

PAGERANK OPTIMIZATION BY EDGE SELECTION

BALÁZS CSANÁD CSÁJI^{†‡}, RAPHAËL M. JUNGERS^{†§}, AND VINCENT D. BLONDEL[†]

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 web-page ranking – and can be interpreted as the average portion of time spent at the node by an infinite random walk in the graph. 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 approach provides a linear programming formulation that can then be solved by standard techniques. We also show in the paper that, under the slight modification of the problem for which we are given mutually exclusive pairs of edges, the problem of PageRank optimization becomes NP-hard.

Key words. graph algorithms, computational complexity, Markov decision processes

AMS subject classifications. 05C85, 68Q25, 90C40

1. Introduction. The importance of a node in a directed graph can be measured by its PageRank. The PageRank of a node can be interpreted as the average portion of time spent at the node by an infinite random walk [14]. PageRank is traditionally applied for ordering web-pages of the web-graph in which the edges represent hyperlinks, but it also has many other applications [2], for example, in bibliometrics, ecosystems, spam detection, web-crawling, semantic networks, keyword searching in relational databases and word dependencies in natural language processing.

It is of natural interest to search for the maximum or minimum PageRank that a node (web-page) can have depending on the presence or absence of some of the edges (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 web-pages by suitably placing hyperlinks on his own site or by buying advertisements or making alliances with other sites [1, 8]. 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 web-page by computing the maximum and minimum PageRank that the node may possibly have depending on the presence or absence of the hidden hyperlinks [12]. These hidden edges are often referred to as “fragile links”.

It is known that if we place a new edge in a directed graph, then the PageRank of the target node 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 [8]. 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 subset of edges is proposed by Ishii and Tempo [12]. They are motivated by the problem of “fragile links” and they mention the lack of polynomial time algorithms to this problem. Then, using interval matrices, they propose an approximate solution [12].

[†]Department of Mathematical Engineering, Université catholique de Louvain, Belgium

[‡]Computer and Automation Research Institute (SZTAKI), Hungarian Academy of Sciences

[§]Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, USA

In the paper we show that this 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 of polynomial time computability is based on a *linear programming* formulation, but we also propose an iterative algorithm that has appealing properties, for example, it always finds an optimal solution in finite steps and, under some assumptions, its running time is polynomial. In a last section, we prove that under the slight modification of the problem for which we are given mutually exclusive constraints between pairs of edges, the problem of PageRank optimization becomes NP-hard.

2. Definitions and Preliminaries. In this section we first give an overview on PageRank computation, then we define the problem of optimizing the PageRank of a given node via a subset of edges, called fragile links. We also give a brief introduction to stochastic shortest path problems, since they constitute the basis of our solution.

2.1. PageRank. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph, where $\mathcal{V} = \{1, \dots, n\}$ 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 $1/\text{deg}(i)$ if j is an out-neighbor of i , where $\text{deg}(\cdot)$ denotes out-degree. This defines a *Markov chain* on the nodes of the graph with transition-matrix

$$P \doteq (D_A^{-1}A)^T \quad \text{with} \quad D_A \doteq \text{diag}(A\mathbf{1}) \quad (2.1)$$

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

The PageRank vector, $\boldsymbol{\pi}$, of the graph is defined as the *stationary* (steady-state) *distribution* of the above described homogeneous Markov chain, more precisely, as

$$P\boldsymbol{\pi} = \boldsymbol{\pi}, \quad (2.2)$$

where $\boldsymbol{\pi}$ is non-negative and $\boldsymbol{\pi}^T\mathbf{1} = 1$. Since P is an irreducible stochastic matrix, we know, e.g., from the *Perron-Frobenius* theorem, that $\boldsymbol{\pi}$ exists and it is unique.

Now, we turn to the more 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 dangling nodes [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 once we have reached 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 define a Markov chain similarly to (2.1), but this chain may not have a unique stationary distribution. To solve this problem, the PageRank vector, $\boldsymbol{\pi}$, of \mathcal{G} is defined as the stationary distribution of the “Google matrix” [14] defined as

$$G \doteq (1 - c)P + c\boldsymbol{z}\mathbf{1}^T, \quad (2.3)$$

where \boldsymbol{z} is a positive “personalization” vector satisfying $\boldsymbol{z}^T\mathbf{1} = 1$, and $c \in (0, 1)$ is a “damping” constant. In practice, values between 0.1 and 0.15 are usually applied for

c and $z = (1/n)\mathbf{1}$ [2]. The Markov chain defined by G is *regular* (or ergodic) that is *irreducible* and *aperiodic*, therefore, its stationary distribution, π , uniquely exists and, moreover, the Markov chain converges to π from any initial distribution [15].

The idea of PageRank is that $\pi(i)$ can be interpreted as the “importance” of i . Thus, π defines a *linear order* on the nodes by treating $i \leq j$ if and only if $\pi(i) \leq \pi(j)$.

