Stochastic Budget Optimization in Internet Advertising

Bhaskar DasGupta
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
dasgupta@cs.uic.edu

S. Muthukrishnan
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
muthu@cs.rutgers.edu

January 18, 2012

Abstract

Internet advertising is a sophisticated game in which the many advertisers "play" to optimize their return on investment. There are many "targets" for the advertisements, and each "target" has a collection of games with a potentially different set of players involved. In this paper, we study the problem of how advertisers allocate their budget across these "targets". In particular, we focus on formulating their *best response* strategy as an optimization problem. Advertisers have a set of keywords ("targets") and some stochastic information about the future, namely a probability distribution over *scenarios* of cost vs click combinations. This summarizes the potential states of the world assuming that the strategies of other players are fixed. Then, the best response can be abstracted as *stochastic budget optimization* problems to figure out how to spread a given budget across these keywords to maximize the expected number of clicks.

We present the *first known* non-trivial poly-logarithmic approximation for these problems as well as the first known hardness results of getting better than logarithmic approximation ratios in the various parameters involved. We also identify several special cases of these problems of practical interest, such as with fixed number of scenarios or with polynomial-sized parameters related to cost, which are solvable either in polynomial time or with improved approximation ratios. Stochastic budget optimization with scenarios has sophisticated technical structure. Our approximation and hardness results come from relating these problems to a special type of (0/1, bipartite) quadratic programs inherent in them. Our research answers some open problems raised by the authors in (Stochastic Models for Budget Optimization in Search-Based Advertising, Algorithmica, 58 (4), 1022-1044, 2010).

1 Introduction

This paper deals with the problem of how advertisers allocate their budget in Internet advertising. In sponsored search, users who pose queries to internet search engines are not only provided search results, but also a small set of text ads. These ads are chosen from a set of campaigns set up by advertisers based on the keywords in the search query. A lot of focus has been on how these ads are chosen and priced, which is via an auction that is by now well known [2, 10, 20][1]. Our focus is instead on the problem faced by advertisers. Even small advertisers have many *keywords*, a budget in mind and must figure out how to spread this budget on bids for each of these keywords. This is a highly nontrivial task, and the basis for a separate industry to support advertisers. A similar problem arises with "display ads" where advertisers have *websites* where their ads will be shown

[1]Likewise, there was a lot of work on bidding strategies [4, 11, 19, 23]. This paper extends that body of work by considering a richer model of uncertainty; see subsequent paragraphs.

and need to split their budget for the ad campaign across the sites to be most effective. Likewise, in behavioral targeting, advertisers have to decide how to spread their budget across *behavior groups*. In all these cases, therefore, advertisers have various "targets" and wish to split their budget across them to optimize their ad campaigns.

Consider the sponsored search example and fix an advertiser A. They have many keywords that they would like to target for their ads. How should they bid for each, given some overall budget they can spend? There is a sophisticated underlying game in which the many advertisers "play" to optimize their return on investment simultaneously. For each keyword and for each instance of auction triggered by this keyword, there is potentially a different set of competing advertisers involved. Building effective strategies is challenging amidst so many parameters. A fundamental and widely accepted proposal is for the advertiser A to pursue a *best response* strategy, *i.e.*, fix the strategies of other advertisers and pick the best strategy as one's response. Besides being a simple and easy strategy to understand and hence suitable for experimentation by advertisers, best response has desirable properties. For example, in the absence of budgets and for single repeated auctions, special type of best response by every player leads to the VCG outcome [5, 6, 10, 20].

In order to help the advertisers implement this best response strategy, search engines provide them with expected bid versus clicks function for each keyword[2]. Assuming that the rest of the world is fixed, these functions provide an estimate of the expected number of clicks an advertiser would obtain by bidding a certain value on that keyword. These functions can also be "learned" by an advertiser to some extent by systematically trying out various bids. Finding advertiser's best response bidding strategy then becomes an optimization problem where the goal is to maximize the expected number of clicks assuming access to these functions. The resulting problems are in the spirit of the Knapsack problem [4, 11, 19, 23] with many of them solvable nearly exactly or with constant factor approximations.

A more general approach is to acknowledge that, in reality, the bids vs clicks functions are *not fixed*, but rather random variables with unknown correlations and uncertainties: number of queries (and hence, clicks and budget spent on a keyword) change each day, relative occurrences of keywords change (*e.g.*, searches for beach and snow are complementary[3]), and so on. Therefore, one has to consider a specific stochastic model for these random variables and then maximize the *expected number of clicks* under that model. This approach was initiated in [19] leading to a stochastic budget optimization problem that is studied in this paper.

1.1 Organization of the paper

For convenience of the readers, we organize the rest of the paper in the following manner.

- We start with Section 2 which describes all of our stochastic budget optimization models and corresponding computational problems precisely, starting from the simplest one, together with some comments and justifications about the model. In the last subsection of this section (Section 2.5), we fix some notational uniformity for readers convenience.
- In Section 3, we summarize the results obtained in this paper. For the benefit of the reader, we group the results into two categories, namely a set of main results that deal with the com-

[2]See, for example, Traffic Estimator at `http://adwords.google.com/support/aw/bin/answer.py?hl=en&answer=8692`, bidding tutorial at `http://adwords.google.com/support/aw/bin/answer.py?hl=en&answer=163828` and bid simulator at `http://adwords.google.com/support/aw/bin/answer.py?hl=en&answer=138148`

[3]See `www.google.com/trends?q=beach\%2C+snow&ctab=0&geo=all&date=all&sort=0` for yearly and `www.google.com/trends?q=clubs\%2C+stocks&ctab=0&geo=all&date=mtd&sort=0` for weekly trends.

putational complexity issues of the original models without restrictions and a set of additional results that deal with variations and special cases of the models defined in Section 2.

The remaining sections of the paper, excluding conclusion and references, deal with precise statements of our results and technical details of their proofs. For complex proofs, we first provide a more informal overview of the steps in the proof before proceeding with technical details. These sections are organized in the following manner.

- In Section 4 we discuss the quadratic integer programming reformulations of the various SBO problems.
- In Section 5 we state and prove our poly-logarithmic approximation algorithms for SSBO and MULTI-SSBO problems (main result **(R1)**).
- In Section 6, we state and prove our approximation-hardness results for both SSBO and MULTI-SSBO problems (main result **(R2)**).
- Section 7 contain all other results:
  - In Section 7.1 we show that many SSBO problems have improved solutions if certain parameters are restricted in their range of values.
  - In Section 7.2 we show the limitations of semidefinite programming based approaches for solving SSBO problems.

2 Scenario Model for Stochastic Budget Optimization

We discuss the scenario model[4] and related problems using the language of sponsored search[5]. We use the suffix SSBO (Scenario Stochastic Budget Optimization) for various acronyms for different versions of our problems. For the convenience of the readers and to delay introducing more involved notations, we first start with a slightly simpler version of the model involving only one slot. We refer to this version as the "uniform cost" case and describe it in the next section.

2.1 Single Slot Case: Uniform Cost Model

This basic model starts with the following assumptions:

- There is a single slot for advertising.
- We have a set of n keywords $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_n$ with the keyword $\mathcal{K}_j$ having a *cost-per-click* d_j (a positive integer).
- We have a positive integer B denoting the *budget* for the advertiser.
- We have a collection of m "scenarios" where the ith scenario is characterized by the following parameters:
  - A probability of ε_i ($\sum_{i=1}^{m} \varepsilon_i = 1$).

[4]The scenario model was introduced in [19]. For a very detailed discussion of prior works related to the approach in the model, see Section 1.4 of [19].

[5]Our discussion can easily be adopted to other internet ad channels like display ads and behavioral targeting.

– A "click vector" $(a_{i,1}, a_{i,2}, \ldots, a_{i,n})$ where each $a_{i,j} \geq 0$ is an integer. Each $a_{i,j}$ denotes the number of clicks obtained by the jth keyword $\mathcal{K}_j$ in the ith scenario.

Scenarios can be thought of as sampling the model over various times[6].

