Portfolio Optimization for Influence Spread

Naoto Ohsaka
The University of Tokyo
ohsaka@is.s.u-tokyo.ac.jp

Yuichi Yoshida
National Institute of Informatics
and
Preferred Infrastructure, Inc.
yyoshida@nii.ac.jp

ABSTRACT

Motivated by viral marketing, stochastic diffusion processes that model influence spread on a network have been studied intensively. The primary interest in such models has been to find a seed set of a fixed size that maximizes the expected size of the cascade from it. Practically, however, it is not desirable to have the risk of ending with a small cascade, even if the expected size of the cascade is large. To address this issue, we adopt conditional value at risk (CVaR) as a risk measure, and propose an algorithm that computes a portfolio over seed sets with a provable guarantee on its CVaR. Using real-world social networks, we demonstrate that the portfolio computed by our algorithm has a significantly better CVaR than seed sets computed by other baseline methods.

1. INTRODUCTION

With the increasing popularity of online social networking services, analyzing the spread of rumors, news, and memes over social networks has become more important. One of the central applications of the spread of such influence is *viral marketing* [11, 21]. In viral marketing, a company promotes a new product by distributing its free or discount samples to a small group of influential individuals with the aim of triggering a large cascade of product adoptions through the word-of-mouth effects. Hence, it is essential to extract highly influential individuals from social networks for successful marketing.

The problem of finding such individuals was formulated by Kempe, Kleinberg, and Tardos [15] as a discrete optimization problem called *influence maximization*, which asks for selecting a seed vertex set of a specific size, say k, that maximizes the "expected" size of cascades under a certain stochastic diffusion model. Although influence maximization was proven to be NP-hard under well-established diffusion models [15], due to the *monotonicity* and *submodularity* of influence functions, the optimal solutions can be approximated within a factor of $1 - 1/e$ by using a simple greedy strategy [19].

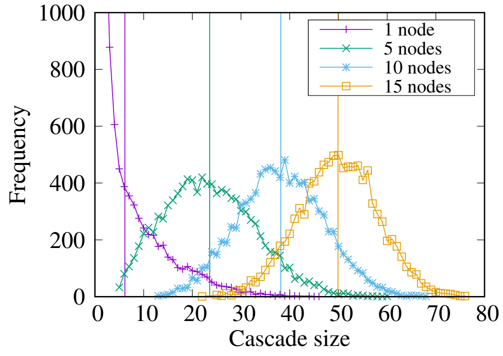


Figure 1: Histogram of cascade sizes for seed sets selected by the standard greedy algorithm in the independent cascade model (see Section 3) on Physicians (see Section 6). Vertical lines correspond to the mean values.

However, solutions obtained by optimizing influence functions are not necessarily appropriate for practical use. In particular, the "risk" of leading cascades that are much smaller than the average has not been taken into account in the framework of influence maximization. To observe whether it occurs in reality, we run a standard greedy algorithm under a certain setting and plot the histogram of cascade sizes for obtained seed sets in Figure 1. The histogram for a seed set of size one is heavily skewed, which means that cascades starting from these seeds will immediately terminate with a large probability. Although the histograms for seed sets of sizes five, ten, and fifteen are roughly symmetrical, the cascade size can be close to zero with a non-negligible probability. This is unfavorable behavior to adopt these seeds as marketing strategies.

1.1 Our contributions

In this study, we consider the problem of finding a low-risk strategy in terms of cascade sizes. In the field of financial economics and actuarial science, various risk measures have been proposed to quantify risk. Here, we adopt the *conditional value at risk* (CVaR) [22, 23], which is one of the most popular risk measures because of its good mathematical properties [1]. In our context, roughly speaking, CVaR measures the expected cascade size in the worst α-fraction of cases, where α is typically chosen to be 0.01 or 0.05.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052628

In contrast to influence maximization, computing or approximating the maximum CVaR of a seed set of a fixed size, say, k, is a difficult task. Indeed, Maehara [17] proved that under a reasonable assumption in computational complexity, there is no polynomial time multiplicative approximation algorithm for this problem. To avoid this difficulty, we take a different approach and consider *portfolio optimization*, in which we select a distribution over seed sets of size k rather than selecting a single seed set of size k.

We explain the effectiveness of using a portfolio over seed sets instead of a single seed set by considering an example of viral marketing. Suppose that a company wants to show ads on various kinds of products to users of a social network. The company hopes that the cascade size is large for *every product*, where the cascade size denotes the number of users who recognize the advertised product through the word-of-mouth effects. Suppose that the company can show ads to k users for each product, where k is determined by the budget. Typically, k is small. If the company selects k users and show ads of all the products to them, then there could be a product that are not recognized well. On the other hand, if the company selects a portfolio and show ads to the k users sampled from the portfolio for each product, we may have a better chance that all the products are well recognized. To find such a portfolio, the notion of CVaR is useful.

The main difficulty in solving our portfolio optimization problem is that, although our problem can be well approximated by a linear program, it has exponentially many variables, that is, weight for each of $\binom{n}{k}$ vertex sets, where n is the number of vertices. Hence, existing algorithms for portfolio optimization cannot be applied to our problem. To overcome this difficulty, we propose a novel approximation algorithm based on the multiplicative weights algorithm [3]. An essential ingredient in the multiplicative weights algorithm is a separation oracle for a constraint that is a convex combination of the constraints in the original problem. Although we cannot precisely check whether the constraint is satisfied, we can approximately check it by solving an influence maximization problem. For any $\epsilon > 0$, our algorithm returns a portfolio consisting of a reasonably small number of seed sets that is guaranteed to have a CVaR of at least the optimum minus $1/e + \epsilon$ (after normalization).

We conduct experiments using real-world social networks and demonstrate that the portfolio computed by our algorithm has a significantly larger CVaR than seed sets computed by the greedy strategy and a degree-based heuristic, which do not make portfolios consisting of two or more seed sets.

1.2 Organization

We discuss related work in Section 2. Then, we introduce basic notions in Section 3. In Section 4, we show that, in order to find a portfolio with a large CVaR, it suffices to sample several realizations of cascades and then to optimize CVaR over them. We describe and analyze our portfolio optimization algorithm in Section 5. We show our experimental results in Section 6 and then conclude in Section 7.

2. RELATED WORK

Several methods based on gradient descent have been proposed to optimize CVaR in the literature [8, 14, 24, 27, 28], and they are effectively used in finance [4, 25] and optimizing Markov decision processes [5, 8, 9, 13]. However we cannot apply those methods to our problem because there are exponentially many variables to optimize.

Zhang *et al.* [31] considered the problem of finding the smallest seed set for which the value at risk (VaR) exceeds a certain threshold. Here, VaR is the α-percentile of the cascade size for some parameter $\alpha \in (0, 1)$. The authors showed that the greedy strategy yields a certain approximation guarantee. The CVaR of a portfolio over seed sets and the VaR of a seed set cannot be directly compared because the CVaR of a seed set is no more than its VaR whereas the maximum CVaR of a portfolio over seed sets is no less than the maximum CVaR of a single seed set. A drawback of their method is that their approximation guarantee depends on somewhat artificial parameters that are hard to compute. In contrast, our approximation guarantee is clean, that is, $1/e + \epsilon$.

Deng *et al.* [10] considered the problem of finding the smallest seed set for which the number of individuals who will be influenced with high probability exceeds a certain threshold. This problem seems nothing to do with optimizing VaR or CVaR. Their algorithm is a merely heuristic and has no approximation guarantee.

