Towards Distributed Two-Stage Stochastic Optimization

Yuval Emek
Technion - Israel Institute of Technology, Haifa, Israel
yemek@technion.ac.il

Noga Harlev
Technion - Israel Institute of Technology, Haifa, Israel
snogazur@campus.technion.ac.il

Taisuke Izumi
Nagoya Institute of Technology, Japan
t-izumi@nitech.ac.jp

Abstract

The *weighted vertex cover* problem is concerned with selecting a subset of the vertices that covers a target set of edges with the objective of minimizing the total cost of the selected vertices. We consider a variant of this classic combinatorial optimization problem where the target edge set is not fully known; rather, it is characterized by a probability distribution. Adhering to the model of *two-stage stochastic optimization*, the execution is divided into two stages so that in the first stage, the decision maker selects some of the vertices based on the probabilistic forecast of the target edge set. Then, in the second stage, the edges in the target set are revealed and in order to cover them, the decision maker can augment the vertex subset selected in the first stage with additional vertices. However, in the second stage, the vertex cost increases by some inflation factor, so the second stage selection becomes more expensive.

The current paper studies the two-stage stochastic vertex cover problem in the realm of *distributed graph algorithms*, where the decision making process (in both stages) is distributed among the vertices of the graph. By combining the stochastic optimization toolbox with recent advances in distributed algorithms for weighted vertex cover, we develop an algorithm that runs in time $O(\log(\Delta)/\varepsilon)$, sends $O(m)$ messages in total, and guarantees to approximate the optimal solution within a $(3+\varepsilon)$-ratio, where m is the number of edges in the graph, Δ is its maximum degree, and $0 < \varepsilon < 1$ is a performance parameter.

2012 ACM Subject Classification Theory of computation → Distributed computing models

Keywords and phrases weighted vertex cover, distributed graph algorithms, two-stage stochastic optimization, primal-dual

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2019.32

Funding *Yuval Emek*: This work has been funded in part by an Israeli Ministry of Science and Technology grant number 3-13565.
Taisuke Izumi: This work was supported by JST SICORP Grant Number JPMJSC1606 and JSPS KAKENHI Grant Number JP19K11824, Japan.

1 Introduction

Distributed computing models are adversarial in nature: if the future bears any kind of uncertainty, then we better prepare for the worst. While this approach is sensible in some situations, e.g., when dealing with faults in critical systems, it seems to be exaggerated in others: Should we still aim for the worst even if we hold reliable forecasts that point to more optimistic outcomes? Is it not a little bit paranoid to assume that the future is always determined by a malicious adversary?

With this motivation in mind, the current paper adopts the *two-stage stochastic optimization* model, where the future uncertainty is resolved by nature's coin tosses rather than by a malicious adversary. This popular model is adjusted to the realm of *distributed graph algorithms* and applied to the *weighted vertex cover* problem. The adaptation of weighted vertex cover to the distributed stochastic setting makes it applicable to many real-life scenarios, especially in the context of network applications. For example, a distributed construction of a minimum vertex cover could come in handy for tasks such as monitoring traffic over high demand links. More often than not, the set of high demand links is not fully known in advance and decisions regarding the placement of monitors must be made based on probabilistic forecasts.

1.1 Distributed Two-Stage Stochastic Vertex Cover

The problem considered in this paper is a distributed version of the *two-stage stochastic vertex cover (2SVC)* problem. In its centralized version, the 2SVC problem is defined over an undirected graph $G = (V, E)$ and a vertex *cost* function $c : V \to \mathbb{R}_{\geq 0}$. Given a *target* edge subset $T \subseteq E$, the goal is to purchase a vertex subset $X \subseteq V$ that *covers* T in the sense that every edge in T has at least one endpoint in X.

The crux of the 2SVC problem is that the target T is not known explicitly; rather, it is drawn from a probability distribution π over 2^E. A 2SVC algorithm `Alg` constructs the covering vertex subset X in two *stages*: In the first stage, `Alg` constructs a vertex subset $X_1 \subseteq V$ based on π, but without knowing the realized target T. Then, in the second stage, T is revealed and `Alg` augments X_1 with a vertex subset $X_2 \subseteq V - X_1$ to obtain a valid cover $X = X_1 \cup X_2$ for T, but the vertices are now costlier: each vertex $v \in X_2$ costs $\sigma \cdot c(v)$, where $\sigma > 1$ is an *inflation factor* specified with the problem instance (together with G, c, and π). The objective is to minimize `Alg`'s total cost $c(X_1) + \sigma \cdot c(X_2)$ in expectation, where $c(V') = \sum_{v \in V'} c(v)$ for every $V' \subseteq V$ and the expectation is taken over the probability distribution π from which the target T is drawn and the random coin tosses of the algorithm (if any).

In the current paper, we adapt the 2SVC problem to the distributed setting where there is no centralized decision maker. Adhering to the standard assumptions in the domain of distributed graph algorithms (cf. [32]), the computation in both the first and second stages is distributed among the vertices in V that are identified with processing units that operate in synchronous *rounds* and can communicate with each other by exchanging messages of bounded size over the edges in E (referred to as the *CONGEST* model in [32]). This means that at the beginning of the first stage, the vertices hold no knowledge of the global topology of G. Moreover, we assume that π is a product distribution, so that $\pi(T) = (\prod_{e \in T} \pi(e)) \cdot (\prod_{e \in E - T}(1 - \pi(e)))$, and that it is provided to the vertices in a distributed manner: at the beginning of the first stage, vertex $v \in V$ is aware of the individual edge probability $\pi(e)$ if and only if e is incident on v.

The *approximation ratio* of a distributed 2SVC algorithm `Alg` is the minimum ρ such that for every graph $G = (V, E)$, cost function $c : V \to \mathbb{R}_{\geq 0}$, target (product) distribution π over 2^E, and inflation factor $\sigma > 1$, it is guaranteed that the expected cost paid by `Alg` is at most

$$\rho \times \min_{X_1 \subseteq V} \left(c(X_1) + \sum_{T \subseteq E} \pi(T) \cdot \min_{X_2 \subseteq V \,:\, X_1 \cup X_2 \text{ covers } T} \sigma \cdot c(X_2) \right) .$$

In other words, the performance of `Alg` is measured in comparison to an optimal (omnipotent) centralized benchmark.

1.2 Our Contribution

In this paper, we design a distributed algorithm, referred to as `Alg`, that for any $0 < \varepsilon < 1$, outputs a $(3 + \varepsilon)$-approximation for the 2SVC problem. Denoting the number of vertices in G by n and its maximum degree by Δ and assuming that the vertex costs $c(v)$ and performance parameter ε can be encoded using $O(\log n)$ bits, `Alg` completes the first stage in $O(\log(\Delta)/\varepsilon)$ rounds and the second stage in $O(1)$ rounds, sending a total of $O(1)$ messages per edge, each of size $O(\log n)$. We assume for simplicity that the vertices are equipped with unique IDs (encoded using $O(\log n)$ bits), but these can be easily replaced by any local symmetry breaking oracle (e.g., an arbitrary edge orientation). Beyond that, the vertices are not assumed to hold any global knowledge of the graph, including the parameters n and Δ.