The PageRank vector can be approximated by the *power method*, namely, by

$$x_{n+1} \doteq Gx_n, \quad (2.4)$$

where x_0 is an arbitrary stochastic vector. It can also be directly computed by

$$\pi = c(I - (1 - c)P)^{-1}z, \quad (2.5)$$

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, therefore, it is invertible.

2.2. PageRank Optimization. We will investigate the problem when 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 [12]. 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. 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 respect to the graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E} \setminus \mathcal{F}_-)$. The problem is that how should we configure the fragile links, in order to maximize (or minimize) the PageRank of v ? The Max-PageRank problem can be summarized as follows:

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.

The Min-PageRank problem can be stated similarly. We will concentrate on the Max problem but a straightforward modification of our approach can deal with the Min problem, as well. We will show that Max-PageRank can be solved in polynomial time even if the damping constant and the personalization vector are part of the input.

There are finitely many configurations, consequently, we can try to compute them one-by-one. If we have d fragile links, then there are 2^d possible graphs that we can achieve. Given a graph, we can compute its PageRank vector in $O(n^3)$, via a matrix inversion¹. The resulting “brute force” 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 [17] that a random walk on an undirected graph (a time-reversible Markov chain) has the stationary distribution $\pi(i) = \text{deg}(i)/2m$ for all i , where m denotes the number of edges and $\text{deg}(i)$ is the degree of i . Hence, in order to maximize the PageRank of v , we should keep edge $(i, j) \in \mathcal{F}$ if and only if $i = v$ or $j = v$.

¹It can be done a little bit faster, in $O(n^{2.376})$, using the Coppersmith-Winograd algorithm [7].

2.3. Stochastic Shortest Path Problems. In this section we give an overview on stochastic shortest path problems, since our solutions are built upon their theory.

Stochastic shortest path (SSP) problems are generalizations of (deterministic) shortest path problems [4, 5]. In an SSP problem the transitions between the nodes are uncertain, but we have some control over their probability distribution. We aim at finding a control policy (a function from nodes to controls) such that by applying this policy we can reach a given target state with probability one while minimizing the expected sum of costs (which depends on the applied controls), as well. SSP problems are special undiscounted *Markov decision processes* (MDPs) with finite state and action spaces and with an absorbing, cost-free terminal state. They are of high practical importance since they arise in many real-world domains, for example, several problems in operations research can be formulated as an SSP problem or contains it as a sub-problem. Typical examples are routing and resource allocation problems.

2.3.1. Problem Formulation. 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 $\mathbb{S} = \{1, \dots, n, n+1\}$, 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 $p(j|i, u)$ 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. Namely, for all state i, j and action $u \in \mathcal{U}(i)$, $p(j|i, u) \geq 0$ and

$$\sum_{j=1}^{n+1} p(j|i, u) = 1. \quad (2.6)$$

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 action $u \in \mathcal{U}(\tau)$,

$$p(\tau|\tau, u) = 1, \quad g(\tau, u, \tau) = 0. \quad (2.7)$$

The problem is to find a control *policy* such that it reaches state τ with probability one and minimizes the expected cumulative costs, as well. A (stationary, Markovian) *deterministic* policy is a function from states to actions: $\mu : \mathbb{S} \rightarrow \mathbb{U}$. A *randomized* policy can be formulated as $\mu : \mathbb{S} \rightarrow \Delta(\mathbb{U})$, where $\Delta(\mathbb{U})$ denotes the set of all probability distributions over set \mathbb{U} . It can be shown that every policy induces a *Markov chain* on the state space [9]. 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 that we incur starting from a state and following μ thereafter; that is, for all state i , it is defined as

$$J^\mu(i) \doteq \lim_{k \rightarrow \infty} \mathbb{E}_\mu \left[\sum_{t=0}^{k-1} g(i_t, u_t, i_{t+1}) \mid i_0 = i \right], \quad (2.8)$$

where i_t and u_t are random variables representing the state and the action taken at time t , respectively. Naturally, state i_{t+1} is of distribution $p(\cdot|i_t, u_t)$ and action u_t is of distribution $\mu(i_t)$, or $u_t = \mu(i_t)$ if the applied policy was deterministic.

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 policy, as well.