Mean-variance optimization [18] is another popular method for computing portfolios. However, it is known to underestimate the risk of rare, but catastrophic events [2], and hence we do not consider it here.

3. PRELIMINARIES

For an integer $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, 2, \ldots, n\}$. For a set V and an integer $k \leq |V|$, we define $\binom{V}{k}$ as the family of subsets of V of size k.

For a random variable X, let $\mathcal{D}_X$ denote the distribution of X. Let $f_X : \mathbb{R} \to \mathbb{R}$ and $F_X : \mathbb{R} \to \mathbb{R}$ be the probability density function and the cumulative distribution function of the distribution $\mathcal{D}_X$, respectively. For a distribution $\mathcal{D}$, $X \sim \mathcal{D}$ means that X is sampled from $\mathcal{D}$. Let $\mathrm{supp}(X)$ denote the support of X. We say that a random variable or a distribution is *discrete* if its support is finite.

We say that a function $f : 2^V \to \mathbb{R}$ is *submodular* if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for any $S, T \subseteq V$.

3.1 Value at risk and conditional value at risk

Let X be a random variable and $\alpha \in (0, 1)$. The *VaR* of X at a significance level α, denoted by $\mathrm{VaR}_\alpha(X)$, is the α-percentile of X:

$$\mathrm{VaR}_\alpha(X) = \inf\{\tau \in \mathbb{R} : F_X(\tau) \geq \alpha\}.$$

The *CVaR (of profit)* of X at a significance level α [7], denoted by $\mathrm{CVaR}_\alpha(X)$, is defined as

$$\mathrm{CVaR}_\alpha(X) = \frac{1}{\alpha} \int_0^\alpha \mathrm{VaR}_\gamma(X) \mathrm{d}\gamma.$$

It is known that $\mathrm{CVaR}_\alpha(X)$ can be written as a solution to the following optimization problem [22]:

$$\mathrm{CVaR}_\alpha(X) = \max\Bigl\{\tau - \frac{1}{\alpha} \mathop{\mathbf{E}}_{X}[\max\{\tau - X, 0\}] : \tau \in \mathbb{R}\Bigr\}. \quad (1)$$

The maximum is attained by choosing $\tau = \mathrm{VaR}_\alpha(X)$ [22]. CVaR is a more common risk measure than VaR because CVaR satisfies several useful mathematical properties such as superadditivity and concavity in X [1].

Remark 3.1. *In the standard definition of CVaR, we consider the opposite (right) tail of the distribution because X usually represents loss and we want to avoid a large loss. We adopt the present definitions because we regard X as profit and want to avoid a small profit.*

3.2 Influence spread in the independent cascade model

We introduce the *independent cascade* (IC) model, a widely adopted stochastic model for influence spread posed by Goldenberg et al. [12]. In the IC model, each vertex is either active or inactive. Given a digraph $G = (V, E)$ with an edge probability p_e for each edge e and a seed set S of initially active vertices, the diffusion process continues in discrete steps according to the following randomized rule. When a vertex u becomes active for the first time, then u tries to activate each of its current inactive out-neighbors v. Each trial independently succeeds with probability p_{uv}, and in the case of success, v becomes active at the next step. In the subsequent steps, u cannot make any further trial. The process is repeated until no new activation is possible. Let $R_G(S)$ be the (discrete) random variable representing the number of vertices influenced by S in the IC model. The *influence function* $f_G : 2^V \to \mathbb{R}_+$ is defined as $f_G(S) = \mathbf{E}[R_G(S)]$. It is known that f_G is submodular [15].

3.3 Portfolio optimization for influence spread

For a finite set V and an integer $k \in \mathbb{N}$, let $\Delta_{V,k}$ be the $\binom{|V|}{k}$-dimensional simplex indexed by a set in $\binom{V}{k}$, that is, $\Delta_{V,k} = \{\boldsymbol{v} \in \mathbb{R}^{\binom{V}{k}} \mid \sum_{S \in \binom{V}{k}} v_S = 1\}$. In the portfolio optimization problem for influence spread in the IC model, the input consists of a digraph $G = (V, E)$ on n vertices with edge probabilities, an integer $k \in \mathbb{N}$, and a significance parameter $\alpha \in (0, 1)$. Then, the objective is to find a portfolio $\boldsymbol{\pi} \in \Delta_{V,k}$ over sets of size k that maximizes the CVaR of influenced vertices at significance level α, that is, $\mathrm{CVaR}_\alpha(\sum_{S \in \binom{V}{k}} \pi_S R_G(S))$.

4. APPROXIMATION OF CVAR BY EMPIRICAL CVAR

Let X be a random variable. For an integer $s \in \mathbb{N}$, we define the *empirical distribution* made by s samples from X, denoted by $\hat{\mathcal{D}}_{X,s}$, as the uniform distribution over $\{X^1, \ldots, X^s\}$, where $X^1, \ldots, X^s$ are independent samples from $\mathcal{D}_X$. Note that $\hat{\mathcal{D}}_{X,s}$ itself is a random variable. Let $\hat{X} \sim \hat{\mathcal{D}}_{X,s}$ be a random variable. In this section, we show that, if s is large enough, then $\mathrm{CVaR}_\alpha(X)$ and $\mathrm{CVaR}_\alpha(\hat{X})$ do not differ much, with high probability over the choice of $\hat{\mathcal{D}}_{X,s}$. More formally, we show the following.

Lemma 4.1. *Let X be a discrete random variable bounded in $[0, 1]$ and $\alpha, \epsilon, \delta \in (0, 1)$. Let $\hat{X} \sim \hat{\mathcal{D}}_{X,s}$ be a random variable for $s = \Theta(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$. Then, with probability at least $1 - \delta$, we have*

$$|\mathrm{CVaR}_\alpha(X) - \mathrm{CVaR}_\alpha(\hat{X})| \le \epsilon.$$

For a discrete random variable X, we create another random variable X^c as follows: We choose a sufficiently small value $\xi > 0$. Then, we sample a random variable X and we sample X^c from the uniform distribution over $[X - \xi, X + \xi]$. The advantage of considering X^c instead of X is that the cumulative distribution function is continuous. In particular, $F_{X^c}(\mathrm{VaR}_\alpha(X^c)) = \alpha$ holds. On the other hand, $F_X(\mathrm{VaR}_\alpha(X))$ may be different from α.

We have the following three lemmas.

Lemma 4.2. *Let X be a discrete random variable and $\alpha, \epsilon \in (0, 1)$. Then, we have*

$$|\mathrm{CVaR}_\alpha(X) - \mathrm{CVaR}_\alpha(X^c)| \le \xi.$$

Proof. Note that $|\mathrm{VaR}_\gamma(X) - \mathrm{VaR}_\gamma(X^c)| \le \xi$ for every γ because the probability mass at $a \in \mathrm{supp}(X)$ is distributed to $[a - \xi, a + \xi]$. Then, we have

$$\begin{aligned}
&|\mathrm{CVaR}_\alpha(X) - \mathrm{CVaR}_\alpha(X^c)| \\
&= \frac{1}{\alpha}\Big|\int_0^\alpha \mathrm{VaR}_\gamma(X) - \mathrm{VaR}_\gamma(X^c)\mathrm{d}\gamma\Big| \\
&\le \frac{1}{\alpha}\int_0^\alpha \Big|\mathrm{VaR}_\gamma(X) - \mathrm{VaR}_\gamma(X^c)\Big|\mathrm{d}\gamma \\
&\le \xi.
\end{aligned}$$ □

Lemma 4.3. *Let X and Y be discrete random variables. Suppose $\sup_{x \in \mathbb{R}} |F_X(x) - F_Y(x)| \le \epsilon$ for some $\epsilon > 0$. Then, we have*

$$\sup_{x \in \mathbb{R}} |F_{X^c}(x) - F_{Y^c}(x)| \le \epsilon$$

by choosing a small enough ξ ($\xi < \min\{|a - b| : a, b \in \mathrm{supp}(X) \cup \mathrm{supp}(Y), a \neq b\}$ is enough).