Our general goal is to compute n selection variables $x_1, x_2, \ldots, x_n$, where x_j corresponds to the jth keyword, to *maximize* a suitable total payoff. A crucial aspect of the discussed formulation is that, if the budget is not limiting, then the payoff corresponds to the total number of expected clicks, but if the budget turns out to be limiting for any scenario then the payoff *scales* the total number of expected clicks by the fraction that the budget would provide[7]. Based on the above intuition, our precise goal is *maximize* the total *expected* payoff over *all* scenarios, *i.e.*,

$$\mathsf{maximize}\ \mathbb{E}[\text{payoff}] = \sum_{i=1}^{m} \mathbb{E}[\text{payoff}_i]$$

where the $\mathsf{expected\ payoff}$ $\mathbb{E}[\text{payoff}_i]$ for the ith scenario is

$$\mathbb{E}[\text{payoff}_i] = \begin{cases} \varepsilon_i \sum_{j=1}^{n} a_{i,j} x_j, & \text{if } \sum_{j=1}^{n} a_{i,j} d_j x_j \leq B \\ \dfrac{B}{\sum_{j=1}^{n} a_{i,j} d_j x_j} \left(\varepsilon_i \sum_{j=1}^{n} a_{i,j} x_j \right), & \text{otherwise} \end{cases} \tag{1}$$

Following [19], we distinguish between two versions of the problem based on the nature of the selection variables:

Integral version (Uniform-Int-Ssbo)**:** $x_j \in \{0, 1\}$ for all j. This corresponds to the case when based on the stochastic information, either the advertiser chooses to win and pay for all clicks for a keyword, or not at all. Hence, the strategy of the advertiser is *deterministic*.

Fractional version (Uniform-Frac-Ssbo)**:** $0 \leq x_j \leq 1$ for all j. This can be thought of as a strategy in which the advertiser treats these numbers as *probabilities* and bids for the keywords in a *randomized* fashion based on these probabilities, thereby only winning (and paying for) a portion of all clicks and impressions for each keyword. If the deterministic strategy is hard to compute and provides a solution of bad quality then the randomized strategy is more desirable.

Other than the scenario model, there are at least two other possible models for stochastic budget optimization as discussed in [19]. In the *proportional model* there is just one global random variable for the total number of clicks in the day that keeps the relative proportions of clicks for different keywords the same, whereas in the *independent keywords model* each keyword comes with its own probability distribution. However, among all these models this scenario-based model is perhaps one of the most natural model of reality and provides an appropriate middle ground between complex arbitrary joint probability distribution and a single distribution for all keywords. It was shown in [19] that both Uniform-Int-Ssbo and Uniform-Frac-Ssbo are NP-hard. In the sequel, we assume without loss of generality that $1 = d_1 \leq d_2 \leq \ldots \leq d_n$.

[6]Scenarios can be provided by the search engine for the advertisers, or used by the search engines to bid on behalf of advertisers. Similarly, advertisers and other search engine optimizers can also "infer" scenarios indirectly using trends and other data provided by search engines.

[7]The underlying assumption is that, within a scenario, the queries and keywords are well-mixed and, when budget runs out, the ad campaign is halted for the period as is currently done. The queries and keywords are well-mixed not only because of aggregation of streams from millions of users but also because of ad throttling that spreads out the eligible ad campaigns over the period of a scenario. See [19] for exact details of justification.

2.2 Single Slot Case: General Model

In a more realistic version of the SSBO problems the cost-per-click values may vary *slightly* over a range of scenarios due to their small errors in estimation. This can be modeled by introducing a stretch parameter (*small integer*)[8] $1 \leq \kappa = O\left(\mathsf{poly}(\log(m+n))\right)$. Now, d_j stands for the *basic* cost-per-click for the keyword $\mathcal{K}_j$, whereas the *real* cost-per-click for the keyword $\mathcal{K}_j$ in the ith scenario is denoted by $c_{i,j}$, with $c_{i,j} \in [d_j, \kappa d_j)$[9]. Then, Equation (1) can be simply updated by replacing d_j in the equation of the ith scenario by $c_{i,j}$. We refer to the integral and fractional versions of this general case as INT-SSBO and FRAC-SSBO, respectively; note that the UNIFORM-SSBO problems are obtained from the corresponding SSBO problems by setting $\kappa = 1$.

2.3 Multi Slot Model

In the multi-slot case there are $s \geq 1$ *slots* for each keyword with the *generalized second price auction* for these slots. Let $d_{j,k}$ be an integer denoting the value of the *basic* cost-per-click the kth slot of the jth keyword; we assume $d_{j,1} \leq d_{j,2} \leq \cdots \leq d_{j,s}$. Let $c_{i,j,k} \in [d_{j,k}, \kappa d_{j,k})$ denote the value of the *real* cost-per-click for the kth slot of the jth keyword in the ith scenario where κ is the stretch parameter as in Section 2.2, and let $B > 0$ denote the *budget* (a positive integer) for the advertiser. Our goal is now to compute a set of sn selection variables $x_{j,k}$ where the selection variable $x_{j,k}$ corresponds to kth slot for the jth keyword. We again have a collection of m scenarios where the ith scenario is characterized via:

- a probability ε_i ($\sum_{i=1}^{m} \varepsilon_i = 1$), and
- a "click vector" $(a_{i,j,1}, a_{i,j,2}, \ldots, a_{i,j,s})$ where each $a_{i,j,k}$ is a non-negative integer denoting the number of clicks obtained by the kth slot of the jth keyword $\mathcal{K}_j$ in the ith scenario.

The goal is to compute the allocation variables $x_{j,k}$'s with the constraints

$$\forall\, j:\ \sum_{k=1}^{s} x_{j,k} \leq 1 \tag{2}$$

to maximize the total expected payoff

$$\mathbb{E}[\text{payoff}] = \sum_{i=1}^{m} \mathbb{E}[\text{payoff}_i]$$

where

$$\mathbb{E}[\text{payoff}_i] = \begin{cases} \varepsilon_i \sum_j \sum_k a_{i,j,k} x_{j,k}, & \text{if } \sum_j \sum_k a_{i,j,k} c_{i,j,k} x_{j,k} \leq B \\ \dfrac{B}{\sum_j \sum_k a_{i,j,k} c_{i,j,k} x_{j,k}} \left(\varepsilon_i \sum_j \sum_k a_{i,j,k} x_{j,k} \right), & \text{otherwise} \end{cases} \tag{3}$$

We again distinguish between two versions of the problem:

Integral version (INT-MULTI-SSBO): $x_{j,k} \in \{0,1\}$ for all j and k. Here, $x_{j,k} = 1$ if the advertiser selects the kth slot for the jth keyword, and $x_{j,k} = 0$ otherwise.

[8] Throughout the paper, the notation $\mathsf{poly}(a)$ denotes a polynomial in a, *i.e.*, a^c for some positive constant c.

[9] For example, the stretch parameter κ allows us to model situations such as when the real costs can be drawn from a probability distribution with a mean around $\frac{1+\kappa}{2} d_j$ with a negligible probability of occurring outside a range of $\pm\frac{1-\kappa}{2} d_j$ of the mean. Note that this is just an illustration. We do not assume any specific probability distribution for the variations of the real costs per click except that it varies within an interval of length κ.

Fractional version (FRAC-MULTI-SSBO)**:** $0 \leq x_{j,k} \leq 1$ for all j and k. Here, $x_{j,k}$ denotes the probability that the advertiser selects the kth slot for the jth keyword and $1 - (\sum_{k=1}^{s} x_{j,k})$ is the probability with which the advertiser does not bid on the jth keyword at all.

Note that the scenario model for multi-slot stochastic budget optimization is *quite different* in nature from the other multi-slot models such as the one discussed in [11] since, for example, one can go under or over the budget in one scenario to get a better overall expected payoff.

2.4 Relevance and Significance of Scenario Models

Scenario models are a popular way of modeling optimization problems involving uncertainties in parameters by creating a number of scenarios that depict the probability distribution of various possibilities and then provide a solution that optimizes the expectations of outcomes over these scenarios. The scenario model is important for at least two reasons as explained in [19], which we state below. Firstly, market analysts often think of uncertainty by *explicitly* creating a set of a few model scenarios, possibly attaching a weight to each scenario. Secondly, the scenario model gives us an important tool into understanding the fully general problem with arbitrary joint distributions. Allowing the full generality of an arbitrary joint distribution gives us significant modeling power, but poses challenges to the algorithm designer. Since a naive explicit representation of the joint distribution requires space exponential in the number of random variables, one often represents the distribution implicitly by a sampling oracle. A common technique, Sampled Average Approximation, is to replace the true distribution by a uniform or non-uniform distribution over a set of samples drawn by some process from the sampling oracle, effectively reducing the problem to the scenario model. In addition to their usual applications in operations research (*e.g.*, see [9]), this approach is getting more and more attention in Wall Street as financial portfolios are being created in this way (*e.g.*, see [22]). For example, Cocco, Consiglio and Zenios in [8] developed a scenario-based optimization model for asset and liability management of participating insurance policies with minimum guarantees and Mausser and Rosen in [15] developed three scenario optimization models for portfolio credit risk.

