 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 induces a *Markov chain* on the state space [10]. 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

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

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 $u_t = \mu(i_t)$ in case we apply a deterministic policy. Note that by 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 applied control 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 operator* [5], that is,

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

for all states $i \in \mathbb{S}$ and value functions $J : \mathbb{S} \to \mathbb{R}$. The Bellman operator of a (randomized) policy μ is defined for all state i as

$$(T_{\mu}J)(i) \triangleq \sum_{u\in\mathcal{U}(i)} \mu(i, u) \sum_{j=1}^{n+1} p(j \mid i, u)[g(i, u, j) + J(j)], \tag{6}$$

where $\mu(i, u)$ is the probability that policy μ chooses action u in state i.

Given the assumptions above, *value iteration* converges in SSPs [4],

$$\lim_{k\to\infty} T_{\mu}^{k} J = J^{\mu}, \qquad \lim_{k\to\infty} T^{k} J = J^{*}. \tag{7}$$

Operators T and T_{μ} are *monotone* and, assuming that (APP) all policies are proper, T and T_{μ} are *contractions* w.r.t. a weighted maximum norm [5].

From a given value function J, it is straightforward to get a policy, e.g., by applying a *greedy* policy [5] with respect to J that is, for all state i,

$$\mu(i) \in \arg\min_{u\in\mathcal{U}(i)} \sum_{j=1}^{n+1} p(j \mid i, u)[g(i, u, j) + J(j)]. \tag{8}$$

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 [5]. General solution methods include value iteration, policy iteration, Gauss–Seidel method, Q-learning, SARSA and TD(λ): temporal difference learning [5,10,23].

Later, we will apply a variant of the *policy iteration* (PI) algorithm. The basic version of PI works as follows. We start with an arbitrary *proper* policy, μ_0. In iteration k we first *evaluate* the actual policy, μ_k, by solving the linear system, $T_{\mu_k} J^{\mu_k} = J^{\mu_k}$, and then we *improve* the policy by defining μ_{k+1} as the greedy policy w.r.t. J^{μ_k}. The algorithm terminates if $J^{\mu_k} = J^{\mu_{k+1}}$. Assuming (A1) and (A2), PI generates an improving sequence of proper policies and it always finds an optimal solution in a finite number of iterations [5].

It is known that all of the three classical variants of MDPs (finite horizon, infinite horizon discounted cost and infinite horizon average cost) can be solved in polynomial time [18]. Moreover, these classes of problems are P-complete [21]. In the case of SSP problems, they can be reformulated as *linear programming* (LP) problems [5], more precisely, the optimal cost-to-go, $J^*(1), \ldots, J^*(n)$, solves the following LP in variables $x_1, \ldots, x_n$:

$$\text{maximize} \qquad \sum_{i=1}^{n} x_i \tag{9a}$$

$$\text{subject to} \qquad x_i \leq \sum_{j=1}^{n+1} p(j \mid i, u)[g(i, u, j) + x_j] \tag{9b}$$

for all states i and actions $u \in \mathcal{U}(i)$. Note that the value of the termination state, 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 [13] (in the number of variables, the number of constraints and the binary size of the input), this reformulation already provides a way to solve an SSP problem in *polynomial time*.

Assuming that all policies are proper (APP), the state space can be *partitioned* into nonempty subsets $S_1, \ldots, S_r$ such that for any $1 \leq q \leq r$, state $i \in S_q$ and action $u \in \mathcal{U}(i)$, there exists some $j \in \{\tau\} \cup S_1 \cup \cdots \cup S_{q-1}$ such that $p(j \mid i, u) > 0$. Then, if assumption (APP) holds, value iteration can find an optimal policy after a number of iterations that is bounded by a polynomial in L (the binary input size) and η^{-2r}, where η is the smallest positive transition probability [24]. Since policy iteration converges no more slowly than value iteration [22], policy iteration also terminates in iterations bounded by a polynomial in L and η^{-2r}, assuming (APP).

3. PageRank optimization as a Markov decision process

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

Let $(X_0, X_1, \ldots)$ 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

$$\boldsymbol{\varphi}(i) \triangleq \mathbb{E}\left[\inf\{t \geq 1 : X_t = i\} \mid X_0 = i\right]. \tag{10}$$

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

$$\boldsymbol{\pi}(i) = \frac{1}{\boldsymbol{\varphi}(i)}, \tag{11}$$

for all states i, where $\boldsymbol{\pi}$ is the stationary distribution of the Markov chain [17]. 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 Eq. (11) that the problem of maximizing [minimizing] the PageRank of a node is equivalent to the problem of 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 [5], 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 of the graph. This assumption guarantees that there are no dangling nodes as well as there are no nodes with only fragile links (which would be latent dangling nodes).