Two-stage (and more generally, multi-stage) stochastic optimization is an important research domain (see Sec. 1.3) that, to the best of our knowledge, has not been studied yet in the context of distributed graph algorithms. We hope that the first step that the current paper makes into the interface between these fascinating research domains will ignite further exploration of distributed algorithms for stochastic graph theoretic problems.

1.3 Related Work

The field of stochastic optimization dates back to the mid-fifties with the seminal papers of Dantzig [12] and Beale [8] on stochastic linear programming and is studied extensively since then. Comprehensive accounts of stochastic programming under both continuous and discrete models can be found in [35, 27, 10].

Initiated in the paper of Dye, Stougie, and Tomasgard [14], the study of approximation algorithms for stochastic optimization problems has recently gained a lot of attention [21, 13, 34, 15, 19, 3]. Much of this literature deals with the *finite scenario* model, where the scenarios that may occur in the second stage are listed explicitly as part of the input. A different approach, sometimes called the *black-box* model, relies on sampling access to a probability distribution that is not necessarily provided explicitly. Shmoys and Swamy obtained approximation algorithms in this model for problems in the framework of two-stage [36] and multi-stage [38] stochastic optimization. They also wrote a broad survey on approximation algorithms for a large class of two-stage stochastic linear and integer programs in both the finite scenario and black-box models [39].

Among other methods, Shmoys and Swamy [38] used the *sample average approximation* approach that reduces many problems in the black-box model to their finite scenario counterpart. This approach essentially feeds the finite scenario algorithm with samples taken from the (implicit) probability distribution. We remark that although this approach has proven to be very successful for centralized algorithms [36, 38, 37], it seems to be less suitable for distributed settings.

As discussed later on, our work builds upon the (centralized) framework of Gupta et al. [18] for general stochastic optimization problems. This is a generic framework that given an approximation algorithm with certain desired properties for (the deterministic version) of a combinatorial optimization problem $\mathcal{P}$, yields an approximation algorithm for the two-stage (and multi-stage) stochastic versions of $\mathcal{P}$. Among other combinatorial optimization problems, Gupta et al. applied their framework to weighted vertex cover, obtaining a (centralized) 3-approximation 2SVC algorithm. This has been improved to a $(2 + \varepsilon)$-approximation by Srinivasan [37] using the sample average approximation approach.

In its deterministic version, vertex cover is a classic combinatorial optimization problem, listed among the 21 NP-hard problems in Karp's seminal paper [24]. The unweighted version of this problem can be approximated within a factor of 2 simply by finding a maximal matching (see, e.g., [11]). A 2-approximation polynomial-time algorithm for the weighted version of vertex cover was introduced by Nemhauser and Trotter [30]. Bar-Yehuda and Even [5] presented the first linear-time 2-approximation algorithm using the primal-dual technique and later on [6], repeated this result to demonstrate their local ratio technique. The currently best known approximation for weighted vertex cover problem is $(2 - \Theta(1/\sqrt{\log n}))$ due to Karakostas [23]. Assuming the unique games conjecture, this cannot be improved much further as Khot and Regev [25] established a lower bound of $2 - \varepsilon$ for any constant $\varepsilon > 0$.

In the distributed setting, there is a long line of work on approximation algorithms for unweighted vertex cover and on the related maximal matching problem [22, 20, 31, 1, 33, 7, 16]. For the weighted version of vertex cover, Khuller et al. [26] present a $(2 + \varepsilon)$-approximation algorithm that runs in $O(\log \varepsilon^{-1} \log n)$ rounds. The algorithm of Kuhn et al. [29] obtains the same approximation ratio with an $O(\varepsilon^{-4} \log \Delta)$ run-time. Åstrand and Suomela [2] designed a 2-approximation algorithm whose run-time is $O(\Delta + \log^* W)$, where W is the maximum weight. A randomized 2-approximation algorithm that runs in $O(\log n + \log W)$ was presented by Grandoni et al. [17]. This has been improved to a run-time of $O(\log n)$ by Koufogiannakis and Young [28].

Building upon the classic local ratio technique, Bar-Yehuda et al. [4] have recently devised a deterministic $(2 + \varepsilon)$-approximation algorithm for weighted vertex cover with $O\left(\frac{\log \Delta}{\varepsilon \cdot \log \log \Delta}\right)$ run-time. This algorithm has been improved by Ben-Basat et el. [9] to obtain a run-time whose dependency on $1/\varepsilon$ is logarithmic rather than linear. With the right choice of the performance parameter ε, this leads to a deterministic 2-approximation algorithm for weighted vertex cover whose run-time is $O\left(\frac{\log n \cdot \log \Delta}{\log^2 \log \Delta}\right)$. As discussed further in the sequel, the approach behind the algorithms of [4, 9] inspires the main algorithmic building block of our 2SVC approximation algorithm.

1.4 Organization of the Paper

The remainder of this paper is organized as follows. Following some preliminary definitions presented in Sec. 2, we discuss in Sec. 3 the technical challenges that had to be overcome en route to developing our distributed 2SVC approximation algorithm. The algorithm is then developed in Sec. 4 and analyzed in Sec. 5. We conclude in Sec. 6 with some further discussion and open questions.

2 Preliminaries

Relaxed Vertex Cover. We now define a slightly different version of the 2SVC problem, called *relaxed 2SVC (r2SVC)*. In this version we are allowed to make partial payments for the vertices in each of the two stages based on the notion of a *payment* function $p : V \to \mathbb{R}_{\geq 0}$. We say that a payment function p *covers* the edge subset $F \subseteq E$ if the vertex subset $X^p = \{v \in V \mid p(v) \geq c(v)\}$ is a cover for F.

An algorithm $\texttt{Alg}^{\mathrm{rel}}$ for the r2SVC problem constructs the payment functions $p_1 : V \to \mathbb{R}_{\geq 0}$ and $p_2 : V \to \mathbb{R}_{\geq 0}$ in the first and second stages, respectively, with the requirement that the function $p_1 + p_2$, defined so that $(p_1 + p_2)(v) = p_1(v) + p_2(v)$, covers the target edge subset T. The cost paid by $\texttt{Alg}^{\mathrm{rel}}$ is $c(p_1) + \sigma \cdot c(p_2)$, where the cost $c(p)$ of a payment function $p : V \to \mathbb{R}_{\geq 0}$ is defined to be $c(p) = \sum_{v \in V} p(v)$.

By definition, any 2SVC algorithm can be transformed into a r2SVC algorithm with the same cost by setting $p_i(v) = c(v)$ if $v \in X_i$ and $p_i(v) = 0$ otherwise for $i = 1, 2$. Gupta et al. [18] established the converse direction: any r2SVC algorithm can be transformed into a 2SVC algorithm with the same expected cost. Although their proof aims at centralized algorithms, it is straightforward to see that it holds also for the distributed version of the (r)2SVC problem, yielding the following lemma (a proof is added for completeness).