2.3.2. Bellman Equations. We say that policy μ_1 is better than or equal to policy μ_2 if for all state 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 equation*, $TJ^* = J^*$, where T is the *Bellman operator* [5]; that is, for all i ,

$$(TJ)(i) \doteq \min_{u \in \mathcal{U}(i)} \sum_{j=1}^{n+1} p(j | i, u) \left[g(i, u, j) + J(j) \right]. \quad (2.9)$$

The Bellman operator of a (randomized) policy μ is defined for all state i as

$$(T_\mu J)(i) \doteq \sum_{u \in \mathcal{U}(i)} \mu(i, u) \sum_{j=1}^{n+1} p(j | i, u) \left[g(i, u, j) + J(j) \right], \quad (2.10)$$

where $\mu(i, u)$ is the probability that policy μ chooses action u in state i . Naturally, this definition is valid for deterministic policies, as well, since they are special cases of randomized ones. Given the assumptions above we also have $T_\mu J^\mu = J^\mu$ and the standard *value iteration* converges in SSPs [4] for all initial vector J , that is

$$\lim_{k \rightarrow \infty} T_\mu^k J = J^\mu, \quad \lim_{k \rightarrow \infty} T^k J = J^*. \quad (2.11)$$

It is also known that a policy μ is optimal if and only if $T_\mu J^* = TJ^* (= J^*)$. Moreover, the Bellman operator, T , and the Bellman operator of any policy μ , T_μ , are *monotone* and, assuming that (APP) all allowed control policies are proper, T and T_μ are *contractions* with respect to a weighted maximum norm [5].

From a given value function J , it is straightforward to get a policy, for example, by applying a *greedy* (with respect to J) policy; that is, for all state i ,

$$\mu(i) \in \arg \min_{u \in \mathcal{U}(i)} \sum_{j=1}^{n+1} p(j | i, u) \left[g(i, u, j) + J(j) \right]. \quad (2.12)$$

The *policy iteration* (PI) algorithm 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 with respect to J^{μ_k} . The algorithm terminates if $J^{\mu_k} = J^{\mu_{k+1}}$. It is known that, assuming (A1) and (A2), PI generates an improving sequence of proper policies and finds an optimal solution in finite number of iterations [5].

There are many solution methods for solving MDPs, e.g., in the fields of reinforcement learning and dynamic programming. Most of these algorithms aim at finding (or approximating) the optimal value function, since it immediately gives an optimal policy, as well. Such methods are, e.g., value iteration, policy iteration, Gauss-Seidel method, Q-learning, Q(λ), SARSA and TD(λ): temporal difference learning [5, 9, 20].

2.3.3. Computational Complexity. 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 [16]. Moreover, these classes of problems are P-complete [18]. In the case of SSP problems, it is known that they can be

reformulated as *linear programming* (LP) problems [5]. More precisely, the optimal cost-to-go function, $J^*(1), \dots, J^*(n)$, solves the following LP in variables x_1, \dots, x_n :

$$\text{maximize} \quad \sum_{i=1}^n x_i \quad (2.13a)$$

$$\text{subject to} \quad x_i \leq \sum_{j=1}^{n+1} p(j|i, u) \left[g(i, u, j) + x_j \right] \quad (2.13b)$$

for all state i and for all action $u \in \mathcal{U}(i)$. 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 (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.

Though, value iteration usually requires an infinite number of steps to obtain the optimal value function, stopping rules can be defined which guarantee that the greedy policy with respect to the current value function approximation will be optimal [16].

Assuming that all policies are proper (APP), the state space can be partitioned into nonempty subsets S_1, \dots, 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 $j \in \{\tau\} \cup S_1 \cup \dots \cup S_{q-1}$ such that $p(j|i, u) > 0$. 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 of the MDP [21].

Since policy iteration converges no more slowly than value iteration [19], it also terminates in iterations bounded by a polynomial in L and η^{-2r} , assuming (APP).

3. Optimizing PageRank as a Markov Decision Process. Before we prove that efficient (polynomial time) 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 homogeneous Markov chain defined on a finite set Ω . The *expected first return time* of a state $i \in \Omega$ is defined as follows

$$\varphi(i) \doteq \mathbb{E}[\min\{t \geq 1 : X_t = i\} | X_0 = i]. \quad (3.1)$$

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

$$\boldsymbol{\pi}(i) = \frac{1}{\varphi(i)}, \quad (3.2)$$

for all state i , where $\boldsymbol{\pi}$ is the stationary distribution of the chain [15]. 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 equation (3.2) that maximizing (minimizing) the PageRank of a given node is equivalent to the problem of minimizing (maximizing) the expected first return time to this node.

We are going to show that the Max-PageRank problem can be efficiently formulated as a *Markov decision process* (MDP), more precisely, as a *stochastic shortest path* (SSP) problem [4, 5], where “efficiently” means that in 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 to presentation, 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 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 assumption will be equivalent to assuming that there is at least one *proper* policy, which statement can be checked in polynomial time [5]. 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, which implies that all policies are proper.

3.2. Simple SSP Formulation. First, let us consider an instance of the Max-PageRank Problem. We will build an associated SSP problem as follows. 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 SSP problem.