In sponsored search, this is an appropriate model and embodies the "best response" strategy. There is a complex function that maps the state of the world and the users to the queries they pose and their actions such as whether they click on ads. The search engines give a limited amount of information to help advertisers[10], and advertisers can learn various scenarios that determine their click vs cost behaviors to some extent by running experiments, analyzing their web traffics *etc.* However, sponsored search products only provide a limited bidding language to structure one's campaign[11] and hence, necessarily, most advertisers have to target different scenarios simultaneously with each bidding choice. This is the stochastic budget optimization problem we study in this paper. One natural idea is for advertisers to recognize in real time the particular scenario one faces and then apply the best bidding for that scenario. However, this is difficult to do in practice because of limited and delayed information in the system, and it is also expensive to implement. Furthermore, scenario models provide us with an important first step into understanding the fully general problem with arbitrary joint distributions that might be hard to model and analyze since, for example, naive explicit representation of a joint distribution may require space that is exponential in the number of random variables. Instead, techniques such as Sampled Average Approximation explained in the preceding paragraph are used, effectively reducing the problem to

[10]For example, see `https://adwords.google.com/select/TrafficEstimatorSandbox`

[11]See for example, `http://algo.research.googlepages.com/ec09-partI.pdf`

the scenario model. Thus, stochastic budget optimization problems under the scenario model are very appropriate for sponsored search applications.

We do acknowledge that other strategies besides the "best response" may be used by advertisers in practice[12], and stochastic budget optimization algorithms proposed here are not currently used within the practical tools that are publicly available. Nevertheless, best response is a reasonable strategy (even recommended by some search engines), and indeed many anecdotal conversations with advertisers and sponsored search optimizers have clearly indicated to us that they would like to bid to balance across myriad of scenarios. Our algorithms in this paper (even the dynamic programming based ones) can be *easily* implemented in current systems.

2.5 Notational Remarks

As the reader may have already observed, precise definitions of the various models involve a lot of variables and subscripts. To make the exposition clearer, we will therefore adopt the following conventions:

- For variables involving keywords, scenarios and (for the multi-slot model) slots, we will use subscripts i, j and k (and their obvious variations such as i_1, i', *etc.*) for scenarios, keywords and slots, respectively.

- Variables such as m, n, $\mathcal{K}_j$, d_j, B, ε_i, $a_{i,j}$, $a_{i,j,k}$, $c_{i,j}$, $c_{i,j,k}$, x_j, $x_{j,k}$, payoff, payoff$_i$, κ, s and B, when used in the context of the stochastic budget optimization models, will be used for their intended meanings as described in Sections 2.1—2.3.

- Note that:
  - m, n, d_j, B, $a_{i,j}$, $a_{i,j,k}$, $c_{i,j}$, $c_{i,j,k}$ and s are positive integers;
  - $0 \leq \varepsilon_i \leq 1$ and $\sum_{i=1}^{m} \varepsilon_i = 1$;
  - $1 \leq \kappa = O\left(\mathsf{poly}(\log(m+n))\right)$ is an integer. We refer to this in the sequel by the phrase "κ is a small integer".

- The size of an input instance of our SBO problems, which we will denote by size-of-input and which is *crucial in differentiating polynomial-time algorithms from pseudo-polynomial-time algorithms*, is as follows:
  - For INT-SSBO and FRAC-SSBO:
$$\mathsf{size\text{-}of\text{-}input} = \mathsf{poly}\left(m + n + \left(\max_{\substack{1\leq i\leq m \\ 1\leq j\leq n}} \log_2 a_{i,j} \right) + \left(\max_{\substack{1\leq i\leq m \\ 1\leq j\leq n}} \log_2 c_{i,j} \right) + \left(\max_{1\leq i\leq m} \frac{1}{\varepsilon_i} \right) \right).$$
  - For INT-MULTI-SSBO and FRAC-MULTI-SSBO,
$$\mathsf{size\text{-}of\text{-}input} = \mathsf{poly}\left(s + m + n + \left(\max_{\substack{1\leq i\leq m \\ 1\leq j\leq n \\ 1\leq k\leq s}} \log_2 a_{i,j,k} \right) + \left(\max_{\substack{1\leq i\leq m \\ 1\leq j\leq n \\ 1\leq k\leq s}} \log_2 c_{i,j,k} \right) + \left(\max_{1\leq i\leq m} \frac{1}{\varepsilon_i} \right) \right).$$

On rare occasions, if we need to reuse the above-mentioned indices or variables and thus deviate from these conventions, the accompanying text will make the deviation clear.

[12]By other strategies, we mean strategies in which the advertiser does not fix the strategies of other advertisers.

3 Summary of Results and Proof Techniques

[19] left the computational complexity issues of the scenario model as the main open problem after showing that both the integral and fractional versions of this problem, even for single slot case, are NP-hard and noting that *no* non-trivial approximability results are known. While prior results for (S)BO problems exploit insights from the Knapsack problem to associate some potential payoff with each keyword, a central difficulty encountered in directly applying those techniques for our models is that payoff from a keyword can be very different from one scenario to another.

3.1 Summary of Results

We provide a slightly coarse summary of the results obtained in this paper; precise bounds are available in the corresponding technical section that proves the result.

Main Results

(R1) (Approximation algorithms): We provide algorithms that run in *near-linear* time and achieve the following approximation ratios[13]:

- $\min\{O(m), O(\kappa \log d_n)\}$-approximation for both Int-Ssbo and Frac-Ssbo and,
- $\min\left\{O(m), O\left(s\,\kappa \log\Delta \log^2(m+n)\right)\right\}$-approximation for Int-Multi-Ssbo and Frac-Multi-Ssbo, where $\Delta = \max_{j,k} d_{j,k}$.

(R2) (Approximation hardness for the single and multi slot cases) We show that, unless $\mathsf{ZPP} = \mathsf{NP}$, there exist instances of Int-Ssbo and Frac-Ssbo, with n keywords and $m = n$ scenarios each with equal probability, such that *any* polynomial-time algorithm for solving these problems must have an approximation ratio of any one of the following (for any constant $0 < \varepsilon < 1$):

- $\Omega\left(m^{1-\varepsilon}\right)$ (and, thus, also $\Omega\left(n^{1-\varepsilon}\right)$), or
- $\Omega\left(\kappa \log^{1-\varepsilon} d_n\right)$.

This *almost matches* the upper bounds in **(R1)**. Thus, we cannot in general improve the approximation bound in **(R1)**.

Since Ssbo problems are special case of Multi-Ssbo problems for $s = 1$, the approximation hardness bounds for Ssbo can be extended to Multi-Ssbo, providing lower bounds of the form $\Omega\left(m^{1-\varepsilon}\right)$, $\Omega\left(n^{1-\varepsilon}\right)$, or $\Omega\left(\log\kappa \cdot \log^{1-\varepsilon} d_n\right)$ for Multi-Ssbo instances with n keywords, $m = n$ scenarios and s slots. We also show that Int-Multi-Ssbo is $\mathsf{MAX\text{-}SNP}$-hard for $s = 2$ even when $\kappa = 1$ and $c_{j,k} = 1$ for all j and k.

Other Results

In addition to the main results, we also prove a number of other results dealing with variations and special cases of our problems.

Fixed parameter tractability issues: For certain parameter ranges of practical interest we show that these optimization problems can be solved efficiently. If m or ns is fixed, Frac-Multi-Ssbo has a polynomial time solution with an absolute error of δ for any fixed $\delta > 0$. If

[13] The reader is reminded that $\kappa = O\left(\mathsf{poly}(\log(m+n))\right)$.

additionally bids are polynomial in size, Int-Multi-Ssbo also has a polynomial time solution with an absolute error of δ for any fixed $\delta > 0$.

Limitations of semi-definite programming based approaches: The lower bounds in **(R2)** have $\varepsilon < 1$ and thus leave a "very small" gap between this lower bound and the upper bounds described in **(R1)**. It is natural to ask if the gap could be eliminated; for example can we design an approximation algorithm for the special case for $\kappa = 1$ whose approximation ratio is, say, $o\left(\frac{m}{\log m}\right)$ or $o\left(\frac{\log d_n}{\log\log d_n}\right)$? Although we are unable to provide a concrete proof that such a polynomial time approximation algorithm does not exist, we nonetheless observe that the natural semidefinite programming relaxation will not work since it has a large integrality gap of $\frac{m}{2} = \Theta\left(\frac{\log d_n}{\log\log d_n}\right)$.