▶ **Lemma 2.1.** *Any distributed r2SVC algorithm* $\texttt{Alg}^{\text{rel}}$ *can be transformed into a distributed 2SVC algorithm* $\texttt{Alg}$ *with the same expected cost.*

Proof. Let p_1 and p_2 be the payment functions constructed by $\texttt{Alg}^{\text{rel}}$ in the first and second stages, respectively. We construct the 2SVC algorithm $\texttt{Alg}$ as follows. In the first stage $\texttt{Alg}$ includes vertex $v \in V$ in the vertex subset X_1 with probability $\min\{1, p_1(v)/c(v)\}$. In the second stage, $\texttt{Alg}$ includes v in the vertex subset X_2 if $v \in X^{p_1+p_2}$ and it was not already selected in the first stage. By the linearity of expectation, the expected payment made by $\texttt{Alg}$ in each stage is up-bounded by the payment made by $\texttt{Alg}^{\text{rel}}$ in the same stage. ◀

Following Lem. 2.1, we focus hereafter on designing a distributed algorithm for the r2SVC problem.

The Framework of [18]. Our distributed algorithm for the r2SVC problem is based on the (centralized) *boosted-sampling* algorithm of Gupta et al. [18] for that problem.[1] This is a randomized algorithm that is compiled from two algorithmic building blocks. The first building block is a *covering payment* algorithm, referred to as $\texttt{Alg}^{\text{pay}}$, that given an edge subset $F \subseteq E$, constructs a payment function $p : V \to \mathbb{R}_{\geq 0}$ that covers F.[2] We say that $\texttt{Alg}^{\text{pay}}$ is an *α-approximation* covering payment algorithm if the cost of the payment function p constructed by $\texttt{Alg}^{\text{pay}}$ is guaranteed to satisfy $c(p) \leq \alpha \cdot c(\texttt{Opt}(F))$, where $\texttt{Opt}(\text{F})$ is an optimal cover for F.

The second building block of the boosted-sampling algorithm is an *augmentation algorithm*, referred to as $\texttt{Alg}^{\text{aug}}$, that given two edge subsets $F, F' \subseteq E$ and a payment function $p : V \to \mathbb{R}_{\geq 0}$ that covers F, constructs a payment function p' so that $p + p'$ covers F'.

In the first stage, the boosted-sampling algorithm creates a random edge subset S by sampling each edge $e \in E$ with probability $\min\{1, \sigma\pi(e)\}$. Following that, it invokes $\texttt{Alg}^{\text{pay}}$ on S to obtain a payment function p_1 that covers S. In the second stage, when the actual edge subset T is revealed, the boosted-sampling algorithm invokes $\texttt{Alg}^{\text{aug}}$ on (S, T, p_1) to construct the payment function p_2 so that $p_1 + p_2$ covers T.

An essential ingredient in the analysis of the boosted-sampling algorithm is the notion of *cost-sharing* functions. These functions provide a useful way to bound the overall cost by allocating it to the edges that need to be covered. Formally, a function $\xi : 2^E \times E \to \mathbb{R}_{\geq 0}$ is said to be a *cost-sharing* function if it satisfies the following two properties for any edge subset $F \subseteq E$:

P1. $\xi(F, e) = 0$ for any edge $e \in E - F$; and

P2. $\sum_{e \in F} \xi(F, e) \leq c(\texttt{Opt}(F))$.

The cost-sharing function ξ is said to be *β-unistrict* with respect to $\texttt{Alg}^{\text{pay}}$ and $\texttt{Alg}^{\text{aug}}$ if it satisfies the following property for any edge subset $F \subset E$ and edge $e \in E - F$:

P3. $\beta \cdot \xi(F \cup \{e\}, e) \geq c(\texttt{Alg}^{\text{aug}}(F, \{e\}, \texttt{Alg}^{\text{pay}}(F)))$.

[1] As mentioned in Sec. 1.3, Gupta et al. develop a generic framework and the boosted-sampling algorithm is actually suitable for many different combinatorial optimization problems. Since our focus in the current paper is restricted to weighted vertex cover, we present an adaptation of this algorithm to that specific problem.

[2] We assume that the underlying graph $G = (V, E)$ and the vertex cost function c are fixed and do not explicitly pass them as arguments to the various algorithms described hereafter.

$$\begin{array}{llll} \text{(P)} & \min \sum_{v \in V} x_v c(v) & \qquad \text{(D)} & \max \sum_{e \in E} y_e \\ \text{s.t.} & x_u + x_v \geq 1 \quad \forall (u,v) \in E & \qquad \text{s.t.} & \sum_{e: v \in e} y_e \leq c(v) \quad \forall v \in V \\ & x_u \geq 0 \quad \forall v \in V & & y_e \geq 0, \quad \forall e \in E \end{array}$$

Figure 1 The linear program relaxation of weighted vertex cover (P) and its dual program (D), which is actually the linear program relaxation of the *b-matching* problem.

In other words, the cost of augmenting the payment function p constructed by $\texttt{Alg}^{\text{pay}}$ for covering F to a payment function $p + p'$ that covers e as well, is at most $\beta \cdot \xi(F \cup \{e\}, e)$.

▶ **Theorem 2.2** ([18]). *Given an α-approximation covering payment algorithm $\texttt{Alg}^{\text{pay}}$, an augmentation algorithm $\texttt{Alg}^{\text{aug}}$ and a cost-sharing function that is β-unistrict with respect to $\texttt{Alg}^{\text{pay}}$ and $\texttt{Alg}^{\text{aug}}$, the boosted-sampling algorithm provides an $(\alpha + \beta)$-approximation algorithm for the r2SVC problem.*

Additional Notation. Throughout, we denote the number of vertices and the number of edges in the underlying graph $G = (V, E)$ by $n = |V|$ and $m = |E|$, respectively. Let $\deg(v)$ be the degree of vertex v in G and let $\Delta = \max_{v \in V} \deg(v)$ be the maximum degree.

3 Technical Challenges

In their centralized 2SVC algorithm, Gupta et al. [18] construct the first-stage solution using the following classic (centralized) primal-dual algorithm for weighted vertex cover (refer to Fig. 1 for the weighted vertex cover linear program relaxation and its dual program): The algorithm continuously raises the dual variables associated with the edges, concurrently for all of them, until a dual constraint associated with some vertex v becomes tight; at this point, v joins the vertex cover and the edges incident on v are *frozen* so that their dual variables will not be raised any further. The algorithm terminates when all edges are frozen.

A key ingredient in the analysis of Gupta et al. is that the dual variables can serve as a 1-unistrict cost-sharing function. This powerful observation utilizes the *identical runs* property stating that for any $F \subset E$ and $e \in E - F$, the runs of the algorithm on F and on $F \cup \{e\}$ are identical (with respect to the dual variables of the edges in F) up to the point when the dual variable of e becomes frozen.