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 this state will be “turned off” (deactivated). Therefore, for each state i , the allowed set of actions is $\mathcal{U}(i) = \mathcal{P}(\mathcal{F}_i)$, where \mathcal{P} denotes the power set and \mathcal{F}_i is the set of all 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 state j that can be reached from state i using a fixed or an activated fragile link (an out-neighbor of i w.r.t. the current configuration) is

$$p(j | i, u) \doteq \frac{1}{a_i + b_i(u)}. \quad (3.3)$$

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

$$g(i, u, j) \doteq \begin{cases} 0 & \text{if } i = v_t, \\ 1 & \text{otherwise,} \end{cases} \quad (3.4)$$

for all state i, j and action 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 an appropriate configuration of fragile links) the expected first return time to state v . Consequently, its solution is equivalent to *maximizing* the PageRank of node v .

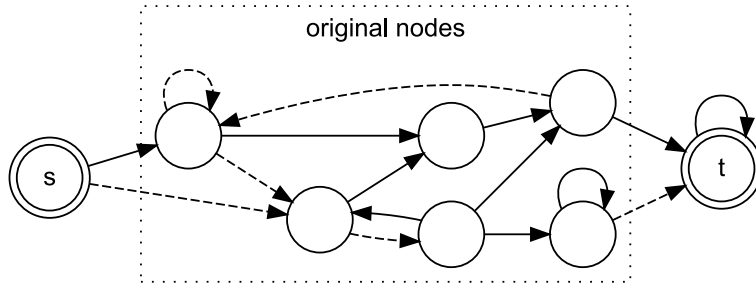


FIGURE 3.1. An example for the SSP reformulation of minimizing the mean first return time. The starting state is $s = v_s$ and the target state is $t = v_t$. The dashed edges denote fragile links.

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 , assuming the fragile link configuration of μ , as

$$\pi(v) = \frac{1}{J^\mu(v_s)}, \quad (3.5)$$

where we applied the convention of $1/\infty = 0$, which is needed when v is not recurrent under the configuration of 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, e.g., 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

$$J^\mu(1) = \mathbf{1}^\top (I - P_\mu)^{-1} e_1, \quad (3.6)$$

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

$$(I - P_\mu)^{-1} = \sum_{n=0}^{\infty} P_\mu^n, \quad (3.7)$$

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

3.3. Computational Complexity. 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 [16, 18]. 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 are going to reduce the maximum number of allowed actions per state to two while only slightly increasing the number of states.

3.4. 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 had 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 was “on” then we went with *probability one* to state j , however, if our action was “off”, then we went back with *probability one* to state i (Figure 3.2).

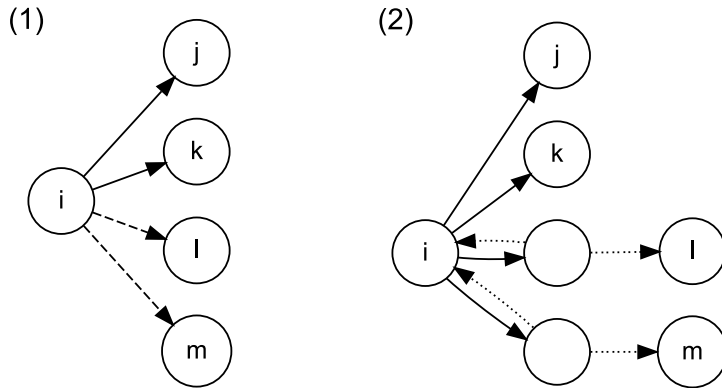


FIGURE 3.2. An example for inserting auxiliary sates for fragile links. The left hand side presents the original situation, in which dashed edges are fragile links. The right hand side shows the refined reformulation, where the dotted edges represent possible actions with deterministic outcomes.

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 links², 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 have returned to state i we

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

have again $1/(a+b)$ probability to go to state j and so on. Now, we are going to compute the probability of eventually arriving at state j if we start in state i .

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 j does not change by this. Then, we have

$$\mathbb{P}(\exists t : X_t = j \mid \forall s < t : X_s = i) = \quad (3.8a)$$

$$= \sum_{t=1}^{\infty} \mathbb{P}(X_t = j \mid X_{t-1} = i) \prod_{s=1}^{t-1} \mathbb{P}(X_s = i \mid X_{s-1} = i) = \quad (3.8b)$$

$$= \sum_{t=1}^{\infty} \frac{1}{a+b} \left(\frac{b-k}{a+b}\right)^{t-1} = \frac{1}{a+b} \sum_{t=0}^{\infty} \left(\frac{b-k}{a+b}\right)^t = \frac{1}{a+k}. \quad (3.8c)$$

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) node 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 i to j according to the original random walk.

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 immediate-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,

$$g(i, u, j) \doteq \begin{cases} 0 & \text{if } i = v_t \text{ or } j = f_{il} \text{ or } u = \text{“off”}, \\ 1 & \text{otherwise,} \end{cases} \quad (3.9)$$

for all state 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 $u = \text{“on”}$. 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 do we visit auxiliary nodes.