Proof. It suffices to show that $|F_{X^c}(x) - F_{Y^c}(x)| \le \epsilon$ holds for every $x \in [a - \xi, a + \xi]$, where a is an element in $\mathrm{supp}(X) \cup \mathrm{supp}(Y)$. Indeed, we have

$$\begin{aligned}
&|F_{X^c}(x) - F_{Y^c}(x)| \\
&\le \max\{|F_{X^c}(a - \xi) - F_{Y^c}(a - \xi)|, |F_{X^c}(a + \xi) - F_{Y^c}(a + \xi)|\} \\
&\qquad\qquad \text{(by piecewise linearity of } F_{X^c} \text{ and } F_{Y^c}.) \\
&= \max\{|F_X(a - \xi) - F_Y(a - \xi)|, |F_X(a + \xi) - F_Y(a + \xi)|\} \\
&\le \epsilon.
\end{aligned}$$ □

Lemma 4.4. *Let X and Y be random variables bounded in $[-c, c]$ whose cumulative distribution functions are continuous. Suppose*

$$\sup_{x \in \mathbb{R}} |F_X(x) - F_Y(x)| \le \epsilon$$

for some $\epsilon > 0$. Then, we have

$$|\mathrm{CVaR}_\alpha(X) - \mathrm{CVaR}_\alpha(Y)| \le (1 + c)\epsilon.$$

Proof. Let $\tau = \mathrm{VaR}_\alpha(X)$ and $\tau' = \mathrm{VaR}_\alpha(Y)$. Then, we have

$$\begin{aligned}
&|\mathrm{CVaR}_\alpha(X) - \mathrm{CVaR}_\alpha(Y)| \\
&= \Big|\mathop{\mathbf{E}}_X[X \mid X \le \tau] - \mathop{\mathbf{E}}_Y[Y \mid Y \le \tau']\Big| \\
&\le \Big|\mathop{\mathbf{E}}_X[X \mid X \le \tau] - \mathop{\mathbf{E}}_Y[Y \mid Y \le \tau]\Big| \\
&\quad + \Big|\mathop{\mathbf{E}}_Y[Y \mid Y \le \tau] - \mathop{\mathbf{E}}_Y[Y \mid Y \le \tau']\Big|
\end{aligned}$$

The first term is bounded by

$$\Big|F_X(\tau) - F_Y(\tau)\Big| \le \epsilon.$$

The second term is bounded by

$$\begin{aligned}\Big|\int_{\tau'}^{\tau} x f_Y(x)\mathrm{d}x\Big| &\le c\Big|\int_{\tau'}^{\tau} f_Y(x)\mathrm{d}x\Big| \\ &= c|F_Y(\tau) - F_Y(\tau')| \\ &\le c|F_Y(\tau) - F_X(\tau)| + c|F_X(\tau) - F_Y(\tau')| \\ &\le c\epsilon.\end{aligned}$$

Here, we use the fact that $F_X(\tau) = F_Y(\hat{\tau}) = \alpha$ because $F_X(\cdot)$ and $F_Y(\cdot)$ are continuous. Combining these two inequalities, we obtain the desired result. □

Now we consult the following lemma:

Lemma 4.5 (Dvoretzky–Kiefer–Wolfowitz inequality)**.** *Let X be a random variable, $s \in \mathbb{N}$, and $\hat{X} \sim \hat{\mathcal{D}}_{X,s}$. For any $\epsilon > 0$, we have*

$$\sup_{x\in\mathbb{R}} |F_X(x) - F_{\hat{X}}(x)| < \epsilon$$

with probability at least $1 - 2\exp(-2s\epsilon^2)$.

Proof of Lemma 4.1. First, we have $|\mathrm{CVaR}_\alpha(X) - \mathrm{CVaR}_\alpha(X^c)| \le \xi$ and $|\mathrm{CVaR}_\alpha(\hat{X}) - \mathrm{CVaR}_\alpha(\hat{X}^c)| \le \xi$ by Lemma 4.2 (Here $\hat{X}^c$ is a continuous version of $\hat{X}$).

Next, we set the hidden constant in s large enough so that, by Lemma 4.5, we obtain $\sup_{x\in\mathbb{R}} |F_X(x) - F_{\hat{X}}(x)| \le \epsilon/4$ with probability at least $1-\delta/2$. Then, we have $\sup_{x\in\mathbb{R}} |F_{X^c}(x) - F_{\hat{X}^c}(x)| \le \epsilon/4$ by Lemma 4.3. It follows that $|\mathrm{CVaR}_\alpha(X^c) - \mathrm{CVaR}_\alpha(\hat{X}^c)| \le (2+\xi)\epsilon/4$ holds by Lemma 4.4 because X^c and $\hat{X}^c$ are bounded in $[-\xi, 1+\xi]$.

Combining these inequalities yields

$$\begin{aligned}&|\mathrm{CVaR}_\alpha(X) - \mathrm{CVaR}_\alpha(\hat{X})| \\ &\le |\mathrm{CVaR}_\alpha(X) - \mathrm{CVaR}_\alpha(X^c)| + |\mathrm{CVaR}_\alpha(X^c) - \mathrm{CVaR}_\alpha(\hat{X}^c)| \\ &\quad + |\mathrm{CVaR}_\alpha(\hat{X}^c) - \mathrm{CVaR}_\alpha(\hat{X})| \\ &\le 2\xi + \frac{(2+\xi)\epsilon}{2} \le \epsilon\end{aligned}$$

by setting ξ small enough. □

5. PORTFOLIO OPTIMIZATION FOR INFLUENCE SPREAD

Let $G = (V, E)$ be a digraph on n vertices with edge probabilities. For a set $S \subseteq V$, let X_S denote a random variable $R_G(S)/n$ bounded in $[0, 1]$. Let $\boldsymbol{X} = (X_S)_{S\in\binom{V}{k}}$. We aim to find a portfolio $\boldsymbol{\pi} \in \Delta_{V,k}$ that maximizes $\mathrm{CVaR}_\alpha(\langle\boldsymbol{\pi}, \boldsymbol{X}\rangle)$. In this section, we show an algorithm with the following guarantee:

Theorem 5.1. *There exists an algorithm that, given a digraph $G = (V, E)$ with edge probabilities, an integer $k \in \mathbb{N}$, and $\alpha, \epsilon, \delta \in (0, 1)$, returns a portfolio $\boldsymbol{\pi} \in \Delta_{V,k}$ such that*

$$\mathrm{CVaR}_\alpha(\langle\boldsymbol{\pi}, \boldsymbol{X}\rangle) \ge \gamma^* - 1/e - \epsilon$$

with probability at least $1-\delta$, where $\gamma^* = \max_{\boldsymbol{\pi}} \mathrm{CVaR}_\alpha(\langle\boldsymbol{\pi}, \boldsymbol{X}\rangle)$ is the optimal CVaR. The running time is $O(\frac{knms}{\epsilon^2})$, where $n = |V|$, $m = |E|$, and $s = \widetilde{O}\big(\frac{1}{\epsilon^2}\big(\frac{k}{\epsilon^2}\log\frac{n}{k} + \log\frac{1}{\delta}\big)\big)$.