Dual of Ssbo problems: Finally, in some cases, the *dual* of the stochastic budget optimization problem may be of interest, where we are given a target expected number of clicks and the goal is to minimize the expected budget spent while reaching the target. We present some exact and approximate results for this dual version of the problem.

3.2 Brief Overview of Proof Techniques

In general, budget optimization problems are akin to knapsack problems[14]. But the stochastic budget optimization problems studied in this paper are different because their budgets are "soft", *i.e.*, they can be exceeded, if under a suitable scaling they meet the budget constraint, and this improves the objective function. The stochastic budget optimization problems can be more insightfully thought of as special bipartite quadratic programs (these with ± 1 variables correspond to Grothendieck's inequality with a nice history, but we have 0/1 variables). Standard approaches to solving other special cases of quadratic programs, for example, using relaxations via semi-definite programming, do not provably work as we show. Instead, for upper bounds, we take alternative combinatorial approaches. For showing hardness results, we use intuitions from connections of our problems to these quadratic programs. For one proof, we show reduction from the hard instances of the *maximum independent set* problem [14] on graphs to the bipartite 0/1 quadratic integer programming reformulations of Frac-Ssbo and Int-Ssbo. While anecdotally one may indeed believe these problems to be computationally hard, our results show that this is not true for many ranges of parameters of interest, but do identify the parameter settings that make them computationally hard. Taken together, our results are the *first known* non-trivial complexity results for stochastic budget optimization problems under the scenario model beyond NP-hardness.

4 Sbo Problems and Bipartite Quadratic Integer Programs

In this section we show how to reformulate various Sbo problems as *bipartite quadratic integer programs* (QIP). These reformulations are heavily used in later proofs in the paper. A bipartite quadratic program is a quadratic program in which there is a bipartition of variables such that every term involves at most one variable from each partition. A well-known example of such a (strict) quadratic program on variables taking ± 1 values is the so-called Grothendieck's inequality [1]. However, as will show later, our quadratic program differs significantly in nature from this inequality.

[14] See for example, `http://algo.research.googlepages.com/ec09_pub.pdf`

4.1 Ssbo and QIP

(* Quadratic program (Q1) *)

(* $w_{i,j} = y_{i,j}\, c_{i,j}$ for all i and j *)

$$\text{maximize} \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_i\, x_j\, y_{i,j}$$

subject to

$$\forall\, 1 \le i \le m\colon\ \alpha_i \left(\sum_{j=1}^{n} w_{i,j} x_j \right) \le B_i$$

$$\forall\, 1 \le i \le m\colon\ 0 \le \alpha_i \le 1$$

$$\forall\, 1 \le j \le n\colon\ 0 \le x_j \le 1$$

(* Quadratic program (Q2) *)

(* $w_{i,j,k} = y_{i,j,k}\, c_{i,j,k}$ for all i, j and k *)

$$\text{maximize} \sum_{i=1}^{m} \alpha_i \left(\sum_{j=1}^{n}\sum_{k=1}^{s} x_{j,k}\, y_{i,j,k} \right)$$

subject to

$$\forall\, 1 \le i \le m\colon\ \alpha_i \left(\sum_{j=1}^{n}\sum_{k=1}^{s} w_{i,j,k}\, x_{j,k} \right) \le B_i$$

$$\forall\, 1 \le j \le n\colon\ \textstyle\sum_{k=1}^{s} x_{j,k} \le 1$$

$$\forall\, 1 \le i \le m\colon\ 0 \le \alpha_i \le 1$$

$$\forall\, 1 \le j \le n\ \forall\, 1 \le k \le s\colon\ 0 \le x_{j,k} \le 1$$

Figure 1: Quadratic Integer Programs for Sbo problems. Y is a matrix with non-negative entries ($y_{i,j}$ for **(Q1)** and $y_{i,j,k}$ for **(Q2)**) and $B_1, B_2, \ldots, B_m$ are positive real numbers.

We show how to reformulate Ssbo as a bipartite quadratic integer program. Consider the quadratic program **(Q1)** in Fig. 1. By "integral version" of **(Q1)** we refer to replacing the constraint $0 \le x_i \le 1$ by $x_i \in \{0, 1\}$.

Proposition 1. *The quadratic program* **(Q1)** *and its integral version are equivalent to* Int-Ssbo *or* Frac-Ssbo*, respectively.*

Proof. Consider an instance of Ssbo. Let $y_{i,j} = \varepsilon_i\, a_{i,j}$, $w_{i,j} = c_{i,j}\, y_{i,j}$ and $B_i = \varepsilon_i\, B$. Then, the inequality $\sum_{j=1}^{n} a_{i,j} c_{i,j} x_j \le B$ becomes $\sum_{j=1}^{n} y_{i,j} c_{i,j} x_j \le B_i \equiv \sum_{j=1}^{n} w_{i,j} x_j \le B_i$ and the fraction $\frac{B}{\sum_{j=1}^{n} a_{i,j} c_{i,j} x_j}$ becomes $\frac{B_i}{\sum_{j=1}^{n} w_{i,j} x_j}$. Conversely, given an instance of **(Q1)**, let $B = \sum_{i=1}^{m} B_i$, $\varepsilon_i = \frac{B_i}{B}$ and $a_{i,j} = \frac{y_{i,j}}{\varepsilon_i}$. Thus, $\varepsilon_i\, a_{i,j} = y_{i,j}$, the inequality $\sum_{j=1}^{n} w_{i,j} x_j \le B_i \equiv \sum_{j=1}^{n} y_{i,j} c_{i,j} x_j \le B_i$ is the same as $\sum_{j=1}^{n} a_{i,j} c_{i,j} x_j \le B$ and the fraction $\frac{B_i}{\sum_{j=1}^{n} w_{i,j} x_j}$ is the same as $\frac{B}{\sum_{j=1}^{n} a_{i,j} c_{i,j} x_j}$. Thus, in the sequel, we assume such a correspondence.

Now, consider a solution vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ for **(Q1)**. Then $\mathbf{x}$ also defines a solution vector for Ssbo. We must verify that this is indeed a valid solution vector with a correct expected payoff. Let $Q_i = \sum_{j=1}^{n} w_{i,j} x_j$. If $\alpha_i Q_i < B_i$ then $\alpha_i = 1$ since otherwise the solution for **(Q1)** can be further improved, and then $\mathbb{E}[\text{payoff}_i] = \sum_{j=1}^{n} y_{i,j} x_j$, which is correct. Otherwise $\alpha_i Q_i = B_i$ and then $\mathbb{E}[\text{payoff}_i] = \frac{B_i}{B_i/\alpha_i} \sum_{j=1}^{n} y_{i,j} x_j = \alpha_i \sum_{j=1}^{n} y_{i,j} x_j$, which is also correct. This shows that for every instance of **(Q1)** there is a corresponding instance of Ssbo with the same expected payoff.

Now, consider a solution vector $\mathbf{x}$ for Ssbo. Then, if $Q_i \ge B_i$ then $\alpha_i = B_i/Q_i$ otherwise $\alpha_i = 1$. It is easy to see in the same manner that this provides a valid solution of **(Q1)** with the same objective value. $\square$

Relationship to the Standard Knapsack Problems If $m = \kappa = 1$ and α_1 is set to a fixed constant, then **(Q1)** reduces a special linear program which is equivalent to the so-called (fractional) knapsack problem which is well-studied in the literature. Extending this analogy, by the phrase "the standard fractional knapsack problem corresponding to the ith row of and pth through qth column of Y", we will mean the linear program as shown in Fig. 2 (it is easy to see that there is an optimal solution of this linear program in which $\alpha_i = \min\left\{1, \frac{B_i}{\left(\sum_{j=p}^{q} c_{i,j}\, y_{i,j}\right)}\right\}$). Since $d_p \leq d_{p+1} \leq \cdots \leq d_q$, the following well-known fact follows.

maximize $\alpha_i \sum_{j=p}^{q} y_{i,j}\, x_j$
subject to $\alpha_i \left(\sum_{j=p}^{q} c_{i,j}\, y_{i,j}\, x_j\right) \leq B_i$
$\forall\, p \leq j \leq q\colon\ 0 \leq x_j \leq 1$

Figure 2: LP for ith row and pth through qth column of Y.