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 we can achieve with suitably setting the fragile links is $J^*(v_s)$, where J^* is the optimal cost-to-go function, consequently, the *maximum* PageRank of node v is $1/J^*(v_s)$.

It is also easy to see that if we want to compute the *minimum* possible PageRank of node v , then we should simply define a new immediate-cost function as $\hat{g} = -g$, where g is defined by equation (3.9). If the optimal cost-to-go function of this modified SSP problem is \hat{J}^* , then the *minimum* PageRank that node v can have is $1/|\hat{J}^*(v_s)|$.

3.5. Computational Complexity. 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} ,

$$\text{maximize} \quad \sum_{i \in \mathcal{V}} x_i + \sum_{(i,j) \in \mathcal{F}} x_{ij} \tag{3.10a}$$

$$\text{subject to} \quad x_{ij} \leq x_i, \quad \text{and} \quad x_{ij} \leq x_j + 1, \quad \text{and} \tag{3.10b}$$

$$x_i \leq \frac{1}{deg(i)} \left[\sum_{(i,j) \in \mathcal{E} \setminus \mathcal{F}} (x_j + 1) + \sum_{(i,j) \in \mathcal{F}} x_{ij} \right], \tag{3.10c}$$

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, hence, $v_s, v_t \in \mathcal{V}$. Also note that the value of the target state, x_{v_t} , is fixed at zero.

3.6. Handling Dangling Nodes. Now, we will show that assumption (AD) can be omitted and our result is independent of how the dangling nodes are handled.

Suppose that we have chosen a rule according to which the dangling nodes are handled, e.g., as 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 we can safely restrict the optimization in a way that maximum one of the links can be activated in each fragile node. This does not affect the optimal PageRank, 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, \dots, 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.

3.7. 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 3.6. Note that assumption (AR) is always satisfied in case of damping.

Damping can be interpreted as in each step we continue the random walk with probability $1 - c$ and we restart it with probability c , where $c \in (0, 1)$ is a damping constant. In this latter case, we choose the new starting state according to the probability distribution of a positive and stochastic personalization vector \mathbf{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 called teleportation [14].

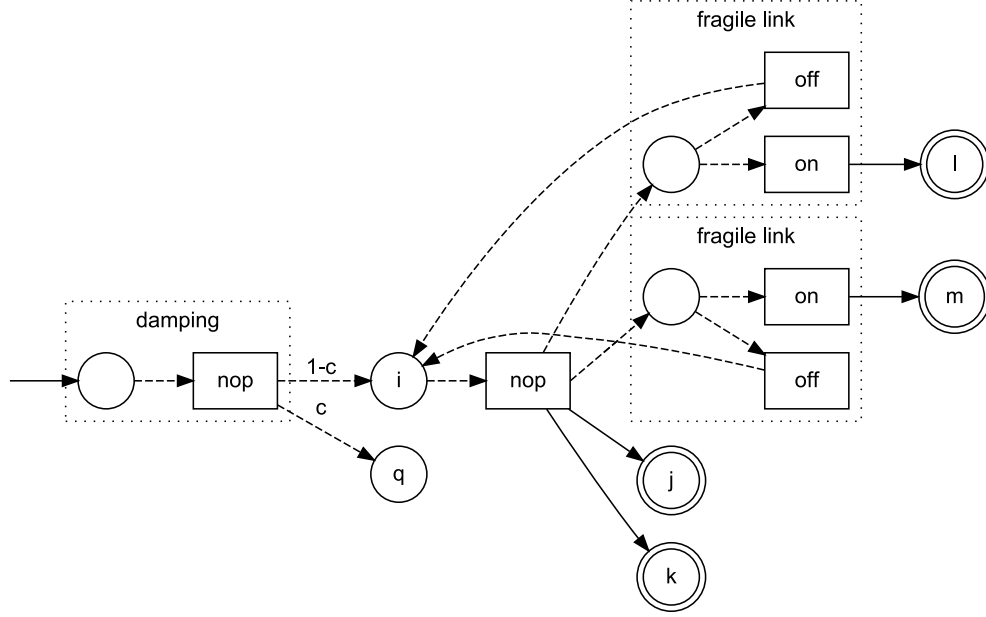


FIGURE 3.3. An illustration of damping: the substructure of a node of the original digraph. Circles represent states and boxes represent actions. The double circles indicate that in fact the actions do not directly lead to these states, but to their similarly defined substructures. State q denotes the global “teleportation” state from which the states can be reached with distribution defined by the personalization vector. Dashed edges help determining zero cost events: if a state-action-state path has only dashed edges, it means that this triple has zero cost, otherwise, its cost is one.