The recent weighted vertex cover algorithm of Bar-Yehuda et al. [4] (see also the algorithm of Ben-Basat et al. [9]) can be viewed as a distributed implementation of this primal-dual approach.[3] They cleverly replace the centralized method of continuously (and concurrently) increasing the dual variables by a stepwise increase based on a *request-response* iterative process. An inherent property of the algorithms of [4, 9] is that in each step of the iterative process, the amount requested by vertex v from an adjacent vertex u, and hence also the amount responded by u to v's request, depend on the number of edges incident on v that have not been frozen yet. Consequently, the runs of the algorithm on $F \subset E$ and on $F \cup \{e\}$,

[3] The authors of [4, 9] present their algorithms using the equivalent local ratio approach.

for some $e \in E - F$, may differ from each other already at the early stages of the execution. As the amounts of responses sent over edge $e = (u, v)$ determine the value of the dual variable associated with e, this distributed algorithm does not satisfy the aforementioned identical runs property.

To overcome this obstacle, we develop a variant of [4]'s algorithm that does satisfy the identical runs property. This is done by grouping the iterative process' steps into *phases* so that vertex v requests the same amount throughout all steps of the phase. The crucial point here is that this amount is now fixed in advance and in particular, does not depend on the number of unfrozen edges incident on v. At the same time, our algorithm preserves the primal-dual structure in a way that allows us to efficiently combine it with an augmentation (second stage) algorithm. The main challenge is then to show that the dual variables serve as a $(1+\varepsilon)$-unistrict cost-sharing function. We prove that our algorithm is only $O(\log\log\Delta)$ factor slower than the original algorithm of [4] (whose run-time is optimal as a function of Δ). Moreover, our algorithm has a message complexity of $O(m)$, an improvement over the algorithm of [4] that sends $O\left(\frac{m \log \Delta}{\varepsilon \cdot \log\log\Delta}\right)$ messages.

4 A Distributed Algorithm for 2SVC

In this section, we present our distributed 2SVC algorithm `Alg`. As mentioned in Sec. 2, we actually present an algorithm for the r2SVC problem (Lem. 2.1 ensures that it can be transformed into a 2SVC algorithm) that can be viewed as a distributed implementation of the boosted-sampling algorithm of [18].

Similarly to the centralized version of the boosted-sampling algorithm, our distributed algorithm also constructs the random edge subset S by sampling each edge $e \in E$ with probability $\min\{1, \sigma\pi(e)\}$, only that now this sampling is done in a decentralized manner by one of e's endpoints (say, the one with the smaller ID). Following this sampling process, the vertices know which of their incident edges are included in S.

The first stage is then completed by running the distributed covering payment algorithm $\texttt{Alg}^{\text{pay}}$ presented in Sec. 4.1 on S, generating a payment function p_1 that covers S. In the second stage when the target edge subset $T \subseteq E$ is revealed, the vertices run the distributed augmentation algorithm $\texttt{Alg}^{\text{aug}}$ presented in Sec. 4.2 on S, T, and p_1 to obtain a payment function p_2 so that $p_1 + p_2$ covers T. For clarity of the exposition, we first implement our algorithms so that they may send messages over each edge in every round without taking the message size into account. Then, in Sec. 4.3, we explain how $\texttt{Alg}^{\text{pay}}$ and $\texttt{Alg}^{\text{aug}}$ can be modified to send $O(m)$ messages in total, each of size $O(\log n)$.

4.1 A Distributed Covering Payment Algorithm

Our distributed covering payment algorithm $\texttt{Alg}^{\text{pay}}$ gets as input an edge set F and constructs a payment function p that covers F. Fix some sufficiently small performance parameter $\varepsilon > 0$ (the relation of the performance parameter ε fixed here to the guaranteed approximation ratio is revealed later on). $\texttt{Alg}^{\text{pay}}$ works in phases, each consisting of $3 \cdot \lceil 2/\varepsilon \rceil$ rounds that are divided into $\lceil 2/\varepsilon \rceil$ contiguous round triples referred to as *steps*. Each step begins with a *request* round followed by a *response* round, in which the vertices exchange messages referred to as *requests* and *responses*, respectively. The last (third) round in the step is a *status* round in which the vertices report whether they remain *active* (initially, all vertices are active). Every active vertex $v \in V$ maintains a *weight* variable $w(v)$ which is initialized with $c(v)$ and is monotonically non-increasing throughout the steps of the algorithm. In addition, vertex v maintains a set $A(v)$ of active neighbors.

Consider some active vertex $v \in V$. We use (j, i) to denote step $i = 0, 1, \ldots, \lceil 2/\varepsilon \rceil - 1$ of phase j. In the request round of step (j, i), vertex v sends to each active neighbor $u \in A(v)$ the message

$$\texttt{request}_{j,i}(v, u) \; = \; 2^j \cdot \frac{\varepsilon c(v)}{\deg(v)} \, ,$$

where recall that $\deg(v)$ denotes the degree of v in the underlying graph G. In the following response round, v processes the request messages received from its active neighbors one-by-one (in an arbitrary order). For each request message $\texttt{request}_{j,i}(u, v)$, v sends a response message

$$\texttt{response}_{j,i}(v, u) \; = \; \min \left\{ \texttt{request}_{j,i}(u, v), w(v) - \varepsilon c(v) \right\}$$

and subtracts $\texttt{response}_{j,i}(v, u)$ from $w(v)$. We say that the request $\texttt{request}_{j,i}(u, v)$ is *fully responded* if $\texttt{response}_{j,i}(v, u) = \texttt{request}_{j,i}(u, v)$. Notice that this update rule ensures that $w(v) \geq \varepsilon c(v)$ throughout the response round.

In the status round, v updates the weight variable $w(v)$ by subtracting $\texttt{response}_{j,i}(u, v)$ from $w(v)$ for each response message received from an active neighbor $u \in A(v)$. Upon completion of this update process, if $w(v) \leq \varepsilon c(v)$, then v becomes inactive, sends a designated `inactive` message to its active neighbors, and marks itself as a *covering* vertex (the role of the covering vertices is revealed soon). A vertex whose active neighbor set is empty also becomes inactive. We use $w_{inact}(v)$ to denote the remaining weight of v when it becomes inactive.

The payment function $p : V \to \mathbb{R}_{\geq 0}$ constructed by $\texttt{Alg}^{\text{pay}}$ is defined by setting

$$p(v) \; = \; \min \left\{ \frac{c(v) - w_{inact}(v)}{1 - \varepsilon}, c(v) \right\} \, . \tag{1}$$

This means that $p(v)$ is always up-bounded by $c(v)$ and that $p(v) = c(v)$ if and only if v ends up as a covering vertex. Refer to Pseudocode 1 for a pseudocode description of $\texttt{Alg}^{\text{pay}}$.