Fact 1. [12] *An optimal solution to the linear program in Fig. 2 ("optimal payoff for the ith row and pth through qth column of Y") is a "prefix solution",* i.e., *there is an index j' such that $x_j = 1$ for $j < j'$, $0 < x_{j'} \leq 1$ and $x_j = 0$ for $j > j'$.*

4.2 Multi-Ssbo and QIP

The quadratic programming reformulation of Multi-Ssbo can also be obtained in a similar manner and is shown as **(Q2)** in Fig. 1.

5 Poly-logarithmic Approximations for Ssbo and Multi-Ssbo (main result (R1))

Theorem 1 (Near-linear time approximation)**.** *There is a*

(i) $\min\{O(m), O(\kappa \log d_n)\}$*-approximation for both* Int-Ssbo *and* Frac-Ssbo*;*

(ii) $\min\{O(m), O(s\kappa \log \Delta)\}$*-approximation for* Frac-Multi-Ssbo *and*

(iii) $\min\left\{O(m), O\left(s\kappa \log \Delta \log^2(m+n)\right)\right\}$*-approximation for* Int-Multi-Ssbo

where, for **(ii)** *and* **(iii)**, $\Delta = \max_{j,k} d_{j,k}$*. All these algorithms can be implemented in linear or near-linear time using standard data structures and algorithmic techniques.*

In the rest of this section, we prove the above theorem. As a first attempt, one might be tempted to use recent techniques in designing efficient algorithms for multiple-knapsack problems [7, 16] for our problem; however it is not difficult to design examples where such approaches fail badly since our budget constraints are "soft" (they can be exceeded if scaling them gives better payoff) and our probabilities are "arbitrary". As a second attempt, one might take our quadratic programming reformulation as discussed in Section 4 and semidefinite-programming based rounding approach such as in [13]. However, it can be shown that the integrality gap of such a reformulation is very large. The failure of these natural approaches shows the difficulty of the problems. Thus, we are led to explore other combinatorial approaches to provide the desired approximation.

5.1 $O(m)$-approximation for Int-Ssbo and Frac-Ssbo

To get a $O(m)$-approximation we can do the following. For each i we solve the standard (integer or fractional) knapsack problem for the ith row of Y; let p_i be the value of an optimal solution. Then, take the best of these solutions, say of value $p = \max_{1 \leq i \leq m}\{p_i\}$. Each fractional knapsack

1.	Partition the keywords into *maximal* groups such that if a group G contains pth through qth keyword then $d_q/d_p \leq 2$ and $d_{q+1}/d_p > 2$. Let $\mathcal{G}$ be the set of such groups.
2.	**For** each group $G \in \mathcal{G}$ consisting of keywords, say $\mathcal{K}_p, \mathcal{K}_{p+1}, \ldots, \mathcal{K}_q$, **do** Set $x_j = 1$ for every $p \leq j \leq q$ and set $x_j = 0$ for all other j; let $\mathbb{E}[\text{payoff}']$ be the payoff of this solution
3.	Output the best of the solutions obtained in **2**.

Figure 3: Algorithm for the case of $\kappa = 1$.

problem can be solved exactly in $O(n \log n)$ time [12] and a $O(n \log n)$ time greedy 2-approximation algorithm for the integer knapsack problem is also well known [17].

We now note that $\mathbb{E}[\text{payoff}] \leq \sum_{i=1}^{m} p_i$. Indeed, consider an optimal solution of Ssbo. If $\alpha_i = 1$, then by definition of p_i we have $\mathbb{E}[\text{payoff}_i] \leq p_i$. If $\alpha_i < 1$, then we set $\alpha_i = 1$ and set a new value of x_j as $x'_j = \alpha_i x_j$. This does not change $\mathbb{E}[\text{payoff}_i]$ and now we again have $\mathbb{E}[\text{payoff}_i] \leq p_i$. Thus, we have $p \geq \mathbb{E}[\text{payoff}]/m$.

If $p = p_i$ for some i, then the solution of the knapsack problem of value p can be extended to a solution of Ssbo by setting $\alpha_{i'} = 0$ for $i' \neq i$.

5.2 $O(\kappa \log d_n)$-approximation for Int-Ssbo and Frac-Ssbo

Case of $\kappa = 1$: Uniform Cost Model

The algorithm is shown in Fig. 3. Consider a group $G \in \mathcal{G}$ consisting of the keywords $\mathcal{K}_p, \mathcal{K}_{p+1}, \ldots, \mathcal{K}_q$. By the "Ssbo problem on G" we mean the instance of the Ssbo problem in which our click input consists of the submatrix $Y_{p,q} = \begin{pmatrix} y_{1,p} & y_{1,p+1} & \cdots & y_{1,q} \\ \vdots & \vdots & \cdots & \vdots \\ y_{m,p} & y_{m,p+1} & \cdots & y_{m,q} \end{pmatrix}$ of Y containing all rows and pth through qth columns, the costs-per-click $d_p, \ldots, d_q$, the budgets $B_1, \ldots, B_m$, and the selection variables $x_p, \ldots, x_q$. Let $\mathbb{E}[\text{payoff}_G]$ be the value of expected payoff of an optimal solution for this subproblem. Since $\max_{G \in \mathcal{G}} \mathbb{E}[\text{payoff}_G] \geq \frac{\mathbb{E}[\text{payoff}]}{|\mathcal{G}|}$ and $|\mathcal{G}| = O(\log d_n)$, the following lemma proves the desired approximation bound.

Lemma 2. $\mathbb{E}[\text{payoff}'] \geq \dfrac{\mathbb{E}[\text{payoff}_G]}{2}$.

Proof. We only need to prove the lemma for the case when $\mathbb{E}[\text{payoff}_G]$ is the total expected payoff of an optimal solution of the Frac-Ssbo problem on G since obviously the total expected payoff of an optimal solution of the Int-Ssbo problem on G is no more than $\mathbb{E}[\text{payoff}_G]$. Let $D = \sum_{j=p}^{q} d_j \, y_{i,j}$ and $\beta = |G|$. By our choice of the group G,

$$d_p \sum_{j=p}^{q} y_{i,j} \leq D \leq d_q \sum_{j=p}^{q} y_{i,j} \leq 2 \, d_p \sum_{j=p}^{q} y_{i,j}.$$

Using the quadratic programming formulation **(Q1)** and remembering that $c_{i,j} = d_j$ when $\kappa = 1$, the Frac-Ssbo instance on G is equivalent to the following quadratic program **(Q3)**:

(* Quadratic program (Q3) *)

maximize $\sum_{i=1}^{m} \alpha_i \left(\sum_{j=p}^{q} y_{i,j}\, x_j \right)$

subject to $\forall\, 1 \leq i \leq m\colon\ \alpha_i \left(\sum_{j=p}^{q} d_j\, y_{i,j}\, x_j \right) \leq B_i$

$\forall\, 1 \leq i \leq m\colon\ 0 \leq \alpha_i \leq 1$

$\forall\, p \leq j \leq q\colon\ 0 \leq x_j \leq 1$

Fix any optimal solution for our Frac-Ssbo instance on G, *i.e.*, fix an optimal solution vector $(\alpha_1^*, \alpha_2^*, \ldots, \alpha_m^*)$ and $(x_p^*, x_{p+1}^*, \ldots, x_q^*)$ of **(Q3)**. In our solution sets $x_p = x_{p+1} = \cdots = x_q = 1$; thus $\alpha_i = \min \left\{ 1,\ \frac{B_i}{D} \right\}$ for every i and $x_j \geq x_j^*$ for every $p \leq j \leq q$.

Case 1: $D \leq B_i$. Then, $\alpha_i = 1 \geq \alpha_i^*$, $x_j = 1 \geq x_j^*$ for $p \leq j \leq q$, and thus

$$\alpha_i \left(\sum_{j=p}^{q} y_{i,j}\, x_j \right) \geq \alpha_i^* \left(\sum_{j=p}^{q} y_{i,j}\, x_j^* \right).$$

Case 2: $D > B_i$. Then, $\alpha_i = \frac{B_i}{D}$. Now, we have

$$\alpha_i \left(\sum_{j=p}^{q} y_{i,j}\, x_j \right) = \left(\frac{B_i}{D} \right) \times \sum_{j=p}^{q} y_{i,j} \geq \left(\frac{B_i}{D} \right) \times \left(\frac{\sum_{j=p}^{q} d_j\, y_{i,j}}{d_q} \right) = \frac{B_i}{d_q} \geq \frac{1}{2} \times \frac{B_i}{d_p}$$

$$\alpha_i^* \left(\sum_{j=p}^{q} y_{i,j} x_j^* \right) \leq \frac{B_i}{\sum_{j=p}^{q} y_{i,j} d_j x_j^*} \times \left(\sum_{j=p}^{q} y_{i,j} x_j^* \right) \leq \frac{B_i}{d_p}$$

where the inequality for α_i^* comes directly from the constraints of **(Q3)**.