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 Figure 3.3). Each action that led 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 q , we have one available action which brings us with z distribution to the newly defined nodes,

$$p(h_i | q) \doteq p(h_i | q, u) \doteq \begin{cases} z(i) & \text{if } i \neq v_s \text{ and } i \neq v_t \\ z(v) & \text{if } i = v_t \\ 0 & \text{if } i = v_s. \end{cases} \quad (3.11)$$

All other transition-probabilities from q are zero. Regarding the immediate-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 state i and action u .

3.8. Computational Complexity. In the current variant, which also takes damping 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

$$\text{maximize} \quad \sum_{i \in \mathcal{V}} (x_i + \hat{x}_i) + \sum_{(i,j) \in \mathcal{F}} x_{ij} + x_q \quad (3.12a)$$

$$\text{subject to} \quad x_{ij} \leq \hat{x}_j + 1, \quad \text{and} \quad \hat{x}_i \leq (1 - c)x_i + cx_q, \quad (3.12b)$$

$$x_{ij} \leq x_i, \quad \text{and} \quad x_q \leq \sum_{i \in \mathcal{V}} \hat{z}_i (\hat{x}_i + 1), \quad (3.12c)$$

$$x_i \leq \frac{1}{\text{deg}(i)} \left[\sum_{(i,j) \in \mathcal{E} \setminus \mathcal{F}} (\hat{x}_j + 1) + \sum_{(i,j) \in \mathcal{F}} x_{ij} \right], \quad (3.12d)$$

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 notation are the same as in LP (3.10) and we also have that the values 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 [11]. Therefore, Max-PageRank can be solved using $O((n + d)^3 L)$ operations under the Turing model of computation. We can conclude that

THEOREM 3.1 (Main Theorem). *The MAX-PAGERANK PROBLEM can be solved in polynomial time even if the damping constant and the personalization vector are part of the input and independently of the way the dangling nodes are handled.*

Note that assumptions (AD) and (AR) are not needed for this theorem, since dangling and fragile nodes can be treated as discussed in Section 3.6 (without increasing the complexity of the problem) and, in case of damping, all policies are proper.

Assuming the damping constant, c , and the personalization vector, 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 $z = (1/n) \mathbb{1}$ [2], we arrive at a *strongly* polynomial time solution, because the other coefficients can be represented using $O(\log n)$ bits.

3.9. State Space Reduction. In the last SSP formulation in $2n + 1$ states there is no real choice in (there is only one available action), therefore, the state space may be reduced. 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 $O(r^3 + r^2 sm + s^2 rm)$, where m denotes the maximum number of allowed actions per state.

We will apply immediate-cost functions of the form $g : \mathbb{S} \times \mathbb{U} \rightarrow \mathbb{R}$. If we had a cost function that also depended on the arrival state, we could redefine it by

$$g(i, u) = \sum_{j=1}^N p(j | i, u) \tilde{g}(i, u, j), \quad (3.13)$$

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

We will call the states in which there is only one available control 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 will assume, w.l.o.g., that the (indices of the) non-decision states are $1, \dots, r$ and the decision states are $r + 1, \dots, r + s$. Finally, we will assume that there exists at least one proper policy.

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

$$P_\mu = \begin{bmatrix} R_0 & R_\mu \\ Q_0 & Q_\mu \end{bmatrix}, \quad (3.14)$$

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. Notice that R_0 and Q_0 do not depend on the applied control policy.

In the reduced problem we will only keep the s decision states and remove the r non-decision states. We are going to redefine the transition-probabilities (and later the costs) 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 as

$$P_0 = \begin{bmatrix} R_0 & 0 \\ Q_0 & I \end{bmatrix}, \quad (3.15)$$

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. There is at least one proper policy, R_0 and Q_0 are the same for all policies and the target state is a decision state, therefore, R_0^k converges to the zero matrix as $k \rightarrow \infty$, thus

$$\lim_{k \rightarrow \infty} P_0^k = \begin{bmatrix} 0 & 0 \\ Q^* & I \end{bmatrix}, \quad (3.16)$$

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 [13] that these probabilities can be calculated using the *fundamental matrix*, $F = (I - R_0)^{-1}$. More precisely,

$$Q^* = Q_0 F = Q_0 (I - R_0)^{-1}, \quad (3.17)$$

and the computation requires a matrix inversion and a matrix multiplication. If we use classical methods, Q^* can be calculated in $O(r^3 + r^2s)$. Note that we could also apply more efficient algorithms, such as the Coppersmith-Winograd method. Using Q^* the transition matrix of policy μ in the reduced problem should be

$$\widehat{P}_\mu = Q_\mu + Q^* R_\mu. \quad (3.18)$$

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. In order to achieve this, we can use

$$\widehat{p}(j | i, u) \doteq p(j | i, u) + \sum_{k=1}^r p(k | i, u) Q_{jk}^* \quad (3.19)$$