4.2 A Distributed Augmentation Algorithm

Our distributed augmentation algorithm $\texttt{Alg}^{\text{aug}}$ gets as input two edge subsets $F, F' \subseteq E$ and a payment function $p : V \to \mathbb{R}_{\geq 0}$ that covers F, and constructs a payment function $p' : V \to \mathbb{R}_{\geq 0}$ so that $p + p'$ covers F' as follows. Define the *reduced* cost of each vertex $v \in V$ to be $c(v) - p(v)$. For each edge $(v_a, v_b) \in F'$, $\texttt{Alg}^{\text{aug}}$ *selects* the endpoint v_i, $i \in \{a, b\}$, with the smaller reduced cost $c(v_i) - p(v_i)$, breaking ties arbitrarily (say, by the vertex IDs). The payment function p' is then defined by setting

$$p'(v) = \begin{cases} c(v) - p(v), & \text{if } v \text{ is selected by } \texttt{Alg}^{\text{aug}} \\ 0, & \text{otherwise} \end{cases} \, .$$

This means that

$$c\left(\texttt{Alg}^{\text{aug}}\left(F, F', p\right)\right) \; \leq \sum_{(v_a, v_b) \in F'} \min \left\{ c(v_a) - p(v_a), c(v_b) - p(v_b) \right\} \, . \tag{2}$$

Algorithm 1 A covering payment algorithm, code for vertex v.

```
1: for $j = 0, 1, \ldots$ do
2:     for $i = 0$ to $\lceil 2/\varepsilon \rceil - 1$ do
           Request Round
3:         for each $u \in A(v)$ do
4:             $\texttt{request}_{j,i}(v,u) \leftarrow 2^j \cdot \frac{\varepsilon c(v)}{\deg(v)}$
5:             Send $\texttt{request}_{j,i}(v,u)$ to $u$
           Response Round
6:         for each $u \in A(v)$ do
7:             $\texttt{response}_{j,i}(v,u) \leftarrow \min\left\{\texttt{request}_{j,i}(u,v), w(v) - \varepsilon c(v)\right\}$
8:             $w(v) \leftarrow w(v) - \texttt{response}_{j,i}(v,u)$
9:             Send $\texttt{response}_{j,i}(v,u)$ to $u$
           Status Round
10:        for each $u \in A(v)$ do
11:            $w(v) \leftarrow w(v) - \texttt{response}_{j,i}(u,v)$
12:        if $w(v) \leq \varepsilon c(v)$ then
13:            Send inactive to all neighbors
14:            return $p(v) \leftarrow c(v)$                ▷ $v$ is marked as a covering vertex
15:        for each inactive message received from $u \in A(v)$ do
16:            $A(v) \leftarrow A(v) - \{u\}$
17:        if $A(v) = \emptyset$ then
18:            return $p(v) \leftarrow \frac{1}{1-\varepsilon}(c(v) - w(v))$
```

4.3 Fewer and Smaller Messages

While $\texttt{Alg}^{\text{aug}}$ requires a single round of communication and hence, does not send more than $O(m)$ messages in total, the aforementioned implementation of $\texttt{Alg}^{\text{pay}}$ dictates sending messages over each edge in every step which sums up to $\Omega(m \log(\Delta)/\varepsilon)$ messages. Here we show that $\texttt{Alg}^{\text{pay}}$ can be modified to send $O(1)$ messages over each edge throughout the execution, which accounts for a total of $O(m)$ messages as well. We then explain how the message size (of both $\texttt{Alg}^{\text{pay}}$ and $\texttt{Alg}^{\text{aug}}$) can be bounded by $O(\log n)$.

The modified $\texttt{Alg}^{\text{pay}}$ starts with a *handshake* round in which each vertex $v \in V$ sends its cost $c(v)$ and degree $\deg(v)$ to all its neighbors. The next messages that v is certain to send are the `inactive` messages sent when v becomes inactive. Consider some neighbor u of v and suppose that it still did not receive an `inactive` message from v. Using the handshake information, vertex u can compute the value of each request message $\texttt{request}_{j,i}(v,u) = 2^j \cdot \varepsilon c(v)/\deg(v)$ without actually receiving it from v. Moreover, as long as u does not hear otherwise from v, it infers that v fully responds to its own requests, namely, that $\texttt{response}_{j,i}(v,u) = \texttt{request}_{j,i}(u,v)$, again, without actually receiving v's response. If v cannot fully respond to u's request in some step (j,i), then v sends the partial response $\texttt{response}_{j,i}(v,u)$ as in the original implementation of $\texttt{Alg}^{\text{pay}}$, but notice that this will happen at most once as v must become inactive in step (j,i).

With this modification, a vertex $v \in V$ sends messages over an incident edge (v,u) in three occasions: (1) a message encoding $c(v)$ and $\deg(v)$ during the designated handshake round of $\texttt{Alg}^{\text{pay}}$; (2) a message encoding a partial response to $\texttt{request}_{j,i}(u,v)$ during the (single) step (j,i) in which v becomes inactive; and (3) a message encoding the reduced cost of v during the execution of $\texttt{Alg}^{\text{aug}}$ in the second stage. The message sent in occasion (1) is of size $O(\log n)$ by the assumption on the vertex costs. To ensure that the messages sent during occasions (2) and (3) are also of size $O(\log n)$, we round down their numeric content to the next power of $1+\varepsilon$. This affects the approximation ratio analyzed in Sec. 5 by a factor of at most $1+\varepsilon$.

5 Analysis

In this section, we analyze the distributed implementation $\texttt{Alg}$ of the boosted-sampling algorithm presented in Sec. 4, establishing the following theorem.

▶ **Theorem 5.1.** *For any sufficiently small $\varepsilon > 0$, $\texttt{Alg}$ completes the first stage in $O(\log(\Delta)/\varepsilon)$ rounds and the second stage in $O(1)$ rounds and guarantees to return a $(3+O(\varepsilon))$-approximation for the r2SVC problem.*

To obtain an approximation ratio of $(3+\varepsilon')$ for some $0 < \varepsilon' < 1$ (as promised in Sec. 1), one merely has to set the performance parameter ε used by $\texttt{Alg}$ so that $\varepsilon \leftarrow \varepsilon'/\kappa$ for the appropriate constant κ hidden in the O-notation in the statement of Thm. 5.1. We start the analysis by proving that $\texttt{Alg}^{\text{pay}}$ constructs a covering payment function.

▶ **Lemma 5.2.** *When invoked on an edge subset $F \subseteq E$, $\texttt{Alg}^{\text{pay}}$ constructs a payment function p that covers F.*

Proof. When a vertex $v \in V$ becomes inactive, either (1) its active neighbor set is empty; or (2) its weight variable $w(v)$ satisfies $w(v) \leq \varepsilon c(v)$. In the latter case, v is a covering vertex with $p(v) = c(v)$. The assertion follows by the design of $\texttt{Alg}^{\text{pay}}$ ensuring that if $(u, v) \in F$, then it cannot be the case that both u and v become inactive due to the former reason. ◀

Let $w_j(v)$ and $A_j(v)$ denote the weight and active neighbor set of v at the beginning of phase j, respectively, and let $\deg_j^A(v)$ denote the size of $A_j(v)$. The following two lemmas play a crucial role in proving that $\texttt{Alg}^{\text{pay}}$ is a $(2+O(\varepsilon))$-approximation covering payment algorithm.