Thus, combining both cases, we have

$$\mathbb{E}[\text{payoff}'] = \sum_{i=1}^{m} \alpha_i \left(\sum_{j=p}^{q} y_{i,j}\, x_j \right) \geq \frac{1}{2} \times \sum_{i=1}^{m} \alpha_i^* \left(\sum_{j=p}^{q} y_{i,j}\, x_j^* \right) = \frac{\mathbb{E}[\text{payoff}_G]}{2}$$

□

Case of $\kappa > 1$: General Single-slot Model

Using our δ-approximation algorithm for Uniform-Ssbo (for $\delta = O(\log d_n)$) as outlined in Fig. 3, we show how to use it as a subroutine to get a $\kappa\,\delta = O(\kappa \log d_n)$-approximation for Int-Ssbo (and, hence, also for Frac-Ssbo). The algorithm is shown in Fig. 4.

1. Replace (truncate) each $c_{i,j}$ by its *new* value $c'_{i,j} = d_j$.

2. Use the approximation algorithm in Fig. 3 with these new truncated values of $c_{i,j}$'s. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ be the solution vectors returned.

3. Output $\mathbf{x}$ and $\boldsymbol{\alpha}' = (\alpha_1', \alpha_2', \ldots, \alpha_m') = \left(\frac{\alpha_1}{\kappa}, \frac{\alpha_2}{\kappa}, \ldots, \frac{\alpha_m}{\kappa} \right)$ as our solution.

Figure 4: $O(\kappa \log d_n)$-approximation algorithm for Int-Ssbo.

We use the following notations:

- $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ and $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_m^*)$ are the solution vectors for an optimal solution of our (original) instance of Ssbo, and $\mathbb{E}[\text{payoff}^*] = \sum_{i=1}^m \alpha_i^* \left(\sum_{j=1}^n y_{i,j} x_j^*\right)$ is the total expected payoff of this optimal solution.
- $\mathbf{x}^+ = (x_1^+, x_2^+, \ldots, x_n^+)$ and $\boldsymbol{\alpha}^+ = (\alpha_1^+, \alpha_2^+, \ldots, \alpha_m^+)$ are the solution vectors for an optimal solution of the truncated instance of Ssbo, and $\mathbb{E}[\text{payoff}^+] = \sum_{i=1}^m \alpha_i^+ \left(\sum_{j=1}^n y_{i,j} x_j^+\right)$ is the total expected payoff of this optimal solution.
- $\mathbb{E}[\text{payoff}] = \sum_{i=1}^m \alpha_i' \left(\sum_{j=1}^n y_{i,j} x_j\right)$ is the total expected payoff of the solution obtained by using the algorithm in Fig. 4.

Proposition 2. *The following statements are true:*

(a) $\mathbf{x}$ *and* $\boldsymbol{\alpha}'$ *correspond to a valid solution of the* Ssbo *instance.*

(b) $\mathbb{E}[\text{payoff}^+] \geq \mathbb{E}[\text{payoff}^*]$.

(c) $\mathbb{E}[\text{payoff}] \geq \frac{\mathbb{E}[\text{payoff}^+]}{\kappa}$.

Thus the algorithm in Fig. 4 is a $O(\kappa \log d_n)$*-approximation.*

Proof.

(a) $\alpha_i' c_{i,j} = \frac{\alpha_i}{\kappa} c_{i,j} \leq \frac{\alpha_i}{\kappa} \kappa\, d_j = \alpha_i d_j$, thus $\alpha_i \left(\sum_{j=1}^n y_{i,j}\, d_j\, x_j\right) \leq B_i$ implies $\alpha_i' \left(\sum_{j=1}^n y_{i,j}\, c_{i,j}\, x_j\right) \leq B_i$.

(b) The solution vectors $\mathbf{x}^*$ and $\boldsymbol{\alpha}^*$ for an optimal solution of the Ssbo instance is also a valid (not necessarily optimal) solution vector for the truncated instance of Ssbo since $c_{i,j}' \leq c_{i,j}$.

(c) This follows since $\alpha_i' = \frac{\alpha_i}{\kappa}$. □

5.3 Approximation Bounds for Frac-Multi-Ssbo and Int-Multi-Ssbo

To get a $O(m)$-approximation we follow the same approach as in Section 5.1. For each i we solve the restriction of the Multi-Ssbo problem on the ith scenario, *i.e.*, the quadratic program **(Q4)** as shown in Fig. 5, and then take the best of these solutions. It is easy to see that an optimal solution of **(Q4)** satisfies $\alpha_i = \min\left\{1, \frac{B_i}{\sum_{j=1}^n \sum_{k=1}^s w_{i,j,k} x_{j,k}}\right\}$. For any fixed value of α_i, **(Q4)** is known in the literature as the *multiple-choice* Knapsack problem with $s\,n$ objects divided into n classes and a knapsack capacity of B_i/α_i; a $O(1)$-approximation algorithm for this problem that runs in $O\left(ns^2\right)$ time is known [17].

(* Quadratic program (Q4) *)

maximize $\alpha_i \left(\sum_{j=1}^n \sum_{k=1}^s y_{i,j,k}\, x_{j,k}\right)$

subject to

$$\alpha_i \left(\sum_{j=1}^n \sum_{k=1}^s w_{i,j,k}\, x_{j,k}\right) \leq B_i$$
$$\forall\, 1 \leq j \leq n:\ \ \sum_{k=1}^s x_{j,k} \leq 1$$
$$0 \leq \alpha_i \leq 1$$
$$\forall\, 1 \leq j \leq n\ \forall\, 1 \leq k \leq s:\ \ 0 \leq x_{j,k} \leq 1$$

Figure 5: Multi-Ssbo restricted to the ith scenario.

We next show that algorithms for the single-slot case can be used for the multi-slot model with appropriate multiplicative factors in the approximation ratio.

Lemma 3. *There exists a* $O(s\,\kappa \log \Delta)$*-approximation (respectively,* $O\left(s \log^2(m+n)\,\kappa \log \Delta\right)$*-approximation) algorithm for* Frac-Multi-Ssbo *(respectively,* Int-Multi-Ssbo*).*

Proof. We first prove our claim for FRAC-MULTI-SSBO. Consider the quadratic program **(Q2)'** obtained from the quadratic program **(Q2)** for FRAC-MULTI-SSBO by removing the constraints $\sum_{k=1}^{s} x_{j,k} \leq 1$ for $1 \leq j \leq n$. If OPT and OPT' are the optimal values of the objective functions of **(Q2)** and **(Q2)'**, respectively, then obviously $\mathsf{OPT}' \geq \mathsf{OPT}$. A straightforward inspection shows that **(Q2)'** can be written down in the same form as **(Q1)** with $s\,n$ variables and m constraints. Thus, using the already proven result of Theorem 1**(i)** we obtain a solution for **(Q2)'** whose objective value is $\frac{\mathsf{OPT}'}{\kappa \log(\max_{j,k} d_{j,k})} = \frac{\mathsf{OPT}'}{\kappa \log \Delta} \geq \frac{\mathsf{OPT}}{\kappa \log \Delta}$ To convert this to a solution of FRAC-MULTI-SSBO(*i.e.*, to satisfy the constraints $\sum_{k=1}^{s} x_{j,k} \leq 1$ for each j) we divide each $x_{j,k}$ by $\sum_{k=1}^{s} x_{j,k}$ which decreases the total payoff by no more than a factor of s.

The result for INT-MULTI-SSBO follows by translating the above worst-case approximation bound for FRAC-MULTI-SSBO to a worst-case approximation of INT-MULTI-SSBO via the following lemma.