(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 of the graph. This assumption is required to have a well-defined PageRank for at least one configuration. In our SSP formulation this assumption will be equivalent to assuming that there is at least one *proper* policy. In case of damping, this assumption is automatically satisfied, as the Markov chain will be irreducible, and accordingly, unichain. On the other hand, irrespective of how we configure fragile links, all policies in the corresponding SSP problem 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 problem. 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* of the associated SSP problem (see Fig. 1).

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, in state i the allowed set of actions is $\mathcal{U}(i) \triangleq \mathcal{P}(\mathcal{F}_i)$, where $\mathcal{P}$ is the power set and $\mathcal{F}_i$ is the set of outgoing fragile links 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 $p(j \mid i, u) \triangleq 1/(a_i + b_i(u))$.

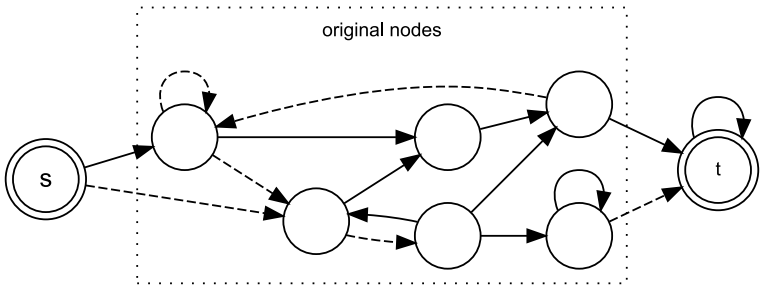

Fig. 1. SSP reformulation: the starting state is $s = v_s$, the target state is $t = v_t$ and the dashed edges denote fragile links. The original nodes in the rectangle exclude v.

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

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

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* the expected first return time to state v. Hence, its solution is equivalent to *maximizing* the PageRank of node 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

$$\boldsymbol{\pi}(v) = \frac{1}{J^\mu(v_s)}, \tag{13}$$

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 $1/J^*(v_s)$.

Most solution algorithms compute the optimal cost-to-go function, J^*, but even if we use a direct policy search method, it is still easy to get back the value function of the policy. We can compute, for example, the expected first return time if we configure the fragile links according to policy μ as follows. For simplicity, assume that $v_s = 1$, then

$$J^\mu(1) = \mathbb{1}^{\mathrm{T}}(I - P_\mu)^{-1}e_1, \tag{14}$$

where e_j is jth canonical basis vector, I is an $n \times n$ identity matrix and P_μ is the *substochastic* transition matrix of the corresponding SSP problem *without the row and column of the target state*, v_t, if we configure the fragile links according to policy μ. Regarding the invertibility of $I - P_\mu$ note that

$$(I - P_\mu)^{-1} = \sum_{n=0}^{\infty} P_\mu^n, \tag{15}$$

and we know that this *Neumann series* converges if $\varrho(P_\mu) < 1$, where $\varrho(\cdot)$ denotes *spectral radius*. Thus, $(I - P_\mu)^{-1}$ is well-defined for all *proper* policies, since it is easy to see that policy μ is proper if and only if $\varrho(P_\mu) < 1$.

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 [18,21]. 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 have 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

Now, we present a refined SSP formulation which will be the base of the proof that shows the polynomial-time computability of Max-PageRank.

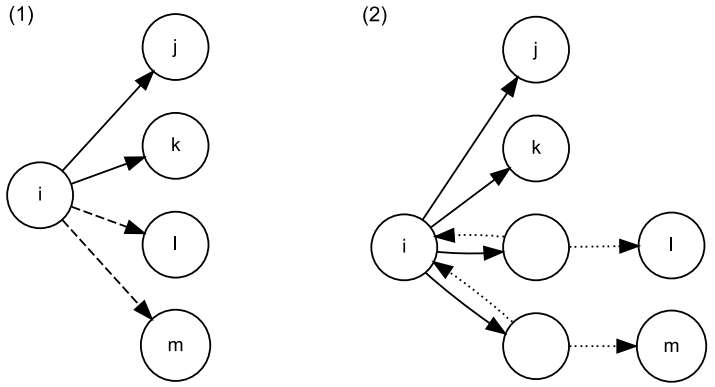

Fig. 2. An example for inserting auxiliary states for fragile links. The left hand side presents the original situation, in which dashed edges are fragile links. The right hand side shows the refined reformulation, where the dotted edges represent possible actions.