Here, $\widetilde{O}(p)$ denotes $O(p\log^c p)$ for some positive integer $c \in \mathbb{N}$. Our algorithm is based on multiplicative weights algorithm, introduced in Section 5.1. We provide a detailed description and analysis of our algorithm in Section 5.2.

Algorithm 1 The multiplicative weights algorithm

```
1: Fix $\eta \le 1/2$ and set $\boldsymbol{w}^{(1)} \leftarrow (1, 1, \dots, 1)$.
2: for $t = 1, 2, \dots, T$ do
3:     Choose a strategy $i \in S$ with probability $p_i^{(t)} \leftarrow \boldsymbol{w}_i^{(t)}/\|\boldsymbol{w}^{(t)}\|_1$.
4:     Observe the costs of the strategies $c_1^{(t)}, \dots, c_s^{(t)}$.
5:     For every strategy $i \in S$, set $w_i^{(t+1)} \leftarrow w_i^{(t)}(1-\eta c_i^{(t)})$.
6: end for
```

5.1 Multiplicative weights algorithm

Consider the following setting. We have a set S of s strategies. and we are required to select one strategy from the set in each round. More specifically, in round t, we select a vector $\boldsymbol{p}^{(t)}$ in the s-dimensional simplex Δ_s. Then we sample a strategy $i \in S$ from the distribution determined by $\boldsymbol{p}^{(t)}$, that is, we sample $i \in S$ with probability $p_i^{(t)}$. Each strategy incurs a certain cost, determined by nature or an adversary. After devising our strategy, all the costs are revealed in the form of the vector $\boldsymbol{c}^{(t)} \in \mathbb{R}^s$. The expected cost to the algorithm using the vector $\boldsymbol{p}^{(t)}$ is $\langle\boldsymbol{p}^{(t)}, \boldsymbol{c}^{(t)}\rangle$. Hence, after T rounds, the total expected cost is $\sum_{t=1}^T \langle\boldsymbol{p}^{(t)}, \boldsymbol{c}^{(t)}\rangle$.

We wish to obtain an algorithm that achieves a total expected cost not too much more than the cost of the best single strategy in hindsight, that is, $\min_{i\in S}\sum_{t=1}^T c_i^{(t)}$. Algorithm 1, called the *multiplicative weights algorithm* (the MW algorithm for short), is known to have this property. More specifically, we obtain the following:

Theorem 5.2 ([3])**.** *Assume that all costs $c_i^{(t)} \in [-1, 1]$. By choosing $T = O(\frac{\log s}{\epsilon^2})$ and $\eta = \sqrt{\frac{\log s}{T}}$, the MW algorithm guarantees that, after T rounds, for any strategy $i \in S$, we have*

$$\frac{1}{T}\sum_{t=1}^T \langle\boldsymbol{c}^t, \boldsymbol{p}^t\rangle \le \frac{1}{T}\sum_{t=1}^T c_i^t + \epsilon.$$

5.2 Our algorithm

Now, we turn to portfolio optimization for influence spread. In what follows, we fix a digraph $G = (V, E)$ with edge probabilities $\{p_e\}_{e\in E}$, $k \in \mathbb{N}$, and $\alpha, \epsilon, \delta \in (0, 1)$. Let n and m be the number of vertices and edges in G, respectively.

As it is difficult to optimize CVaR under $\mathcal{D}_{\boldsymbol{X}}$ directly, we consider the empirical distribution $\hat{\mathcal{D}}_{\boldsymbol{X},s}$ of $\mathcal{D}_{\boldsymbol{X}}$, where $s \in \mathbb{N}$ is determined later. Then, we optimize $\mathrm{CVaR}_\alpha(\langle\boldsymbol{\pi}, \boldsymbol{Y}\rangle)$, where $\boldsymbol{Y}$ is a vector of random variables sampled from $\hat{\mathcal{D}}_{\boldsymbol{X},s}$. Note that the size of $\boldsymbol{X}$ is huge, that is, $\binom{n}{k}$, and we cannot afford to explicitly sample $\boldsymbol{X}^1, \dots, \boldsymbol{X}^s$ from $\mathcal{D}_{\boldsymbol{X}}$. However, it is known that the distribution of $R_G(S)$ coincides with the distribution of the number of vertices reachable from S in the random digraph generated from G by keeping each edge e with probability p_e [15]. Hence, we can implicitly construct $\hat{\mathcal{D}}_{\boldsymbol{X},s}$ by generating digraphs $G^1, \dots, G^s$ from G, and set $X_S^i = R_{G^i}(S)/n$, where $R_{G^i}(S)$ is the number of reachable vertices from S in G^i.

We optimize CVaR under $\hat{\mathcal{D}}_{\boldsymbol{X},s}$. Recalling (1), we consider the following problem:

$$\begin{aligned} &\text{maximize} && \tau - \frac{1}{\alpha s}\sum_{i\in[s]} \max\left\{\tau - \langle \boldsymbol{\pi}, \boldsymbol{X}^i\rangle, 0\right\} \\ &\text{subject to} && \tau \in [0,1] \\ &&& \boldsymbol{\pi} \in \Delta_{V,k}. \end{aligned}$$

We can add the extra constraint $\tau \in [0,1]$ without loss of generality because $X_S^i \in [0,1]$ for every $i \in [s]$ and $S \in \binom{V}{k}$. This problem can be rephrased as the following linear programming:

$$\begin{aligned} &\text{maximize} && \tau - \frac{1}{\alpha s}\sum_{i\in[s]} y_i \\ &\text{subject to} && y_i \geq \tau - \langle \boldsymbol{\pi}, \boldsymbol{X}^i\rangle \ \forall i \in [s] \\ &&& \tau \in [0,1] \\ &&& \boldsymbol{y} \in [0,1]^s \\ &&& \boldsymbol{\pi} \in \Delta_{V,k}. \end{aligned} \tag{2}$$

We can solve (2) by binary search on the objective value and by solving the obtained feasibility problem. In order to simplify the feasibility problem, we introduce several definitions. For $\gamma \in \mathbb{R}$, we define a set

$$\mathcal{P}_\gamma = \Big\{(\tau, \boldsymbol{y}, \boldsymbol{\pi}) : \tau \in [0,1], \boldsymbol{y} \in [0,1]^s, \boldsymbol{\pi} \in \Delta_{V,k}, \tau - \frac{1}{\alpha s}\sum_{i\in[s]} y_i \geq \gamma\Big\}.$$

Given a tuple $(\tau, \boldsymbol{y}, \boldsymbol{\pi})$, we define a vector $\boldsymbol{v} := \boldsymbol{v}(\tau, \boldsymbol{y}, \boldsymbol{\pi})$, where $v_i = y_i + \langle \boldsymbol{\pi}, \boldsymbol{X}^i\rangle - \tau$ for each $i \in [s]$. Then, the feasibility version of (2) can be written as follows.

$$\boldsymbol{v} \geq \boldsymbol{0} \qquad (\tau, \boldsymbol{y}, \boldsymbol{\pi}) \in \mathcal{P}_\gamma. \tag{3}$$

We cannot solve (3) directly because we have exponentially many variables. Instead, we show an algorithm that approximately solves (3) by using the MW algorithm.