Lemma 4. *(*APPROXIMATING INT-MULTI-SSBO VIA FRAC-MULTI-SSBO*) Suppose that we have a η-approximation for* FRAC-MULTI-SSBO. *Then, we also have a $O(\eta\,\gamma)$ approximation for* INT-MULTI-SSBO *where* $\gamma = \begin{cases} \log m, & \textit{if } s = 1 \\ \log^2(m+n), & \textit{otherwise} \end{cases}$

Proof. For a particular value of the vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$, **(Q2)** reduces to a linear program on the variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. For ease of description, we consider the case of $s = 1$ first (*i.e.*, the case of FRAC-SSBO). An inspection of **(Q1)** reveals that this linear program has exactly n variables and m inequalities, where the *ith* inequality D_i (for $1 \leq i \leq m$) is of the form:

$$D_i \overset{\textbf{def}}{\equiv} \alpha_i \left(\sum_{j=1}^{n} w_{i,j}\, x_j \right) \leq B_i$$

Consider a solution $\mathbf{x}^f = (x_1^f, x_2^f, \ldots, x_n^f)$. and $\boldsymbol{\alpha}^f = (\alpha_1^f, \alpha_2^f, \ldots, \alpha_m^f)$, of FRAC-SSBO with $\mathcal{L} = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i^f\, y_{i,j}\, x_j^f$ as the value of its objective. We may assume that $\mathcal{L} > 100 \ln m$ since otherwise the approximation guarantee can be trivially achieved. We employ the following randomized rounding scheme to transform this solution to a solution of INT-SSBO:

- For $i = 1, 2, \ldots, n$, we round x_i^f randomly to 0 and 1 with probabilities x_i^f and $1 - x_i^f$, respectively. Let $x_i \in \{0, 1\}$ be the resulting random variable.

- We return $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ as our solution where $\alpha_i = \frac{\alpha_i^f}{100 \ln m}$ for $1 \leq i \leq m$.

Let $\mathcal{L}' = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i y_{i,j}\, x_j$ be the new value of the objective and let $\mathcal{E}_i$ be the event that inequality D_i holds for this randomized solution. By linearity of expectation $\mathbb{E}[\mathcal{L}'] = \frac{\mathcal{L}}{100 \ln m}$. Consider the inequality D_i, and let $\alpha_i' = \frac{\alpha_i}{B_i + 1}$. By linearity of expectation,

$$\mathbb{E}\left[\alpha_i' \left(\sum_{j=1}^{n} w_{i,j} x_j \right) \right] = \frac{1}{100 \ln m} \times \frac{1}{B_i + 1} \times \alpha_i^f \left(\sum_{j=1}^{n} w_{i,j} x_j^f \right) < \frac{1}{100 \ln m} \times \frac{B_i}{B_i + 1}$$

Since $\alpha_i' = \frac{\alpha_i}{B_i + 1}$, $0 \leq \alpha_i' w_{i,j} x_j = \frac{\alpha_i w_{i,j} x_j}{B_i + 1} \leq \frac{B_i}{B_i + 1} < 1$ and thus $\alpha_i' w_{i,j} x_j$ can be thought of as an independent Poisson trial whose probability of success (a value of 1) is $\alpha_i' w_{i,j} x_j$ and probability of failure (a value of 0) is $1 - \alpha_i' w_{i,j} x_j$. Thus, using standard Chernoff bound [18, Excercise 4.1], we get:

$$\Pr[\mathcal{E}_i \text{ does not hold}] = \Pr\left[\alpha_i \left(\sum_{j=1}^{n} w_{i,j} x_j\right) > B_i\right] = \Pr\left[\alpha_i' \left(\sum_{j=1}^{n} w_{i,j} x_j\right) > \frac{B_i}{B_i+1}\right] < \mathbf{e}^{-3\ln m} < \frac{1}{m^2}$$

In a similar manner, one can show that $\Pr\left[\mathcal{L}' < \frac{\mathcal{L}}{200\ln m}\right] < \frac{1}{m}$. Thus, finally, using union bounds, we get

$$\Pr\left[\mathcal{L}' \geq \frac{\mathcal{L}}{200\ln m} \bigwedge (\wedge_{i=1}^{m} \mathcal{E}_i \text{ holds})\right] \geq 1 - \Pr\left[\mathcal{L}' < \frac{\mathcal{L}}{200\ln m}\right] - \left(\sum_{i=1}^{m} \Pr\left[\mathcal{E}_i \text{ does not hold}\right]\right) > 1 - \frac{2}{m}$$

Thus, we achieve the desired approximation bound with $1 - o(1)$ probability.

For the case of $s > 1$ (*i.e.*, Frac-Multi-Ssbo), the same approach with some modifications works. In a nutshell, we have n additional constraints F_j (for $j = 1, 2, \ldots, n$) of the form $\sum_{k=1}^{s} x_{j,k} \leq 1$. Thus, the total number of inequalities/equalities is $m+n$ and we need to do the analysis with "$\ln(n+m)$" replacing "$\ln m$". The only additional part that needs to be done is to show how to handle the F_j constraints. Notice that the set of variables involved in F_j are disjoint from the set of variables in any other $F_{j'}$ for $j' \neq j$. After rounding, we have $\sum_{k=1}^{s} x_{j,k} \leq 100\ln(m+n)$. We now select one of these variables x_{j_1} to $x_{j,s}$, say $x_{j,\ell}$, such that $x_{j,\ell} = \max_{1 \leq k \leq s}\{\sum_{i=1}^{m} \alpha_i x_{j,k} y_{i,j,k}\}$, set $x_{j,\ell} = 1$ and set $x_{j,k} = 0$ for $k \neq \ell$. After all these normalizations, we loose an additional factor of $100\ln(m+n)$ and all constraints are satisfied. □

Note that the claim in Lemma 4 is "pessimistic" in nature; indeed, as our claim in Theorem 1 shows, for arbitrary parameter range both Int-Ssbo and Frac-Ssbo can be approximated to within the same ratio. □

6 Approximation-hardness Results for Ssbo and Multi-Ssbo (main result (R2))

6.1 Approximation-hardness Bounds for Ssbo

Theorem 5 (Logarithmic inapproximability). *There exist instances of* Int-Ssbo *and* Frac-Ssbo*, with n keywords and $m = n$ scenarios each with equal probability, such that, unless* ZPP = NP*,* any *polynomial-time algorithm for solving these problems must have an approximation ratio of any one of the following:*

- $\Omega\left(m^{1-\varepsilon}\right)$ *(and, thus, also $\Omega\left(n^{1-\varepsilon}\right)$), or*
- $\Omega\left(\kappa \log^{1-\varepsilon} d_n\right)$.

where $0 < \varepsilon < 1$ is any constant.

Proof. We construct instances of Ssbo with n keywords and $m = n$ scenarios such that, for[15] *any* κ and *any* values of $c_{i,j}$ in the range $[d_j, \kappa\, d_j)$, the claimed lower bound holds. We use the reformulation of Frac-Ssbo and Int-Ssbo as a bipartite quadratic program **(Q2)** as discussed in Section 4.

[15] Remember that in Section 2.5 we fixed bounds on κ, namely, $\kappa = O\left(\mathsf{poly}(\log(m+n))\right)$.

The standard maximum independent set (MIS) problem is defined as follows. We are given an undirected graph $G = (V, E)$. A subset of vertices $V' \subseteq V$ is called *independent* if for *every* two vertices $u, v \in V'$ we have $\{u, v\} \notin E$. The goal is to find an independent subset of vertices of *maximum* cardinality. It is known that MIS cannot be approximated to within a factor of $|V|^{1-\varepsilon}$ for any constant $0 < \varepsilon < 1$ unless ZPP= NP [14].