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 return 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 (it is illustrated by Fig. 2). 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 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 state 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. After this modification, the probability of arriving at state j if one starts in state i can be written as

$$\mathbb{P}\left(\exists\, t : X_t = j \mid \forall\, s < t : X_s = i\right) \tag{16a}$$

$$= \sum_{t=1}^{\infty} \mathbb{P}\left(X_t = j \mid X_{t-1} = i\right) \prod_{s=1}^{t-1} \mathbb{P}\left(X_s = i \mid X_{s-1} = i\right) \tag{16b}$$

$$= \sum_{t=1}^{\infty} \frac{1}{a+b} \left(\frac{b-k}{a+b}\right)^{t-1} = \frac{1}{a+b} \sum_{t=0}^{\infty} \left(\frac{b-k}{a+b}\right)^{t} = \frac{1}{a+k}\,. \tag{16c}$$

With this, we 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 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. More precisely, the immediate-cost function should be

$$g(i,u,j) \triangleq \begin{cases} 0 & \text{if } i = v_t \text{ or } j = f_{il} \text{ or } u = \text{"off"}, \\ 1 & \text{otherwise}, \end{cases} \tag{17}$$

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

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 (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, e.g., 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. Thus, 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 v, we should simply define a new immediate-cost function as $\hat{g} = -g$, where g is defined by Eq. (17). If the optimal cost-to-go function of this modified SSP problem is $\hat{J}^*$, 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*. 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},

$$\text{maximize} \quad \sum_{i \in \mathcal{V}} x_i + \sum_{(i,j) \in \mathcal{F}} x_{ij} \tag{18a}$$

$$\text{subject to} \quad x_{ij} \leq x_i\,, \quad \text{and} \quad x_{ij} \leq x_j + 1, \quad \text{and} \tag{18b}$$

$$x_i \leq \frac{1}{deg(i)} \left[\sum_{(i,j) \in \mathcal{E} \setminus \mathcal{F}} (x_j + 1) + \sum_{(i,j) \in \mathcal{F}} x_{ij} \right], \tag{18c}$$

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 "split" into a starting and a target state and the value of the target state, x_{v_t}, is fixed at zero, since it is the termination state of the constructed SSP problem.

3.4. Handling dangling nodes

Now, we will 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 link allowed 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. Naturally, this restriction of the optimization to only one allowed activated fragile link per state is only suitable for fragile nodes, it is not applicable in general, when the node has fixed edges, as well.

It may also be the case that deactivating all of the edges is the optimal solution, for example, 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 jth 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, the PageRank optimization problem can be solved in polynomial time even if assumption (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 will 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 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

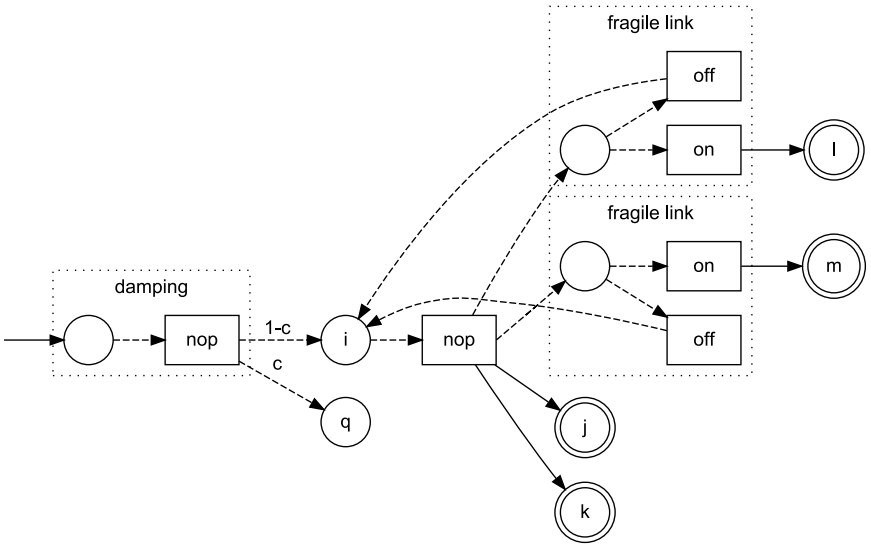

Fig. 3. An illustration of damping: the substructure of a node of the original digraph. Circles represent states and boxes represent actions. State q denotes the global "teleportation" state. Dashed edges help determining zero cost events: if a state-action-state path has only dashed edges, then this triple has zero cost, otherwise, its cost is one.

state 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" [16].

In order to take the effect of damping into account in each step, we place a new auxiliary state h_i "before" each (non-auxiliary) state i (see Fig. 3). 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 the teleportation state q with probability c, except for the target state, v_t, for which h_{v_t} leads with probability one to v_t. In state q, we have only one available action which brings us with distribution $\boldsymbol{z}$ to the newly defined nodes, that is we have

$$p(h_i \mid q) \triangleq p(h_i \mid q, u) \triangleq \begin{cases} \boldsymbol{z}(i) & \text{if } i \neq v_s \text{ and } i \neq v_t \\ \boldsymbol{z}(v) & \text{if } i = v_t \\ 0 & \text{if } i = v_s. \end{cases} \tag{19}$$

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 state q, i.e., $g(q, u, i) = 1$ for all i and u.