▶ **Lemma 5.3.** *For every phase j in the run of $\texttt{Alg}^{\text{pay}}$ and for every vertex $v \in V$ that is active at the beginning of phase j, it holds that $\deg_j^A(v) \leq \deg(v)/2^j$.*

Proof. Consider some vertex $v \in V$ and recall that $\deg_0^A(v)$ is the degree of v in (V, F). Since $\deg_0^A(v)$ is up-bounded by $\deg(v)$, the assertion holds for phase $j = 0$. Assume by contradiction that $\deg_j^A(v) > \deg(v)/2^j$ for some phase $j > 0$ where v is still active. This means that v had more than $\deg(v)/2^j$ active neighbors throughout phase $j-1$ and in each of the $\lceil 2/\varepsilon \rceil$ steps of that phase, more than $\deg(v)/2^j$ requests of v were fully responded. Since $\texttt{request}_{j-1,i}(v, u) = 2^{j-1} \cdot \varepsilon c(v)/\deg(v)$ for every $u \in A_{j-1}(v)$ and $i = 0, 1, \ldots, \lceil 2/\varepsilon \rceil - 1$, it follows that

$$w_j(v) \;<\; w_{j-1}(v) - \lceil 2/\varepsilon \rceil \cdot \frac{\deg(v)}{2^j} \cdot 2^{j-1} \cdot \frac{\varepsilon c(v)}{\deg(v)} \;\leq\; w_{j-1}(v) - c(v) \;\leq\; 0\,.$$

Ergo, v must become inactive by the beginning of phase j, in contradiction to the assumption. ◀

▶ **Lemma 5.4.** *For every vertex $v \in V$, the weight variable $w(v)$ remains non-negative throughout the run of $\texttt{Alg}^{\text{pay}}$.*

Proof. Since the weight of v does not decrease once v becomes inactive, it suffices to show that $w(v)$ remains non-negative as long as v is active. In each step the weight of v can decrease only during the response round and the status round. Recall that $w(v) \geq \varepsilon c(v)$ throughout every response round, therefore we are left to show that in every status round, the amount decreased from $w(v)$ is not larger than $\varepsilon c(v)$.

Consider some step (j, i). Each response message $\texttt{response}_{j,i}(u, v)$ that v processes during this step is not larger than $\texttt{request}_{j,i}(v, u) = 2^j \cdot \varepsilon c(v) / \deg(v)$. Therefore, it remains to show that v receives at most $\deg(v)/2^j$ response messages in this step. As the number of active neighbors is monotonically non-increasing, this follows from Lem. 5.3 ensuring that v has at most $\deg(v)/2^j$ active neighbors at the beginning of phase j. ◀

For the purpose of further analysis, we define the functions $y_{j,i} : 2^E \times E \to \mathbb{R}_{\geq 0}$ and $y : 2^E \times E \to \mathbb{R}_{\geq 0}$ as follows. Consider a run of $\texttt{Alg}^{\text{pay}}$ on an edge subset $F \subseteq E$. For each step (j, i) in this run and edge $e = (u, v) \in E$, let $y_{j,i}(F, e) = \texttt{response}_{j,i}(u, v) + \texttt{response}_{j,i}(v, u)$ and let

$$y(F, e) \; = \sum_{j=0,1,\ldots,\; i=0,1,\ldots,\lceil 2/\varepsilon \rceil - 1} y_{j,i}(F, e) \, .$$

We generalize the last definition from single edges e to edge subsets $J \subseteq E$, defining $y(F, J) = \sum_{e \in J} y(J, e)$. In addition, let $F(v) = \{e \in F : v \in e\}$ denote the set of edges in F incident on vertex $v \in V$. Notice that for every vertex $v \in V$, the sum of response messages sent over edges in $F(v)$ equals the total amount decreased from $w(v)$ throughout the execution, which yields the following observation.

▶ **Observation 5.5.** *For every vertex $v \in V$, the function y satisfies $y(F, F(v)) = c(v) - w_{inact}(v)$.*

This facilitates approximating $p(v)$ by $y(F, F(v))$.

▶ **Lemma 5.6.** *When invoked on an edge subset $F \subseteq E$, $\texttt{Alg}^{\text{pay}}$ constructs a payment function p that satisfies*

$$(1 - \varepsilon) \cdot p(v) \; \leq \; y(F, F(v)) \; \leq \; p(v)$$

for every vertex $v \in V$.

Proof. Recall that $p(v) = \min\left\{\frac{1}{1-\varepsilon}(c(v) - w_{inact}(v)), c(v)\right\}$ (see (1)). If $\frac{1}{1-\varepsilon}(c(v) - w_{inact}(v)) \leq c(v)$, then the assertion holds by Obs. 5.5 implying that $p(v) = \frac{1}{1-\varepsilon} \cdot y(F, F(v))$. So, consider the case where $\frac{1}{1-\varepsilon}(c(v) - w_{inact}(v)) > c(v)$ which means that $w_{inact}(v) < \varepsilon \cdot c(v)$. Lem. 5.4 ensures that $w_{inact}(v) \geq 0$, thus by combining it with Obs. 5.5 we conclude that $(1 - \varepsilon) \cdot c(v) < y(F, F(v)) \leq c(v)$. The assertion follows as $p(v) = c(v)$ in this case. ◀

Employing Obs. 5.5 we can establish also the following lemma.

▶ **Lemma 5.7.** *The function $y : 2^E \times E \to \mathbb{R}_{\geq 0}$ satisfies $y(F, F) \leq c(\texttt{Opt}(F))$.*

Proof. For every vertex $v \in V$, Lem. 5.4 guarantees that $w_{inact}(v) \geq 0$, hence, in conjunction with Obs. 5.5, we get that $y(F, F(v)) \leq c(v)$. It follows then that the variables $y(F, e)$, $e \in F$, constitute a feasible solution for the dual program of the linear program relaxation of weighted vertex cover on graph (V, F) with cost function $c(\cdot)$ (refer to Fig. 1 for that program). The assertion follows as a consequence of weak duality. ◀

We are now ready to establish the approximation ratio of $\texttt{Alg}^{\text{pay}}$.

▶ **Lemma 5.8.** $\texttt{Alg}^{\text{pay}}$ *is a* $(2+O(\varepsilon))$*-approximation covering payment algorithm.*

Proof. The payment function p constructed by $\texttt{Alg}^{\text{pay}}$ satisfies

$$\sum_{v \in V} p(v) \;\leq\; (1+O(\varepsilon)) \sum_{v \in V} y(F, F(v)) \;=\; 2 \cdot (1+O(\varepsilon)) \cdot y(F, F) \;\leq\; (2+O(\varepsilon)) \cdot c(\texttt{Opt}(F)) \, ,$$

where the first transition is due to Lem. 5.6 and the last transition follows from Lem. 5.7. ◀

It follows directly from the design of $\texttt{Alg}^{\text{aug}}$ that given two edge subsets $F, F' \subseteq E$ and a payment function $p : V \to \mathbb{R}_{\geq 0}$ that covers F, the payment function p' constructed by $\texttt{Alg}^{\text{aug}}$ when invoked on (F, F', p) augments p so that $p + p'$ covers F'. So, $\texttt{Alg}^{\text{aug}}$ is a valid augmentation algorithm (that runs in $O(1)$ rounds). The desired bound on the approximation ratio of $\texttt{Alg}$ can now be established by identifying a proper cost-sharing function.