for all $i, j > r$ and $u \in \mathcal{U}(i)$. Note that i and j are decision states. Computing the new transition-probability function needs $O(s^2rm)$ 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 [13]. Therefore, we define the new immediate-cost function as

$$\widehat{g}(i, u) \doteq g(i, u) + \sum_{k=1}^r p(k | i, u) \sum_{j=1}^r F_{jk} g(j), \quad (3.20)$$

for all $i > r$ and $u \in \mathcal{U}(i)$, where we do 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, $g(j) \doteq g(j, u)$, where u denotes the only available control action in state j . It is easy to see that computing the new cost-function needs $O(r^2sm)$ operations, naturally, after we have calculated 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 constructing the new, reduced SSP problem is $O(r^3 + r^2sm + s^2rm)$.

3.9.2. Reducing the SSP formulation of Max-PageRank. Applying this result to the last, refined SSP formulation of the Max-PageRank problem (which also included damping and personalization), we can reduce the number of states of the problem to $d + 1$ by constructing another SSP problem as demonstrated above. The construction takes $O(n^3 + d^2n + n^2d)$ operations, after which the problem can be solved in $O(d^3L)$ by using standard LP solvers [11]. Consequently, we have

LEMMA 3.2. MAX-PAGERANK *can be solved in $O(n^3 + d^3L)$ operations.*

4. PageRank Iteration. In this section we will provide an alternative solution to the Max-PageRank problem. We will define a simple iterative algorithm that in each iteration 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 a new assumption as follows.

(ABR) *Bounded Reachability Assumption:* We assume that the target node, v , can be reached from all nodes 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 of the graph by taking at most κ fixed edges. Constant κ is universal which means that it does not depend on the particular problem instance.

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

$$H_k(i) \doteq \mathbb{E}[\min\{t \geq 1 : X_t = v\} \mid X_0 = i], \quad (4.1)$$

for all node i , where the transition matrix of the Markov chain (X_0, X_1, \dots) is P_k defined by equation (2.1) using the adjacency matrix corresponding to the fragile link configuration in iteration k . Then, the configuration is updated in a simple, 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 algorithm *PageRank Iteration* (PRI) and it can be summarized as follows:

THE PAGERANK ITERATION ALGORITHM

<i>Input:</i>	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.	<i>Repeat</i> % iterative evaluation and improvement
4.	$H_k := \mathbf{1}^T(I - Q_k)^{-1}$ % compute the mean hitting times to v
5.	$F_{k+1} := \{(i, j) \in \mathcal{F} : H_k(i) \geq H_k(j) + 1\}$ % improve the configuration
6.	$k := k + 1$ % increase the iteration counter
7.	<i>Until</i> $F_{k-1} \neq F_k$ % until no more improvements are possible
<i>Output:</i>	$1/H_k(v)$, 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 [15]. In our case, the vector of hitting times, H_k , can be computed as

$$H_k = \mathbf{1}^T(I - Q_k)^{-1}, \quad (4.2)$$

where Q_k is obtained from P_k by setting to zero the row corresponding to node v , namely, $Q_k = \text{diag}(\mathbf{1} - e_v)P_k$, where e_v is the v -th 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 [13]. 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 k -th iteration is $\pi_k(v) = 1/H_k(v)$.

THEOREM 4.1. *PAGERANK ITERATION has the following properties:*

- (I) *Assuming (AD) and (AR), it terminates finitely with an optimal solution.*
- (II) *Assuming (ABR), 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 policy μ (for all configurations of fragile links) we have

$$J^\mu(f_{ij}) = \begin{cases} J^\mu(i) & \text{if } \mu(f_{ij}) = \text{“off”}, \\ J^\mu(j) + 1 & \text{if } \mu(f_{ij}) = \text{“on”}, \end{cases} \quad (4.3)$$

for all auxiliary states f_{ij} corresponding to a fragile link. Therefore, we do not have to store the cost-to-go of these states, because 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. Taking each one of these edges has unit cost, consequently, the total cost is infinite for at least one state.

Part II: First, note that assumption (ABR) 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 (ABR), 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, \dots, 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 2.3.3. 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 under the Turing model of computation. \square

5. PageRank Optimization with Constraints. In this section we are going to investigate a variant of the PageRank optimization problem in which there are mutual 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.

We saw that the Max-PageRank problem can be solved in polynomial time even if the damping constant and the personalization vector are part of the input. However, we will see that the Max-PageRank problem under exclusive constraints is already

NP-hard, more precisely, we will show that the decision version of this problem 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 is larger or equal to p . The proof is based on a reduction from the 3SAT problem.

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.

THEOREM 5.1. *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 set of activated fragile links), it is easy to *verify* in polynomial time, e.g, by a matrix inversion, cf. equation (2.5), whether the corresponding PageRank is larger or equal than p .