The straightforward solution, namely, to connect i directly to q without an additional auxiliary state, h_i, does not work, since the transition-probability to q should be constant (i.e., equal to c), but the probabilities of taking a link starting from i change as we change the configuration of fragile links.

In this variant, in which we take damping and personalization into account, 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

$$\text{maximize} \quad \sum_{i \in \mathcal{V}} (x_i + \hat{x}_i) + \sum_{(i,j) \in \mathcal{F}} x_{ij} + x_q \tag{20a}$$

$$\text{subject to} \quad x_{ij} \leq \hat{x}_j + 1, \quad \text{and} \quad \hat{x}_i \leq (1-c)\, x_i + c\, x_q\,, \tag{20b}$$

$$x_{ij} \leq x_i, \quad \text{and} \quad x_q \leq \sum_{i \in \mathcal{V}} \hat{z}_i\, (\hat{x}_i + 1), \tag{20c}$$

$$x_i \leq \frac{1}{deg(i)} \left[\sum_{(i,j) \in \mathcal{E} \setminus \mathcal{F}} (\hat{x}_j + 1) + \sum_{(i,j) \in \mathcal{F}} x_{ij} \right], \tag{20d}$$

for all $i \in \mathcal{V} \setminus \{v_t\}$ and $(i, j) \in \mathcal{F}$, where $\hat{z}_i = p(h_i \mid q)$, $\hat{x}_i$ denotes the cost-to-go of h_i and x_q is the value of the teleportation state. All other notations are the same as in (18) and we also have that x_{v_t} and $\hat{x}_{v_t}$ are fixed at zero.

The above LP problem has $O(n + d)$ variables and $O(n + d)$ constraints, which can thus be solved in $O((n + d)^3 L)$, where L is the binary input size (for rational coefficients) or in $O((n + d)^3 \log \frac{1}{\varepsilon})$, where ε is the desired precision [13]. The result that was proved through Sections 3.3–3.5 is

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

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.

Assuming that c and $\boldsymbol{z}$ can be represented using a number of bits polynomial in n, which is the case in practice, since c is usually 0.1 or 0.15 and $\boldsymbol{z} = (1/n)\,\mathbb{1}$ [2], we arrive at a *strongly* polynomial-time solution, because all other coefficients can be represented using $O(\log n)$ bits.

3.6. State space reduction

In practical situations the state space may be very large which can make direct solutions impractical. Approximate, sampling based methods are usually preferred in these circumstances [5,10,23]. In this section, we show that the state space in the presented SSP formulation can often be reduced.

In the last SSP formulation in $2n + 1$ states there is no real choice (there is only one available action) which allows the reduction of the state space. In this complementary section we are going to show that given an SSP problem with $N = r + s$ states, in which in r states there is only one available action, we can "safely" reduce the number of states to s. More precisely, we will prove that we can construct another SSP problem with only s states which is "compatible" with the original one in the sense that there is a one-to-one correspondence between the policies of the reduced and the original problems, and the value functions of the policies (restricted to the remaining s states) are the same in both problems. Hence, finding an optimal policy for the reduced problem is equivalent to solving the original one. The computational complexity of the construction is $O(r^3 + r^2sm + s^2rm)$, where m denotes the maximum number of allowed actions per state. We will often omit L, the binary size of the input or the desired precision of the computations.

3.6.1. Assumptions

We will apply immediate-cost functions of the form $g : \mathbb{S} \times \mathbb{U} \to \mathbb{R}$. If we have a cost function that also depends on the arrival state (like in the refined variant above), we can redefine it using the expected cost per stage,

$$g(i, u) = \sum_{j=1}^{N} p(j \mid i, u)\, \tilde{g}(i, u, j), \tag{21}$$

which would not affect the outcome of the optimization [3]. The new cost function can be computed using $O(N^2m) = O(r^2m + rsm + s^2m)$ operations.

We will call the states in which there is only one available action as "non-decision" states, while the other states will be called "decision" states. By convention, we classify the target state, τ, as a decision state. We assume, without loss of generality, that the (indices of the) non-decision states are $1, \ldots, r$ and the decision states are $r + 1, \ldots, r + s$. Finally, we also assume that there exists at least one proper control policy.

3.6.2. Constructing the reduced SSP problem

Notice that the transition-matrix of the Markov chain induced by (any) control policy μ of the original SSP problem looks like

$$P_\mu = \begin{bmatrix} R_0 & R_\mu \\ Q_0 & Q_\mu \end{bmatrix}, \tag{22}$$

where $R_0 \in \mathbb{R}^{r\times r}$ describes the transition-probabilities between the non-decision states; $Q_0 \in \mathbb{R}^{s\times r}$ contains the transitions from the non-decision states to the decision states; $Q_\mu \in \mathbb{R}^{s\times s}$ describes the transitions between the decision states and, finally, $R_\mu \in \mathbb{R}^{r\times s}$ contains the transitions from the decision states to the non-decision states. Note that R_0 and Q_0 do not depend on the policy, since they correspond to non-decision states.