▶ **Lemma 5.9.** *The function* $y : 2^E \times E \to \mathbb{R}_{\geq 0}$ *is a* $(1+O(\varepsilon))$*-unistrict cost-sharing function with respect to* $\texttt{Alg}^{\text{pay}}$ *and* $\texttt{Alg}^{\text{aug}}$*.*

Proof. We need to show that y satisfies properties P1–P3 (see Sec. 2) with respect to $\texttt{Alg}^{\text{pay}}$ and $\texttt{Alg}^{\text{aug}}$. When $\texttt{Alg}^{\text{pay}}$ is invoked on an edge subset $F \subseteq E$, no messages are sent over edges that are not in F, so property P1 clearly holds. Moreover, Lem. 5.7 ensures that $\sum_{e \in F} y(F, e) \leq c(\texttt{Opt}(F))$, thus property P2 holds as well.

For property P3, we fix some $F \subset E$ and $e \in E - F$ and show that

$$c\left(\texttt{Alg}^{\text{aug}}(F, \{e\}, \texttt{Alg}^{\text{pay}}(F))\right) \leq (1 + O(\varepsilon)) \cdot y(F \cup \{e\}, e) \, .$$

To that end, we define the following additional notation: Given an edge subset $J \subseteq F$ and a step (j, i), let

$$y_{<(j,i)}(F, J) \;=\; \sum_{(j',i') < (j,i)} \sum_{e \in J} y_{j',i'}(F, e) \, ,$$

where we use the relation $<$ to denote the lexicographic order, and let

$$y_{\leq(j,i)}(F, J) \;=\; y_{<(j,i)}(F, J) + \sum_{e \in J} y_{j,i}(F, e) \, .$$

Consider two runs of $\texttt{Alg}^{\text{pay}}$: run R^1 on F and run R^2 on $F \cup \{e\}$. Let (j, i) be the first step in which an endpoint of e becomes inactive during R^2. Let $R^1_{j,i}$ and $R^2_{j,i}$ denote the runs R^1 and R^2 up to the beginning of step (j, i), respectively.

Observe that $R^1_{j,i}$ and $R^2_{j,i}$ are identical in the sense that $y_{<(j,i)}(F, J) = y_{<(j,i)}(F \cup \{e\}, J)$ for any $J \subseteq F$. In other words, the same response messages are sent over every edge $e' \in F$ throughout $R^1_{j,i}$ and $R^2_{j,i}$. This is due to the fact that the request messages that (an active) vertex $v \in V$ sends in every step (j', i') do not depend on the input of $\texttt{Alg}^{\text{pay}}$, but only on $\deg(v)$ and j'. Since both endpoints of edge e' are active throughout $R^2_{j,i}$, it follows that the request messages sent over every edge in F are the same in $R^1_{j,i}$ and $R^2_{j,i}$, hence so are the response messages.

Let u be the endpoint of e that becomes inactive in step (j, i) of run R^2 (if both endpoints of e become inactive in this step, then take u to be one of them) and recall that u is marked as a covering vertex. We argue that for each edge $e' \in F(u)$, it holds that $y_{j,i}(F \cup \{e\}, e') \leq y_{j,i}(F, e')$. This follows from the fact that in step (j, i) of run R^2, once $w(u)$ reaches the $\varepsilon \cdot c(u)$ threshold, it starts sending response messages with 0 amount over the edges in $F(u)$, while in step (j, i) of R^1, u may still send response messages with positive amounts over these edges. Therefore, $y_{\leq(j,i)}(F \cup \{e\}, F(u)) \leq y_{\leq(j,i)}(F, F(u))$.

At the end of step (j,i) of R^2 we know that u is inactive and $w(u) \leq \varepsilon c(u)$, so the total amount decreased from $w(u)$ throughout this run is at least $(1-\varepsilon) \cdot c(u)$. Therefore, the amount decreased from $w(u)$ throughout R^2 due to messages sent over edges other than e is at least $(1-\varepsilon) \cdot c(u) - y(F \cup \{e\}, e)$, hence

$$y_{\leq(j,i)}(F, F(u)) \geq y_{\leq(j,i)}(F \cup \{e\}, F(u)) \geq (1-\varepsilon) \cdot c(u) - y(F \cup \{e\}, e)\,.$$

Consequently, the total amount decreased from $w(u)$ throughout R^1 is at least $(1-\varepsilon) \cdot c(u) - y(F \cup \{e\}, e)$, thus, at the end of R^1

$$w_{inact}(u) \leq c(u) - ((1-\varepsilon) \cdot c(u) - y(F \cup \{e\}, e)) = \varepsilon c(u) + y(F \cup \{e\}, e)\,. \tag{3}$$

Let p be the payment function constructed by $\texttt{Alg}^{\text{pay}}(F)$ and let p' be the payment function constructed by $\texttt{Alg}^{\text{aug}}(F, \{e = (u,v)\}, p)$. As p is a covering payment function for F, the bound in (2) implies that $c(p') \leq \min\{c(u) - p(u), c(v) - p(v)\} \leq c(u) - p(u)$. Recalling the definition of p (1), if $p(u) = c(u)$, then $c(p') = 0$ and the assertion trivially holds. Otherwise, $p(u) = \frac{c(u) - w_{inact}(u)}{1-\varepsilon}$ which means that

$$c(p') \leq c(u) - p(u) = c(u) - \frac{c(u) - w_{inact}(u)}{1-\varepsilon} = \frac{w_{inact}(u) - \varepsilon c(u)}{1-\varepsilon}\,.$$

Combined with (3), we conclude that

$$c(p') \leq \frac{y(F \cup \{e\}, e)}{1-\varepsilon} = (1 + O(\varepsilon)) \cdot y(F \cup \{e\}, e)\,,$$

thus establishing the assertion. ◀

Combining Lem. 5.8 and 5.9 with Thm. 2.2, we conclude that $\texttt{Alg}$ returns a $(3 + O(\varepsilon))$-approximation for the r2SVC problem, as promised in Thm. 5.1. It remains to analyze $\texttt{Alg}$'s run-time.

▶ **Lemma 5.10.** $\texttt{Alg}$ *completes the first stage in* $O(\log(\Delta)/\varepsilon)$ *rounds.*

Proof. The edge sampling performed at the beginning of the first stage takes $O(1)$ rounds, so it is left to bound the run-time of $\texttt{Alg}^{\text{pay}}$. For every vertex $v \in V$, Lem. 5.3 ensures that after $O(\log \deg_0^A(v)) = O(\log \deg(v))$ phases, the neighbor set of v must be empty which means that v must become inactive. Since each phase consists of $3 \cdot \lceil 2/\varepsilon \rceil$ rounds, vertex v completes the execution in $O(\log(\deg(v))/\varepsilon) = O(\log(\Delta)/\varepsilon)$ rounds. ◀

The second stage includes only the execution of $\texttt{Alg}^{\text{aug}}$, that requires a single round of communication. By that we complete the run-time analysis of $\texttt{Alg}$, thus establishing Thm. 5.1.