In the MW algorithm, given a vector $\boldsymbol{p} \in \Delta_s$, we want to check the feasibility of the following problem.

$$\langle \boldsymbol{p}, \boldsymbol{v}\rangle \geq 0 \qquad (\tau, \boldsymbol{y}, \boldsymbol{\pi}) \in \mathcal{P}_\gamma. \tag{4}$$

Note that the first constraint in the abovementioned problem is a convex combination of the original constraints. To solve (4) in the framework of the MW algorithm, we require the following notion:

Definition 5.3 (Oracle). *For $\rho \in (0,1)$, a ρ-oracle for (4) is an algorithm that acts as follows: Given a vector $\boldsymbol{p} \in \Delta_s$, either it finds a tuple $(\tau, \boldsymbol{y}, \boldsymbol{\pi}) \in \mathcal{P}_\gamma$ such that $\langle \boldsymbol{p}, \boldsymbol{v}\rangle \geq -\rho$, or it declares correctly that (4) is infeasible. Moreover, whenever the oracle returns a tuple $(\tau, \boldsymbol{y}, \boldsymbol{\pi}) \in \mathcal{P}_\gamma$, $|v_i| \leq 1$ holds for every $i \in [s]$.*

Indeed, we can design such an oracle as in the following theorem. Although we defer the proof to Section 5.3, we mention that we obtain $\rho = 1/e$ because we implement the oracle by solving an influence maximization problem, which admits $(1 - 1/e)$-approximation.

Theorem 5.4. *A $1/e$-oracle exists for problem (4) with time complexity $O(knms)$. Moreover, it returns a portfolio $\boldsymbol{\pi}$ consisting of a single vertex set.*

With this oracle, we consider Algorithm 2. In each round t, given a distribution over the constraints $\boldsymbol{p}^{(t)}$, we run the oracle with $\boldsymbol{p}^{(t)}$. We obtain the following guarantee:

Algorithm 2

```
Input: Digraphs $G^1, \ldots, G^s$, $\alpha, \epsilon \in (0,1)$, $\gamma \in [0,1]$.
 1: $T \leftarrow \Theta(\frac{\log s}{\epsilon^2})$, $\eta \leftarrow \sqrt{\frac{\log s}{T}}$, and $\boldsymbol{w}^{(1)} \leftarrow (1, 1, \ldots, 1)$.
 2: for $t = 1, 2, \ldots, T$ do
 3:     $\boldsymbol{p}^{(t)} \leftarrow \boldsymbol{w}^{(t)}/\|\boldsymbol{w}^{(t)}\|_1$.
 4:     Call the $1/e$-oracle for (4) (Theorem 5.4) with $\boldsymbol{p}^{(t)}$.
 5:     if the oracle declares no feasible solution then
 6:         reject.
 7:     end if
 8:     $(\tau^{(t)}, \boldsymbol{y}^{(t)}, \boldsymbol{\pi}^{(t)})$ is the solution returned by the oracle.
 9:     for each $i \in [s]$ do
10:         $\boldsymbol{v}^{(t)} \leftarrow \boldsymbol{v}(\tau^{(t)}, \boldsymbol{y}^{(t)}, \boldsymbol{\pi}^{(t)})$.
11:         $w_i^{(t+1)} \leftarrow w_i^{(t)}(1 - \eta v_i^{(t)})$.
12:     end for
13: end for
14: $\bar{\tau} \leftarrow \frac{1}{T}\sum_{t=1}^T \tau^{(t)}$.
15: $\bar{\boldsymbol{y}} \leftarrow \frac{1}{T}\sum_{t=1}^T \boldsymbol{y}^{(t)}$.
16: $\bar{\boldsymbol{\pi}} \leftarrow \frac{1}{T}\sum_{t=1}^T \boldsymbol{\pi}^{(t)}$.
17: return $(\bar{\tau}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{\pi}})$.
```

Lemma 5.5. *Algorithm 2 either finds a tuple $(\bar{\tau}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{\pi}}) \in \mathcal{P}_\gamma$ such that $\bar{\boldsymbol{v}} := \boldsymbol{v}(\bar{\tau}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{\pi}})$ satisfies $\bar{v}_i \geq -1/e - \epsilon$ for every $i \in [s]$, or correctly concludes that (3) is infeasible. The running time is $O(\frac{knms\log s}{\epsilon^2})$. Moreover, each weight in $\bar{\boldsymbol{\pi}}$ is a multiple of $1/T$.*

Proof. If the oracle declares that there is no $(\tau, \boldsymbol{y}, \boldsymbol{\pi}) \in \mathcal{P}$ such that $\langle \boldsymbol{p}^{(t)}, \boldsymbol{v}^{(t)}\rangle \geq 0$, then we stop because $\boldsymbol{p}^{(t)}$ is proof that the problem (3) is infeasible.

Thus, let us assume that this does not occur, that is, in all rounds t, the oracle manages to find a solution $(\tau^{(t)}, \boldsymbol{y}^{(t)}, \boldsymbol{\pi}^{(t)}) \in \mathcal{P}_\gamma$ such that $\langle \boldsymbol{p}^{(t)}, \boldsymbol{v}^{(t)}\rangle \geq -1/e$. First, note that $(\bar{\tau}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{\pi}}) \in \mathcal{P}_\gamma$ since $\mathcal{P}_\gamma$ is a convex set.

Since the cost vector to the multiplicative weights algorithm is specified as $\boldsymbol{v}^{(t)}$, we conclude that the expected cost in each round is $\langle \boldsymbol{p}^{(t)}, \boldsymbol{v}^{(t)}\rangle \geq -1/e$. Hence, Theorem 5.2 tells us that after T rounds, for any $i \in [s]$, we obtain $\bar{v}_i = \frac{1}{T}\sum_{t=1}^T v_i^{(t)} \geq -1/e - \epsilon$ by setting the hidden constant in T large enough.

The analysis of time complexity is obvious from Theorems 5.2 and 5.4. Each weight of $\bar{\boldsymbol{\pi}}$ is a multiple of $1/T$ because the oracle in Theorem 5.4 returns a portfolio consisting of a single set. □

Now that we can approximately solve feasibility problem (3), we can approximately solve (2) using binary search. See Algorithm 3 for a detailed description of our algorithm.

Corollary 5.6. *Algorithm 3 returns a portfolio $\boldsymbol{\pi} \in \Delta_{V,k}$ such that*

$$\mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}, \boldsymbol{Y}\rangle) \geq \hat{\gamma}^* - 1/e - \epsilon,$$

where $\hat{\gamma}^*$ is the optimal value of (2). The running time is $O(\frac{knms\log s}{\epsilon^2}\log\frac{1}{\epsilon})$. Moreover, each weight of $\boldsymbol{\pi}$ is a multiple of $1/T$.

Proof. We note that $\gamma_h \geq \hat{\gamma}^*$ (because Algorithm 2 always returns a solution when there is a feasible solution to (3)), and $\gamma_h - \gamma_l \leq \epsilon/2$ (by choosing a large enough number of loops). It follows that $\gamma_l \geq \hat{\gamma}^* - \epsilon/2$.