In the reduced problem we will only keep the s decision states and remove the r non-decision states. We will redefine the transition-probabilities between the decision states as if we would "simulate" the progress of the system through the non-decision states until we finally arrive at a decision state. In order to calculate the probabilities of arriving at specific decision states if we started in specific non-decision states, we can define a new Markov chain

$$P_0 = \begin{bmatrix} R_0 & 0 \\ Q_0 & I \end{bmatrix}, \tag{23}$$

where 0 is a $r \times s$ zero matrix and I is an $s \times s$ identity matrix. We can interpret this matrix as if we would replace each decision state by an absorbing state. We assumed that there is at least one proper policy and we know that R_0 and Q_0 are the same for all policies as well as the target state is a decision state, therefore, R_0^k converges to the zero matrix as $k \to \infty$,

thus

$$\lim_{k\to\infty} P_0^k = \begin{bmatrix} 0 & 0 \\ Q^* & I \end{bmatrix}, \tag{24}$$

where Q^* contains the arrival distributions to the decision states if we started in one of the non-decision states. More precisely, Q^*_{ij} is the probability of arriving at (decision) state i if we start at (non-decision) state j. It is known [15] that these probabilities can be calculated using the *fundamental matrix* of the Markov chain, $F = (I - R_0)^{-1}$. More precisely,

$$Q^* = Q_0 F = Q_0 (I - R_0)^{-1}, \tag{25}$$

and the computation requires a matrix inversion and a matrix multiplication. If we use classical methods, Q^* can be calculated in $O(r^3 + r^2 s)$ (the method of Coppersmith and Winograd [7] could also be applied). Using Q^* the transition matrix of μ in the reduced problem should be

$$\widehat{P}_\mu = Q_\mu + Q^* R_\mu. \tag{26}$$

This matrix encodes the idea that if we arrive at a non-decision state, we simulate the progress of the system until we arrive at a decision state. Fortunately, we do not have to compute it for all possible policies, we only need to define the transition-probabilities accordingly:

$$\widehat{p}(j \mid i, u) \triangleq p(j \mid i, u) + \sum_{k=1}^{r} p(k \mid i, u) Q^*_{jk} \tag{27}$$

for all states $i, j > r$ and action $u \in \mathcal{U}(i)$. Note that states i and j are decision states (their indices are larger than r). Thus, computing the new transition-probability function can be accomplished using $O(s^2 rm)$ operations.

We should also modify the immediate-cost function, in order to include the expected costs of those stages that we spend in the non-decision states, as well. It is known that the fundamental matrix contains information about the expected absorbing times. More precisely, F_{jk} is the expected time spent in (non-decision) state j before arriving at a (decision) state (absorption), if the process started in (non-decision) state k [15]. Therefore,

$$\widehat{g}(i, u) \triangleq g(i, u) + \sum_{k=1}^{r} p(k \mid i, u) \sum_{j=1}^{r} F_{jk}\, g(j), \tag{28}$$

for all $i > r$ and $u \in \mathcal{U}(i)$, where we did not denote the dependence of the cost function on the actions for non-decision states, since there is only one available action in each such state. Thus, $g(j) \triangleq g(j, u)$, where u denotes the only available action in state j. Computing the new cost-function needs $O(r^2 sm)$ operations, if we already have the fundamental matrix.

We have only removed non-decision states, in which there is only one allowed action, consequently, it is trivial to extend a policy of the reduced problem to a policy of the original one, and there is a *bijection* between such policies. Since we defined the transition-probabilities and the immediate-cost function in a way that it mimics the behavior of the original problem, solving the reduced problem is equivalent to solving the original one. Summing all computations together, we can conclude that the time complexity of the construction is $O((r^3 + r^2 sm + s^2 rm)L)$, where L is the binary size or precision.

3.6.3. Reducing the SSP formulation of Max-PageRank

Applying this result to the refined SSP formulation of Max-PageRank, we can reduce the number of states to d (without τ) by constructing another SSP problem as demonstrated above. It can be summarized as

Lemma 2. *The* Max-PageRank *problem with a digraph having n nodes and d fragile links can be reduced to an SSP problem with only d states (plus the termination state) by using $O((n^3 + d^2 n + n^2 d)L)$ operations.*

4. PageRank iteration

In the previous sections we saw how to reformulate efficiently the Max-PageRank problem as an SSP problem. This SSP formulation could then be further reformulated as an LP problem, which type of problems are known to be solvable in polynomial time, for example, by interior point methods.