We now reduce the 3SAT problem, whose NP-completeness is well known [10], 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 are going to construct an instance of the Max-PageRank Problem under exclusive constraints that solve 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 pair $(y_{j,l}, y_{j',l'})$ 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 pair of edges $(y_{j,l}, y_{j,l'})$ corresponding to a same clause node, we forbid the corresponding pair. This reduction is suitable, since our graph and the set \mathcal{C} have a size which is *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 first activate all edge which correspond to a literal which is true, and second, if necessary, we deactivate some edges so that for all clause node, there is exactly one leaving edge to t . This graph, which is clearly satisfiable, is strongly connected, and so the expected return time from t to itself 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 a leaving edge going to

node 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

$$E_1 \geq \sum_{l=3}^{\infty} lp(l) \geq \frac{c}{m} \sum_{l=3}^{\infty} l(1 - c)^{l-1} \geq \frac{c}{m} [c^{-2} - 3] \geq 99, \quad (5.1)$$

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

Consider now a satisfiable instance, and build a corresponding graph so that for all clause node, 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, this 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. \square

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 self-loops by adding auxiliary nodes. Finally we have not tried to optimize the factor $c = 1/(100m)$, nor the bound on E_2 . An interesting question is whether a reduction is possible if the damping factor c and the personalization vector 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. A promising and popular way to define such an ordering is to use the PageRank method [6]. The problem of optimizing the PageRank of a given node by changing some of the edges caused a lot of recent interest [1, 8, 12]. The paper considered a 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, which we referred to as ‘‘fragile links’’.

Our main contribution was that we proved that this general problem could be solved optimally in polynomial time under the Turing model of computation, even if the damping constant and the personalization vector were part of the input. The proof is based on reformulating the problem as a stochastic shortest path (SSP) problem (a special Markov decision process) and it results in a linear programming formulation that can be solved by standard techniques, e.g., interior point methods.

Note that we do not need to assume that the graph is simple, namely, it can have multiple edges (and self-loops). Hence, our result can be generalized to PageRank optimization on 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, 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 solution, called the PageRank Iteration (PRI) algorithm, which had many appealing properties, for example, it always finds an optimal solution in finite steps and it runs in polynomial time, under the bounded reachability assumption. We left open the question of knowing whether PRI runs in polynomial time under milder conditions. The analysis and generalization of the PRI algorithm in case of damping and personalization is also a potential research direction.

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

7. Acknowledgments. The authors are very grateful for the friendly and valuable discussions to Paul Van Dooren, Yurii Nesterov, Cristobald de Kerchove, Vincent Traag and Tzvetan Ivanov which helped to improve the paper to a great extent.

REFERENCES

- [1] K. Avrachenkov and N. Litvak. Decomposition of the Google PageRank and optimal linkage strategy. Technical report, INRIA, 2004.
- [2] P. Berkhin. A survey on PageRank computing. *Internet Mathematics*, 2(1):73–120, 2005.
- [3] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, Belmont, Massachusetts, 3rd edition, 2007.
- [4] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.
- [5] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts, 1996.
- [6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International Conference on World Wide Web*, pages 107–117, 1998.
- [7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.
- [8] C. De Kerchove, L. Ninove, and P. Van Dooren. Maximizing PageRank via outlinks. *Linear Algebra and its Applications*, 429:1254–1276, 2008.
- [9] E. A. Feinberg and A. Shwartz, editors. *Handbook of Markov Decision Processes: Methods and Applications*. Kluwer Academic Publishers, 2002.
- [10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, 1990.
- [11] C. C. Gonzaga. *Progress in Mathematical Programming: Interior-Point and Related Methods*, chapter An Algorithm for Solving Linear Programming Problems in $O(n^3L)$ operations, pages 1–28. Springer-Verlag, 1988.
- [12] H. Ishii and R. Tempo. Computing the PageRank variation for fragile web data. *SICE J. of Control, Measurement, and System Integration*, 2(1):1–9, 2009.
- [13] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Van Nostrand, 1960.
- [14] A. N. Langville and C. D. Meyer. *Google’s PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.
- [15] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.
- [16] M. L. Littman, T. L. Dean, and L. P. Kaelbling. On the complexity of solving Markov decision problems. In *In Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, pages 394–402, 1995.
- [17] L. Lovász. Random walks on graphs: A survey. In *Combinatorics: Paul Erdős is Eighty*, volume 2, pages 353–348. Bolyai Society Mathematical Studies, Budapest, 1996.
- [18] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.
- [19] M. L. Puterman. *Markov decision processes*. John Wiley & Sons, 1994.
- [20] R. S. Sutton and A. G. Barto. *Reinforcement learning*. The MIT Press, 1998.
- [21] P. Tseng. Solving H-horizon, stationary Markov decision problems in time proportional to $\log(H)$. *Operations Research Letters*, 9(4):287–297, 1990.