Algorithm 3

```
Input: $G^1, \dots, G^s$, $\alpha, \epsilon \in (0,1)$
 1: $\gamma_l \leftarrow 0$ and $\gamma_h \leftarrow 1$.
 2: for $\Theta(\log \frac{1}{\epsilon})$ times do
 3:     $\gamma \leftarrow (\gamma_l + \gamma_h)/2$.
 4:     Call Algorithm 2 with $G^1, \dots, G^s$, $\alpha$, $\epsilon/2$, and $\gamma$.
 5:     if Algorithm 2 declares no feasible solution then
 6:         $\gamma_h \leftarrow \gamma$.
 7:     else
 8:         $\gamma_l \leftarrow \gamma$.
 9:     end if
10: end for
11: Call Algorithm 2 with $G^1, \dots, G^s$, $\alpha$, $\epsilon/2$, and $\gamma_l$.
12: return the obtained portfolio $\boldsymbol{\pi}$.
```

Algorithm 4

```
Input: A digraph $G = (V, E)$, $\alpha, \epsilon, \delta \in (0,1)$.
 1: Generate digraphs $G^1, \dots, G^s$ from $G$, where $s$ is determined in Theorem 5.1.
 2: Call Algorithm 3 with $G^1, \dots, G^s$, $\alpha$, and $\epsilon/3$.
 3: return obtained portfolio $\boldsymbol{\pi}$.
```

Let $(\tau, \boldsymbol{y}, \boldsymbol{\pi})$ be the solution obtained at Line 11 and let $\boldsymbol{v} := \boldsymbol{v}(\tau, \boldsymbol{y}, \boldsymbol{\pi})$. Then, we obtain $v_i \geq -1/e - \epsilon/2$ for any $i \in [s]$ from Lemma 5.5. This means that $(\tau - 1/e - \epsilon/2, \boldsymbol{y}, \boldsymbol{\pi})$ is a feasible solution to (2) with an objective value of at most $\gamma_l - 1/e - \epsilon/2 \geq \hat{\gamma}^* - 1/e - \epsilon$.

The time complexity and requirement for $\boldsymbol{\pi}$ are obvious from Lemma 5.5. □

The overall algorithm (Algorithm 4) simply generates a small number of digraphs from G and then invoke Algorithm 3. We now analyze Algorithm 4 and prove Theorem 5.1.

Proof of Theorem 5.1. We obtain $\mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}, \boldsymbol{Y} \rangle) \geq \hat{\gamma}^* - 1/e - \epsilon/3$ by Corollary 5.6.

Let $L \in \Delta_{V,k}$ be the set of all vectors such that each coordinate is a multiple of $1/T$. Note that Algorithm 3 returns a portfolio only in L. We invoke Lemma 4.1 with $\epsilon/3$ and $\delta/(|L|+1)$. Since $|L| \leq \binom{\binom{n}{k}+T-1}{T}$ holds, the required number of digraphs is

$$s = \Omega\Big(\frac{1}{\epsilon^2} \log \frac{|L|+1}{\delta}\Big) = \Omega\Big(\frac{1}{\epsilon^2}\big(T \log \frac{\binom{n}{k}+T-1}{T} + \log \frac{1}{\delta}\big)\Big)$$
$$= \Omega\Big(\frac{1}{\epsilon^2}\big(\frac{k \log s}{\epsilon^2} \log \frac{n}{k} + \log \frac{1}{\delta}\big)\Big).$$

By the union bound, we obtain

$$\Big|\mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}', \boldsymbol{X} \rangle) - \mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}', \boldsymbol{Y} \rangle)\Big| \leq \frac{\epsilon}{3} \quad \text{and}$$
$$\Big|\mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}^*, \boldsymbol{X} \rangle) - \mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}^*, \boldsymbol{Y} \rangle)\Big| \leq \frac{\epsilon}{3}$$

for every $\boldsymbol{\pi}' \in L$ and for the optimal portfolio $\boldsymbol{\pi}^* \in \Delta_{V,k}$ with probability at least $1 - \delta$.

Algorithm 5

```
Input: Graphs $G^1, \dots, G^s$, $\alpha \in (0,1)$, and $\boldsymbol{p} \in \mathbb{R}^s_+$.
 1: Assume $p_1 \geq p_2 \geq \cdots \geq p_s$.
 2: for $i = 1$ to $s$ do
 3:     if $p_i \geq \frac{1}{\alpha s}$ then
 4:         $y_i \leftarrow \min\{\alpha s - \sum_{j \in [i-1]} y_j, 1\}$.
 5:     else
 6:         $y_i \leftarrow 0$.
 7:     end if
 8: end for
 9: $\tau \leftarrow 1 - \frac{1}{\alpha s} \sum_{i \in [s]} y_i$.
10: Compute $\tilde{S} \in \binom{V}{k}$ by applying the greedy algorithm to $f_{\boldsymbol{p}}(\cdot)$.
11: $\pi_{\tilde{S}} \leftarrow 1$ and $\pi_S \leftarrow 0$ for every $S \neq \tilde{S}$.
12: if $\langle \boldsymbol{p}, \boldsymbol{v} \rangle \geq -1/e$ then
13:     return $(\tau, \boldsymbol{y}, \boldsymbol{\pi})$.
14: else
15:     Reject.
16: end if
```

Since $\boldsymbol{\pi} \in L$, we obtain

$$\begin{aligned}
&\mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}, \boldsymbol{X} \rangle) \geq \mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}, \boldsymbol{Y} \rangle) - \epsilon/3 \\
&\geq \hat{\gamma}^* - 1/e - 2\epsilon/3 \\
&\geq \mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}^*, \boldsymbol{Y} \rangle) - 1/e - 2\epsilon/3 \\
&\geq \mathrm{CVaR}_\alpha(\langle \boldsymbol{\pi}^*, \boldsymbol{X} \rangle) - 1/e - \epsilon \\
&= \gamma^* - 1/e - \epsilon
\end{aligned}$$

with probability at least $1 - \delta$.

The analysis of the time complexity is obvious. □

5.3 Implementation of the Oracle

In this section, we discuss how to implement an oracle for the feasibility problem (4) and prove Theorem 5.4.

For a set $S \subseteq \binom{V}{k}$, let $\boldsymbol{\pi}^S \in \Delta_{V,k}$ be the vector with $\pi^S_S = 1$ and $\pi^S_{S'} = 0$ for every $S' \neq S$. We define $f_{\boldsymbol{p}} : 2^V \to \mathbb{R}$ as $f_{\boldsymbol{p}}(S) = \sum_{i \in [s]} p_i X^i_S$. Then, the objective of the oracle is maximizing the following value

$$\begin{aligned}
\langle \boldsymbol{p}, \boldsymbol{v} \rangle &= \sum_{i \in [s]} p_i (y_i - \tau) + \sum_{S \in \binom{V}{k}} \pi_S f_{\boldsymbol{p}}(S) \\
&= \sum_{i \in [s]} p_i y_i + \sum_{S \in \binom{V}{k}} \pi_S f_{\boldsymbol{p}}(S) - \tau
\end{aligned} \tag{5}$$

subject to $\boldsymbol{v} \in \mathcal{P}_\gamma$.

Let S^* be a maximizer of the problem $\max_{S \subseteq \binom{V}{k}} f_{\boldsymbol{p}}(S)$. Then, we obtain the following:

Lemma 5.7. *If $(\tau, \boldsymbol{y}, \boldsymbol{\pi})$ is a feasible solution to (4), then we can assume that $\boldsymbol{\pi} = \boldsymbol{\pi}^{S^*}$.*

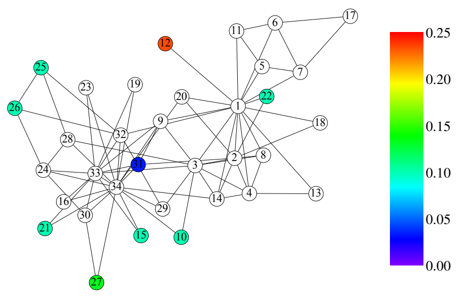
Proof. This is clear from (5); otherwise, we can increase the objective value by decreasing π_S for $S \neq S^*$ with $\pi_S > 0$ and by increasing π_{S^*}. □

Although it is NP-hard to compute S^*, since $f_{\boldsymbol{p}}$ is submodular, we can compute $\tilde{S}$ of size k such that $f_{\boldsymbol{p}}(\tilde{S}) \geq (1 - 1/e) f_{\boldsymbol{p}}(S^*)$.