Now, we will provide an alternative solution to the Max-PageRank problem. We will build on the previous SSP formulation, but instead of using an LP-based solution, we will define a simple iterative algorithm that in each step updates the configuration of the fragile links in a greedy way. Yet, as we will see, this method is efficient in many sense. For simplicity, we will only consider the case without damping ($c = 0$) and we will apply the assumption:

(AB) *Bounded Reachability Assumption*: We assume that the target node, v, can be reached from all nodes of the graph via a bounded length path of fixed edges. In other words, there is a universal constant κ such that node v can be reached from all nodes by taking at most κ fixed edges. The fact that κ is universal means that it does not depend on the particular problem instance.

The algorithm starts with a configuration in which each fragile link is activated. In iteration k it computes the expected first hitting time to v if we start in i and use the current configuration, that is it calculates

$$H_k(i) \triangleq \mathbb{E}\left[\inf\{t \geq 1 : X_t = v\} \mid X_0 = i\right], \tag{29}$$

for all nodes i, where the transition matrix of the Markov chain $(X_0, X_1, \ldots)$ is P_k defined by Eq. (1) using the adjacency matrix corresponding to the fragile link configuration in iteration k. Then, the configuration is updated in a greedy way: a fragile link from node i to node j is activated if and only if $H_k(i) \geq H_k(j) + 1$. The algorithm terminates if the configuration cannot be improved by this way. We call this method the *PageRank Iteration* (PRI) algorithm. The pseudo-code of PRI can be found below.

The PageRank Iteration Algorithm

Input:	A digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a node $v \in \mathcal{V}$ and a set of fragile links $\mathcal{F} \subseteq \mathcal{E}$.	
1.	$k := 0$	% initialize the iteration counter
2.	$F_0 := \mathcal{F}$	% initialize the starting configuration
3.	*Repeat*	% iterative evaluation and improvement
4.	$\quad H_k := \mathbb{1}^{\mathrm{T}}(I - Q_k)^{-1}$	% compute the mean hitting times to v
5.	$\quad F_{k+1} := \{(i,j) \in \mathcal{F} : H_k(i) \geq H_k(j) + 1\}$	% improve the configuration
6.	$\quad k := k + 1$	% increase the iteration counter
7.	*Until* $F_{k-1} \neq F_k$	% until no more improvements are possible
Output:	$1/H_k(v)$, the Max-PageRank of v, and F_k, an optimal configuration.	

Note that the expected first hitting times can be calculated by a system of linear equations [17]. In our case, the vector of hitting times, H_k, is

$$H_k = \mathbb{1}^{\mathrm{T}}(I - Q_k)^{-1}, \tag{30}$$

where Q_k is obtained from P_k by setting to zero the row corresponding to node v, namely, $Q_k = diag\,(\mathbb{1} - e_v)\,P_k$, where e_v is the vth n dimensional canonical basis vector. To see why this is true, recall the trick of Section 3.2, when we split node v into a starting node and an absorbing target node. Then, the expected hitting times of the target state can be calculated by the fundamental matrix [15]. If v can be reached from all nodes, then $I - Q_k$ is guaranteed to be invertible. Note that $H_k(v) = \varphi_k(v)$, where $\varphi_k(v)$ is the expected first return time to v under the configuration in iteration k, therefore, the PageRank of v in the kth iteration is $\boldsymbol{\pi}_k(v) = 1/H_k(v)$.

Theorem 3. PageRank Iteration *has the following properties:*

(I) *Assuming* (AD) *and* (AR)*, the algorithm always terminates in a finite number of iterations and the final configuration is optimal.*
(II) *Assuming* (AB)*, it finds an optimal solution in polynomial time.*

Proof. Part I: We can notice that this algorithm is almost the *policy iteration* (PI) method, in case we apply a formulation similar to the previously presented simple SSP formulation. However, it does not check every possible action in each state. It optimizes each fragile link separately, but as the refined SSP formulation demonstrates, we are allowed to do so. Consequently, PRI is the policy iteration algorithm of the refined SSP formulation. However, by exploiting the special structure of the auxiliary states corresponding to the fragile links, we do not have to include them explicitly. For all allowed policies μ (for all configurations of fragile links) we have

$$J^{\mu}(f_{ij}) = \begin{cases} J^{\mu}(i) & \text{if } \mu(f_{ij}) = \text{“off”}, \\ J^{\mu}(j) + 1 & \text{if } \mu(f_{ij}) = \text{“on”}, \end{cases} \tag{31}$$

for all auxiliary states f_{ij} corresponding to a fragile link. Thus, we do not have to store the value of these states, since they can be calculated if needed.

Notice that $J^{\mu_k}(i) = H_k(i)$, where μ_k is the policy corresponding to the configuration in iteration k. Thus, calculating H_k is the *policy evaluation* step of PI, while computing F_{k+1} is the *policy improvement* step. Since PRI is a PI algorithm, it follows that it always terminates finitely and finds an optimal solution [5] if we start with a *proper* policy and under assumptions (A1) and (A2). Recall that the initial policy is defined by the full configuration, $F_0 = \mathcal{F}$ and that we assumed (AR), that is node v can be reached from all nodes for at least one configuration which means that the corresponding policy is proper. If this holds for an arbitrary configuration, it must also hold for the full configuration, therefore, the initial policy is always proper under (AR). Assumption (A1) immediately follows from (AR) and assumption (A2) follows from the fact that if the policy is improper, we must take infinitely often fixed or activated fragile links with probability one. Since each of these edges has unit cost, the total cost is infinite for at least one state.