6 Discussion and Open Questions

In this paper, we restrict our attention to the case where the target edge set T is drawn from a product distribution, which means that the edge sampling events are independent. The more general case, that allows the edges to exhibit complicated dependencies, is studied extensively in the centralized two-stage stochastic optimization literature and it will be interesting to come up with a distributed method that supports it. This requires overcoming the following obstacle though: it is not clear how to sample the target edge set T in a distributed fashion if the events $e \in T$ and $e' \in T$ depend on each other for edges e and e' that are far away in the graph.

A related open question is concerned with developing a meaningful notion of a finite scenario model for distributed two-stage stochastic optimization. Here too it is not clear how the vertices can select a target edge set in a correlated manner. Overcoming this obstacle may also be the first step towards a distributed sample average approximation approach.

It would be interesting to extend our results to the *k-stage* stochastic optimization model as well. In this model, the vertex cover is constructed in k (that may be larger than 2) stages so that in each stage, the probability distribution gets refined, but the inflation factor is increased. Another interesting extension would be a model where the revelation of the target edge set is not necessarily done for all vertices at once, but rather, different vertices may receive their local image of the target edge set's realization in different rounds.

Besides that, we would like to examine whether the run-time complexity of our algorithm can be improved. Specifically, reducing the dependency on ε from linear to logarithmic would yield a distributed 3-approximation algorithm for the 2SVC problem whose run-time is polylogarithmic (cf. [9]), although we have yet to find a way to do so. Finally, we believe that there are great opportunities in adapting other graph theoretic optimization problems to the distributed stochastic setting.

References

1 Matti Åstrand, Patrik Floréen, Valentin Polishchuk, Joel Rybicki, Jukka Suomela, and Jara Uitto. A local 2-approximation algorithm for the vertex cover problem. In *International Symposium on Distributed Computing*, pages 191–205. Springer, 2009.

2 Matti Åstrand and Jukka Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*, pages 294–302. ACM, 2010.

3 Ilke Bakir, Natashia Boland, Brian Dandurand, and Alan Erera. Scenario set partition dual bounds for multistage stochastic programming: A hierarchy of bounds and a partition sampling approach. *Optimization Online*, 2016.

4 Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. A Distributed (2+ ε)-Approximation for Vertex Cover in O (log Δ/ε log log Δ) Rounds. *Journal of the ACM (JACM)*, 64(3):23, 2017.

5 Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.

6 Reuven Bar-Yehuda and Shimon Even. A Local-Ratio Theorem for Approximating the Weighted Vertex Cover Problem. In *North-Holland Mathematics Studies*, volume 109, pages 27–45. Elsevier, 1985.

7 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM (JACM)*, 63(3):20, 2016.

8 Evelyn ML Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 173–184, 1955.

9 Ran Ben-Basat, Guy Even, Ken-ichi Kawarabayashi, and Gregory Schwartzman. A Deterministic Distributed 2-Approximation for Weighted Vertex Cover in O(\log N\log \varDelta /\log ^2\log \varDelta) Rounds. In *International Colloquium on Structural Information and Communication Complexity*, pages 226–236. Springer, 2018.

10 John R Birge and Francois Louveaux. *Introduction to stochastic programming.* Springer Science & Business Media, 2011.

11 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms.* MIT press, 2009.

12 George B Dantzig. LINEAR PROGRAMMING UNDER UNCERTAINTY. *Science*, 1955.

13 Kedar Dhamdhere, Vineet Goyal, R Ravi, and Mohit Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 367–376. IEEE, 2005.

14 Shane Dye, Leen Stougie, and Asgeir Tomasgard. The stochastic single resource service-provision problem. *Naval Research Logistics (NRL)*, 50(8):869–887, 2003.

15 Uriel Feige, Kamal Jain, Mohammad Mahdian, and Vahab Mirrokni. Robust combinatorial optimization with exponential scenarios. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 439–453. Springer, 2007.

16 Manuela Fischer. Improved Deterministic Distributed Matching via Rounding. In *31st International Symposium on Distributed Computing, DISC*, pages 17:1–17:15, 2017.

17 Fabrizio Grandoni, Jochen Könemann, and Alessandro Panconesi. Distributed weighted vertex cover via maximal matchings. *ACM Transactions on Algorithms (TALG)*, 5(1):6, 2008.

18 Anupam Gupta, Martin Pál, R Ravi, and Amitabh Sinha. Sampling and cost-sharing: Approximation algorithms for stochastic optimization problems. *SIAM Journal on Computing*, 40(5):1361–1401, 2011.

19 Anupam Gupta, R Ravi, and Amitabh Sinha. LP rounding approximation algorithms for stochastic network design. *Mathematics of Operations Research*, 32(2):345–364, 2007.

20 Michal Hanckowiak, Michal Karonski, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. *SIAM Journal on Discrete Mathematics*, 15(1):41–57, 2001.

21 Nicole Immorlica, David Karger, Maria Minkoff, and Vahab S Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 691–700. Society for Industrial and Applied Mathematics, 2004.

22 Amos Israeli and Alon Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

23 George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms (TALG)*, 5(4):41, 2009.

24 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

25 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

26 Samir Khuller, Uzi Vishkin, and Neal Young. A primal-dual parallel approximation technique applied to weighted set and vertex covers. *Journal of Algorithms*, 17(2):280–289, 1994.

27 Willem K Klein Haneveld and Maarten H van der Vlerk. Stochastic integer programming: General models and algorithms. *Annals of Operations Research*, 85:39–57, 1999.

28 Christos Koufogiannakis and Neal E Young. Distributed algorithms for covering, packing and maximum weighted matching. *Distributed Computing*, 24(1):45–63, 2011.

29 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 980–989. Society for Industrial and Applied Mathematics, 2006.

30 George L Nemhauser and Leslie Earl Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

31 Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed computing*, 14(2):97–100, 2001.

32 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2000. URL: `https://books.google.co.il/books?id=T1hFWuDi1CsC`.

33 Valentin Polishchuk and Jukka Suomela. A simple local 3-approximation algorithm for vertex cover. *Information Processing Letters*, 109(12):642–645, 2009.

34 R Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming*, 108(1):97–114, 2006.

35 Rüdiger Schultz, Leen Stougie, and Maarten H van der Vlerk. Two-stage stochastic integer programming: a survey. *Statistica Neerlandica*, 50(3):404–416, 1996.

36 David B Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *Journal of the ACM (JACM)*, 53(6):978–1012, 2006.

37 Aravind Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1305–1313, 2007.

38 Chaitanya Swamy and David B Shmoys. Sampling-based approximation algorithms for multi-stage stochastic optimization. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 357–366. IEEE, 2005.

39 Chaitanya Swamy and David B Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. *ACM SIGACT News*, 37(1):33–46, 2006.