Lemma 5.8. *Assume $p_1 \geq p_2 \geq \cdots \geq p_s$. If $(\tau, \boldsymbol{y}, \boldsymbol{\pi})$ is a feasible solution to (4), then we can assume the following:*

Table 3: Number of positive weights of portfolios obtained by our method ($\alpha = 0.01$).

Seed size k	1	5	10	15
Karate dataset	9	31	36	30
Physicians dataset	10	39	42	45
Advogato dataset	5	36	44	51



v	π_v
12	0.233
27	0.133
10	0.100
15	0.100
21	0.100
22	0.100
25	0.100
26	0.100
31	0.033

Figure 3: Visualization of Karate network. Each vertex is colored according to its portfolio weight ($\alpha = 0.01$).

- *For every $i \in [s]$, if $p_i \geq \frac{1}{\alpha s}$, then we have*
$$y_i = \min\Big\{\alpha s - \sum_{j \in [i-1]} y_j, 1\Big\},$$
and otherwise we have $y_i = 0$.
- $\tau = 1 - \frac{1}{\alpha s}\sum_{i \in [s]} y_i$.

Proof. After fixing $\boldsymbol{\pi}$, we want to maximize $\sum_{i \in [s]} p_i y_i - \tau$ subject to $\tau - \frac{1}{\alpha s}\sum_{i \in [s]} y_i \geq \gamma$.

If there exists $y_i < 1$ and $y_j > 0$ with $i < j$, then we can increase the objective value by increasing y_i and decreasing y_j.

If there exists $y_i > 0$ with $p_i < \frac{1}{\alpha s}$, we can increase the objective value decreasing y_i and increasing τ.

From these observations, we get the claimed solution. $\square$

These lemmas motivate us to consider Algorithm 5.

Proof of Theorem 5.4. Suppose that Algorithm 5 returned a tuple $(\tau, \boldsymbol{y}, \boldsymbol{\pi})$. Then, we clearly have $\langle \boldsymbol{p}, \boldsymbol{v} \rangle \geq -1/e$. In addition, we have $|v_i| \leq 1$.

Suppose that Algorithm 5 has rejected. Then,

$$\langle \boldsymbol{p}, \boldsymbol{v}^* \rangle = \sum_{i \in [s]} p_i y_i + f_{\boldsymbol{p}}(S^*) - \tau \leq \sum_{i \in [s]} p_i y_i + \frac{e}{e-1} f_{\boldsymbol{p}}(\tilde{S}) - \tau$$
$$\leq \frac{e}{e-1}\Big(\sum_{i \in [s]} p_i y_i + f_{\boldsymbol{p}}(\tilde{S}) - \tau\Big) + \frac{1}{e-1} \qquad (\text{As } \tau \leq 1)$$
$$< \frac{e}{e-1}\Big(-\frac{1}{e}\Big) + \frac{1}{e-1} = 0.$$

Hence, Algorithm 5 is a $1/e$-oracle for (4).

A naive implementation of the greedy method takes $O(knms)$ time and we get the desired time complexity. $\square$

We note that we can exploit a heuristic [20] to make the greedy method run much faster, preserving the approximation factor.

6. EXPERIMENTAL EVALUATIONS

In this section, we demonstrate the effectiveness of the proposed algorithm by experiment. We conducted experiments on a Linux server with an Intel Xeon E5-2670 2.60 GHz CPU and 512 GB memory. All algorithms were implemented in C++ and compiled using g++ 4.8.2 with the -O2 option.

6.1 Settings

We used three publicly available real-world network datasets, Karate with 34 vertices and 78 bidirectional edges, Physicians with 117 vertices and 542 directed edges, and Advogato with 5,042 vertices and 78,454 directed edges from the Koblenz Network Collection [16]. We extracted the subgraphs induced by the largest connected components. For each edge uv, we assigned its edge probability the inverse of the out-degree of u. The significance level α was set to be 0.01 and 0.05.

For our method, we set $\epsilon = 0.4$ and assigned $s = \frac{k \log n}{\epsilon^4}, T = \frac{\log s}{\epsilon^2}, \eta = \sqrt{\frac{\log s}{T}}$, and the binary search (for the loop in Algorithm 3) was repeated 32 times. We compared our method with the following three baseline algorithms that output only a single seed set:

- *Greedy* [15], which is a standard greedy algorithm for influence maximization. We adopted a fast implementation [20], which has the same accuracy guarantee as in [15].
- *Degree*, which selects k vertices in the decreasing order of degrees.
- *Random*, which selects k vertices uniformly at random.

6.2 Results

First, we verify the effectiveness of our proposed method. Table 1 reports the CVaR at $\alpha = 0.01$ and $\alpha = 0.05$ of the portfolio obtained by each method. We conducted Monte Carlo simulations of influence spread 10,000 times to obtain an estimation of the CVaR for each portfolio. The proposed method significantly outperformed existing methods for all settings. The difference is especially large when k is small, but even when k is large, e.g., the CVaRs at $\alpha = 0.01$ of our portfolios on Physicians and Advogato for $k = 15$ are 17.5% and 67.6% better than the second best, respectively.

Table 2 shows the expected cascade sizes. Although our method does not explicitly optimize the expected cascade size, we can observe that those of the portfolios obtained by our method are comparable to those of seed sets obtained by the greedy method.

Figure 2 shows the histogram of the cascade size for each method. The histograms of baseline algorithms spread out, which easily result in extremely smaller cascades than the average. On the other hand, the histograms of the portfolios computed by our method are promising; they are well concentrated on the mean value, which are more desirable in terms of risk aversion.

Table 3 shows the number of positive weights of each portfolio obtained by our method. The number of positive weights is at most 100, which are reasonably small.

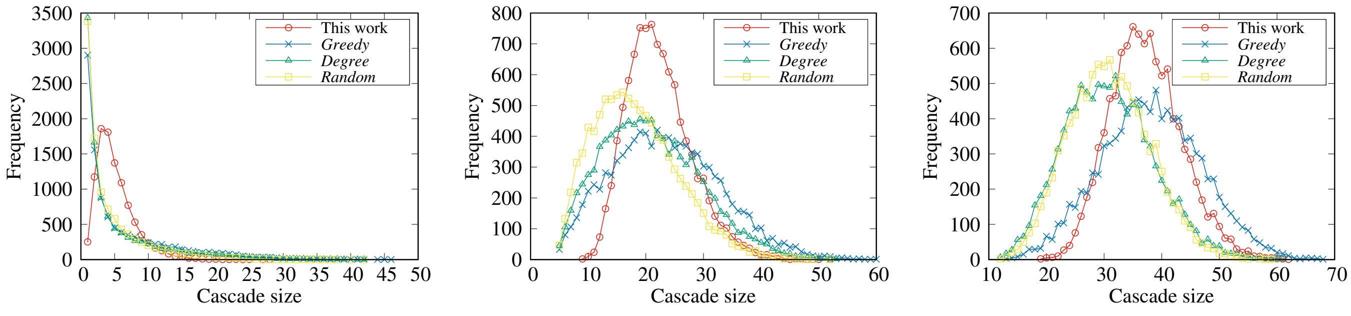
Finally, we show the structure of the portfolio obtained by our method. Figure 3 illustrates the weights in the portfolio obtained by applying our method to Karate network with $k = 1$ and $\alpha = 0.01$. In this network, there are two

Table 1: CVaR for portfolios obtained by each method. Best results are in bold.