Part II: First, note that assumption (AB) implies (AR) and (AD), therefore, we know from Part I that PRI terminates in finite steps with an optimal solution. Calculating the mean first hitting times, H_k, basically requires a matrix inversion, therefore, it

can be done in $O(n^3)$. In order to update the configuration and obtain F_{k+1}, we need to consider each fragile link individually, hence, it can be computed in $O(d)$. Consequently, the problem of whether PRI runs in polynomial time depends only on the number of iterations required to reach an optimal configuration.

Since we assumed (AB), there is a universal constant κ such that for all nodes of the graph there is a directed path of fixed edges from this node to node v which path has at most κ edges. These paths contain fixed (not fragile) edges, therefore, even if all fragile links are deactivated, node v can still be reached with positive probability from all nodes. Consequently, all policies are proper (APP). It is easy to see that we can partition the state space to subsequent classes of states $S_1, \ldots, S_r$, where $r \leq \kappa$, by allocating node i to class S_q if and only if the smallest length path of fixed edges that leads to node v has length q. This partition satisfies the required property described in Section 2.3. Because PRI is a PI variant, PRI terminates with an optimal solution in iterations bounded by a polynomial in L and $\eta^{-2\kappa}$. Since $\eta = 1/m$, where $m \leq n$ is the maximum out-degree in the graph, $\eta^{-2\kappa} = O(n^{2\kappa})$, therefore, PRI runs in polynomial time. □

Though, for the sake of concision, we only presented PRI for the problem without damping and personalization, it is easy to modify the algorithm for the other case, as well. Moreover, since if we apply damping each node can be reached from all other nodes by a constant number of edges, namely via the teleportation state, assumption (AB) is automatically satisfied. Then, the smallest transition probability of the associated SSP problem may be determined by the damping constant and the personalization vector, however, this can be arbitrary small. On the other hand, if the damping constant and the personalization vector are *fixed*, not part of the input, we do not have this problem and hence PRI finds an optimal solution in polynomial time.

5. 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 resulting problem is summarized below.

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 $\boldsymbol{z}$.
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, we will show that the decision version of it is NP-complete. In the decision version, one is given a real number $p \in (0, 1)$ and is asked whether there is a fragile link configuration such that the PageRank of a given node v is larger or equal to p.

Theorem 4. *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. Eq. (3), whether the corresponding PageRank is larger or equal to p.

We now reduce the 3SAT problem, whose NP-completeness is well known [12], 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 the Max-PageRank problem 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 constraints, $\mathcal{C}$, 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 $c = 1/(100\,\mathrm{m})$, 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 $(1-c)\frac{1}{m}(1-c)^{l-2}c$. Therefore, the expected return time is larger than

$$E_1 \geq \sum_{l=3}^{\infty} lp(l) \geq \frac{c}{m}\sum_{l=3}^{\infty} l(1-c)^{l-1} \geq \frac{c}{m}\left[c^{-2}-3\right] \geq 99, \tag{32}$$

where we assumed that $c = 1/(100\text{ m})$ and the personalization vector is $\boldsymbol{z} = (1/n)\,\mathbb{1}$. 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 us 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 $1/8 > 1/(100\text{ m})$, 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 original instance of 3SAT is satisfiable. □

We have tried to keep the NP-hardness proof as short as possible. Several variants are possible. In the above construction, each clause node has three parallel edges linking it to the node t. This might seem not elegant, but it is not difficult to get rid of them by adding auxiliary nodes. Also, it is not difficult to get rid of the self-loops by adding auxiliary nodes. Finally we have not tried to optimize the factor $c = 1/(100\text{ m})$, nor the bound on E_2. An interesting question is whether a reduction is possible if the damping factor c and the personalization vector $\boldsymbol{z}$ cannot depend on the instance.

6. Conclusions

The task of ordering the nodes of a directed graph according to their *importance* arises in many applications from the problem of ranking the results of web-searches to bibliometrics and ecosystems. A promising and popular way to define such an ordering is to use the *PageRank* method [6] and associate the importance of a node with the weight of the node with respect to the stationary distribution of a uniform random walk. The problem of optimizing the PageRank of a given node by changing some of the edges caused a lot of recent interest [20,1,9,14,11]. We considered the general problem of finding the extremal values of the PageRank a given node can have in the case we are allowed to control (activate or deactivate) some of the edges from a given *arbitrary* subset of edges, which we referred to as *fragile links*.