$\alpha = 0.01$	Karate dataset				Physicians dataset				Advogato dataset			
Seed size k	1	5	10	15	1	5	10	15	1	5	10	15
This work	1.7	**8.7**	**15.6**	**21.3**	**1.5**	**12.0**	**23.6**	**34.2**	**1.2**	**18.7**	**42.3**	**69.9**
Greedy	**2.0**	6.8	14.5	20.3	1.0	5.7	16.9	29.1	1.0	7.5	21.5	41.7
Degree	1.0	5.0	10.2	15.4	1.0	5.5	14.2	24.1	1.0	5.3	14.7	26.8
Random	1.0	5.5	12.8	17.6	1.0	5.5	15.1	22.8	1.0	6.5	17.8	34.4
$\alpha = 0.05$	Karate dataset				Physicians dataset				Advogato dataset			
Seed size k	1	5	10	15	1	5	10	15	1	5	10	15
This work	1.9	**9.6**	**16.8**	**22.4**	**1.9**	**13.7**	**26.3**	**37.2**	**1.4**	**22.3**	**48.9**	**78.7**
Greedy	**2.0**	8.1	15.8	21.6	1.0	7.6	20.9	33.4	1.0	9.0	27.1	50.7
Degree	1.0	5.8	10.8	16.1	1.0	7.0	16.9	27.3	1.0	6.5	18.8	34.4
Random	1.0	6.6	13.9	18.6	1.0	6.8	17.8	25.7	1.0	7.9	23.1	43.4

Table 2: Mean value for portfolios obtained by each method. Best results are in bold.

	Karate dataset				Physicians dataset				Advogato dataset			
Seed size k	1	5	10	15	1	5	10	15	1	5	10	15
This work ($\alpha = 0.01$)	3.9	14.0	21.0	26.1	5.6	22.8	37.7	49.0	**11.3**	**50.8**	**94.0**	**131.6**
Greedy	**4.3**	**14.3**	**21.2**	**26.2**	**6.2**	**23.6**	**38.2**	**49.9**	**11.3**	50.7	92.9	130.3
Degree	3.3	10.0	14.8	19.7	5.2	21.0	30.8	40.7	9.9	40.4	68.9	93.6
Random	3.8	12.6	18.9	22.6	4.5	18.3	31.0	38.2	9.1	45.6	86.3	119.6



(a) Seed size $k = 1$ (b) Seed size $k = 5$ (c) Seed size $k = 10$

Figure 2: Histogram of cascade sizes for portfolios constructed by each method on Physicians ($\alpha = 0.01$).

overlapping communities centered at vertices 1 and 34. Our method assigns positive weights to vertices in both communities, for example, vertices 12 and 27, which are adjacent to vertex 1 and 34, respectively. Note that *Greedy*, *Degree*, and *Random* selected vertices 12, 34, and 19 as a seed vertices, respectively.

From the abovementioned results, we conclude that the proposed method effectively resolves the issue concerning the risk of having a small influence spread.

7. CONCLUSIONS

In this study, we considered portfolio optimization over seed sets so that the CVaR of influence spread is maximized. Our polynomial-time algorithm using multiplicative weights (MW) algorithm computed a portfolio whose CVaR is at least the optimum minus $1/e+\epsilon$. The MW algorithm reduces the problem to a sequence of monotone submodular function maximization problems, and we (approximately) solved it by a greedy algorithm. Our experimental results showed that portfolios obtained by our algorithm indeed have much larger CVaRs than those obtained by other baseline methods.

Improving the time complexity is an important future work. For example, when solving monotone submodular function maximization problems, it might be possible to use recent speeding-up techniques for the influence maximization problem [6, 30, 29]. Also in the proof of Theorem 5.1, it might be possible to apply Talagrand's chaining method [26] instead of a simple union bound. Another interesting direction is designing portfolio optimization algorithms for problems involving other stochastic processes on networks such as random walk.

Acknowledgments

N. O. is supported by JSPS Grant-in-Aid for JSPS Research Fellow (No. 16J09440). Y. Y. is supported by JSPS Grant-in-Aid for Young Scientists (B) (No. 26730009), MEXT Grant-in-Aid for Scientific Research on Innovative Areas (No. 24106003), and JST, ERATO, Kawarabayashi Large Graph Project.

8. REFERENCES

[1] C. Acerbi and D. Tasche. On the coherence of expected shortfall. *Journal of Banking & Finance*, 26(7):1487–1503, 2002.

[2] V. Agarwal and N. Y. Naik. Risks and portfolio decisions involving hedge funds. *Review of Financial studies*, 17(1):63–98, 2004.

[3] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[4] P. Artzner, F. Delbaen, J. M. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

[5] N. Bäuerle and J. Ott. Markov decision processes with average-value-at-risk criteria. *Mathematical Methods of Operations Research*, 74(3):361–379, 2011.

[6] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.

[7] J. Calatrava and A. Garrido. Spot water markets and risk in water supply. *Agricultural Economics*, 33(2):131–143, 2005.

[8] Y. Chow and M. Ghavamzadeh. Algorithms for CVaR optimization in mdps. In *NIPS*, pages 3509–3517, 2014.

[9] Y. Chow, A. Tamar, S. Mannor, and M. Pavone. Risk-sensitive and robust decision-making: a CVaR optimization approach. In *NIPS*, pages 1522–1530, 2015.

[10] D. Deng, H. Du, X. Jia, and Q. Ye. Minimum-cost information dissemination in social networks. In *WASA*, pages 83–93, 2015.

[11] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

[12] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211–223, 2001.

[13] W. B. Haskell and R. Jain. A convex analytic approach to risk-aware markov decision processes. *SIAM Journal on Control and Optimization*, 53(3):1569–1598, 2015.

[14] L. J. Hong and G. Liu. Simulating sensitivities of conditional value at risk. *Management Science*, 55(2):281–293, 2009.

[15] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11:105–147, 2015.

[16] J. Kunegis. Konect - the koblenz network collection. In *WWW Companion*, pages 1343–1350, 2013.

[17] T. Maehara. Risk averse submodular utility maximization. *Operations Research Letters*, 43(5):526–529, 2015.

[18] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77, 1952.

[19] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[20] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *AAAI*, pages 138–144, 2014.

[21] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70, 2002.

[22] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

[23] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443–1471, 2002.

[24] O. Scaillet. Nonparametric estimation and sensitivity analysis of expected shortfall. *Mathematical Finance*, 14(1):115–129, 2004.

[25] G. Serraino and S. Uryasev. Conditional value-at-risk (CVaR). In *Encyclopedia of Operations Research and Management Science*, pages 258–266. Springer US, Boston, MA, 2013.

[26] M. Talagrand. Majorizing measures: The generic chaining. *The Annals of Probability*, 24:1049–1103, 1996.

[27] A. Tamar, Y. Chow, M. Ghavamzadeh, and S. Mannor. Policy gradient for coherent risk measures. In *NIPS*, pages 1468–1476, 2015.

[28] A. Tamar, Y. Glassner, and S. Mannor. Optimizing the CVaR via sampling. In *AAAI*, pages 2993–2999, 2015.

[29] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, pages 1539–1554, 2015.

[30] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*, pages 75–86, 2014.

[31] P. Zhang, W. Chen, X. Sun, Y. Wang, and J. Zhang. Minimizing seed set selection with probabilistic coverage guarantee in a social network. In *KDD*, pages 1306–1315, 2014.