Our main contribution is that we proved that this general problem can be solved optimally in *polynomial time* under the Turing model of computation, even if the *damping constant* and the *personalization vector* are part of the input and independently of the way the *dangling nodes* are handled. The proof is based on reformulating the problem as a *stochastic shortest path* problem (a special Markov decision process) and it results in a linear programming formulation that can then be solved by standard techniques.

This solution is weakly polynomial in general, however, if the damping constant and the personalization vector can be represented with bits polynomial in the number of nodes, it becomes strongly polynomial.

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.

Based on the observation that in some of the states of the reformulated SSP problem there is only one available action (thus, we do not have a real choice in them), we showed that the number of states (and therefore the needed computation to solve the problem) could be further reduced.

We also suggested an alternative greedy solution, called the *PageRank Iteration* (PRI) algorithm, which had appealing properties. We analyzed PRI for the Max-PageRank problem *without damping* and showed that it can find an optimal solution in finite steps and it runs in *polynomial time*, under the *bounded reachability* assumption. This latter assumption is always satisfied if we consider the problem *with damping* which indicates that PRI always finds an optimal solution in polynomial time for such problems, in case the damping constant and the personalization vector are *fixed*.

Finally, 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 slightly 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. We left their analysis for further work.

Acknowledgments

The authors are grateful for the valuable discussions to Paul Van Dooren, Yurii Nesterov, Cristobald de Kerchove, Vincent Traag and Tzvetan Ivanov. B. Csáji is an ARC DECRA fellow and R. Jungers is an F.R.S.-FNRS fellow.

References

[1] K. Avrachenkov, N. Litvak, The effect of new links on Google PageRank, Stoch. Models 22 (2006) 319–331.
[2] P. Berkhin, A survey on PageRank computing, Internet Math. (2005) 73–120.
[3] D.P. Bertsekas, Dynamic Programming and Optimal Control, Vol. 2, third ed., Athena Scientific, Belmont, Massachusetts, 2007.
[4] D.P. Bertsekas, J.N. Tsitsiklis, An analysis of stochastic shortest path problems, Math. Oper. Res. 16 (3) (1991) 580–595.
[5] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, Massachusetts, 1996.
[6] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, in: Proceedings of the 7th International World Wide Web Conference, 1998, pp. 107–117.
[7] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, J. Symbolic. Comput. 9 (3) (1990) 251–280.
[8] B.Cs Csáji, V.D. Blondel, R.M. Jungers, PageRank optimization in polynomial time by stochastic shortest path reformulation, in: Procs. of the 21st International Conference on Algorithmic Learning Theory, in: Lecture Notes in Computer Science, vol. 6331, 2010, pp. 89–103.
[9] C. De Kerchove, L. Ninove, P. Van Dooren, Maximizing PageRank via outlinks, Linear Algebra Appl. 429 (2008) 1254–1276.
[10] E.A. Feinberg, A. Shwartz (Eds.), Handbook of Markov Decision Processes: Methods and Applications, Kluwer Academic Publishers, 2002.
[11] O. Fercoq, M. Akian, M. Bouhtou, S. Gaubert, Ergodic control and polyhedral approaches to PageRank optimization, arXiv:1011.2348v1.
[12] M.R. Garey, D.S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., NY, 1990.
[13] C.C. Gonzaga, An algorithm for solving linear programming problems in $o(n^3l)$ operations, in: Progress in Mathematical Programming: Interior-Point and Related Methods, Springer-Verlag, 1988, pp. 1–28.
[14] H. Ishii, R. Tempo, Computing the PageRank variation for fragile web data, SICE J. Control, Meas. Syst. Integr. 2 (1) (2009) 1–9.
[15] J.G. Kemeny, J.L. Snell, Finite Markov Chains, Van Nostrand, 1960.
[16] A.N. Langville, C.D. Meyer, Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, 2006.
[17] D.A. Levin, Y. Peres, E.L. Wilmer, Markov Chains and Mixing Times, American Mathematical Society, 2009.
[18] M.L. Littman, T.L. Dean, L.P. Kaelbling, On the complexity of solving Markov decision problems, in: In Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence, 1995, pp. 394–402.
[19] L. Lovász, Random walks on graphs: a survey, in: Combinatorics: Paul Erdős is Eighty, vol. 2, Bolyai Society Mathematical Studies, Budapest, 1996, pp. 353–348.
[20] M. Olsen, Link building, Ph.D. Thesis, Department of Computer Science, Aarhus University, Denmark, 2009.
[21] C.H. Papadimitriou, J.N. Tsitsiklis, The complexity of Markov decision processes, Math. Oper. Res. 12 (3) (1987) 441–450.
[22] M.L. Puterman, Markov Decision Processes, John Wiley & Sons, 1994.
[23] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 1998.
[24] P. Tseng, Solving H-horizon, stationary Markov decision problems in time proportional to log(H), Oper. Res. Lett. 9 (4) (1990) 287–297.