For notational simplicity, let $n = |V|$ and $a = n^{12}$. Set $m = n$. Select an arbitrary order $v_1, v_2, \ldots, v_n$ of the vertices in V. Intuitively, the ith column and the $(n+1-i)$th row of Y correspond to the vertex v_i and the entries of the matrix Y are such that they are 0 above the reverse diagonal and encodes the adjacency of vertices of G on or below the reverse diagonal. Formally,

$$y_{i,j} = \begin{cases} 0 & \text{if } i+j < n+1 \\ 1 & \text{if } i+j = n+1 \\ 1 & \text{if } i+j > n+1 \text{ and } \{v_{n-i+1}, v_j\} \in E \\ 0 & \text{if } i+j > n+1 \text{ and } \{v_{n-i+1}, v_j\} \notin E \end{cases}$$

Fix $d_1, d_2, \ldots, d_n$ as $d_1 = 1$ and $d_i = a\, d_{i-1}$ for $1 < i \leq n$. Thus, for all sufficiently large n, $\frac{c_{i_1,j_1}}{c_{i_2,j_2}} \geq \frac{\frac{d_{j_1}}{\kappa}}{\kappa\, d_{j_2}} > n^6$ if $j_1 > j_2$. Let $B_i = c_{i,n+1-i}$ for $1 \leq i \leq m = n$. Remembering that $w_{i,j} = c_{i,j}\, y_{i,j}$ for all i and j, we have:

$$w_{i,j} = \begin{cases} 0 & \text{if } i+j < n+1 \\ & \text{or if } i+j > n+1 \text{ and } \{v_{n-i+1}, v_j\} \notin E \\ c_{i,j} & \text{if } i+j = n+1 \\ & \text{or if } i+j > n+1 \text{ and } \{v_{n-i+1}, v_j\} \in E \end{cases}$$

Note that $n^{1-\varepsilon} = m^{1-\varepsilon} = \Omega\left(\kappa \log^{1-\varepsilon'} d_n\right)$, where $0 < \varepsilon' < 1$ is a constant that depends on ε, since $d_n = n^{12\,n}$ and $\kappa = \mathsf{poly}\,(\log(m+n)) = \mathsf{poly}\,(\log(n))$. Let Δ_{ind} and Δ_{Q1} be the maximum number of independent vertices in G and an optimal value of the objective of the fractional or integral version of **(Q1)**, respectively.

Lemma 6. $\Delta_{\text{Q1}} \geq \Delta_{\text{ind}}$.

Proof. Consider an optimal solution V' of MIS on G with $|V'| = \Delta_{\text{ind}}$. We generate a solution of **(Q2)** by setting

$$x_i = \alpha_{n-i+1} = \begin{cases} 1, & \text{if } v_i \in V' \\ 0, & \text{otherwise} \end{cases}$$

Note that, since V' is an independent set, if $i + j > n+1$, $v_i \in V'$ and $\{v_i, v_j\} \in E$ then $v_j \notin V'$ and thus $x_i = \alpha_{n-i+1} = 1$ and $x_j = \alpha_{n-j+1} = 0$.

First, we show that this is indeed a valid solution of **(Q1)**. For any $1 \leq i \leq n-1$, consider the constraint

$$\alpha_{n-i+1} \left(\sum_{j=1}^{n} w_{n-i+1,j}\, x_j \right) \leq B_{n-i+1}.$$

If $\alpha_{n-i+1} = 0$, then the constraint is obviously satisfied since $B_{n-i+1} > 0$. Otherwise, $\alpha_{n-i+1} = x_i = 1$ and thus,

$$\alpha_{n-i+1} \left(\sum_{j=1}^{n} w_{n-i+1,j}\, x_j \right) = \sum_{j=1}^{n} w_{n-i+1,j}\, x_j = c_i + \sum_{\substack{i+j>n+1 \\ \{v_i,v_j\} \in E}} w_{n-i+1,j}\, x_j = c_{n+1-i,i} = B_{n+1-i}$$

Thus, all the constraints *are* satisfied. Finally, the value of the objective function is

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_i\, x_j\, y_{i,j} = \sum_{\substack{i+j=n+1\\ v_j\in V'}}\alpha_i x_j = \sum_{v_j\in V'} x_j = \Delta_{\text{ind}}$$

and thus $\Delta_{\text{Q1}} \geq \Delta_{\text{ind}}$. □

For the other direction, we first need a normalization lemma.

Lemma 7 (Normalization lemma)**.** *Consider an optimal solution of* **(Q1)** *with an objective value of Δ_{Q1}. Then, we can transform this solution to another solution of* **(Q1)** *of objective value Δ'_{Q1} such that:*

(a) $x_i \in \{0,1\}$ *for each* i*;*

(b) $\Delta'_{\text{Q1}} \geq \Delta_{\text{Q1}} - 1$*; and*

(c) *if* $\{x_i, x_j\} \in E$ *then* $x_i + x_j \leq 1$*.*

Proof. Suppose that we are given an optimal solution of **(Q1)** with an objective value of Δ_{Q1}. First, we note some properties of this solution.

Proposition 3. *The following statements are true:*

(i) *for every* i*,* $\alpha_{n-i+1}x_i \leq 1$*, and*

(ii) *for every* i *and* j*, if* $i+j > n+1$ *and* $\{v_i, v_j\} \in E$ *then* $\alpha_j x_i \leq n^{-6}$*.*

Proof. Consider the constraint $\alpha_{n-i+1}\left(\sum_{j=1}^{n} w_{n-i+1,j}x_j\right) \leq B_{n-i+1} = c_{n-i+1,i}$.

Since $w_{n-i+1,i} = c_{n-i+1,i}$, **(i)** follows.

(ii) is equivalent to the claim that $\alpha_{n-i+1}x_j \leq n^{-6}$ if $j > i$. Since $\frac{c_{p,j}}{c_{q,i}} > n^6$ if $j > i$ (for any p and q), **(ii)** follows. □

Now we show how to "normalize" this solution such that each variable x_i is 0 or 1, and the total objective value does not decrease too much. Let $\Gamma = \sum_{i+j\neq n+1}\alpha_i x_j y_{i,j}$. By Proposition 3**(ii)**, $\Gamma \leq n^2 \times n^{-6} = n^{-4}$. Thus, setting $\Phi = \sum_{i+j=n+1}\alpha_i x_j y_{i,j}$, it follows that $\Phi \leq \Delta_{\text{Q1}} \leq \Phi + n^{-4}$. Thus, subsequently we concentrate on the quantity Φ.

If $\alpha_{n-i+1} = 0$ for some i, then we can set $x_i = 0$ without changing the value of Φ. Let $I = \{n-i+1 \,|\, \alpha_{n-i+1} > 0 \text{ and } x_i > 0\}$. Consider the largest index $n-i+1 \in I$. There are two cases to consider:

Case 1: $x_i > n^{-3}$**.** By Proposition 3**(i)**, $\alpha_{n-i+1} < n^{-3}$ and $\alpha_{n-j+1}x_j \leq \alpha_{n-j+1} < n^{-3}$ for every $j > i$ such that $\{v_i, v_j\} \in E$.

We set $\alpha_{n-i+1} = x_i = 1$ and set $x_j = \alpha_{n-j+1} = 0$ for every $j > i$ such that $\{v_i, v_j\} \in E$. The change in Φ is at most $n \times n^{-3} = n^{-2}$.

Case 2: $x_i \leq n^{-3}$**.** We set $\alpha_{n-i+1} = x_i = 0$. The change in Φ is at most n^{-3}.

We now remove the index $n-i+1$ from I and continue with the next largest index. We continue until $I = \emptyset$. Since $|I| \leq n$, the total change in Φ is at most $n^{-1} < 1 - n^{-4}$.

To complete the proof, we select vertices v_j in the independent set if $x_j = 1$. □

To finish the proof of Theorem 5, we simply select those vertices v_i for the independent set such that $x_i = 1$. We have now shown that $\Delta_{\text{ind}} \leq \Delta_{\text{Q1}} \leq \Delta_{\text{ind}} - 1$. Thus, since Δ_{ind} and Δ_{Q1} are within a constant factor of each other and Δ_{ind} cannot be approximated to with a factor of $n^{1-\varepsilon}$ for any constant $0 < \varepsilon < 1$, Δ_{Q1} cannot be approximated to within a factor of $c\,n^{1-\varepsilon}$, or $c\,m^{1-\varepsilon}$, or $c'\,\kappa \log^{1-\varepsilon} d_n$ for some positive constants c and c'. $\square$

6.2 Approximation Hardness Results for Multi-Ssbo

A first natural approach to prove an approximation hardness result for Multi-Ssbo would be to generalize the approximation hardness result for the single-slot case **(Q1)** in Theorem 5 to the multi-slot case **(Q2)**. This can be trivially done by copying the construction of the single-slot case to *one* of the slots in the multi-slot case. However, after this, one can observe that:

> the construction for the single-slot case cannot again be copied to another slot because of the constraints in Equation (2) which state that at most one selection variable in each slot can be set to 1.

Formally, the lower bound construction for **(Q1)** can be extended to **(Q2)** as follows:

- Identify $y_{i,j,1}$ of **(Q2)** with $y_{i,j}$ of **(Q1)** and set $y_{i,j,2} = y_{i,j,3} = \cdots = y_{i,j,s} = 0$ in **(Q2)**.
- Identify $c_{i,j,1}$ of **(Q2)** with $c_{i,j}$ of **(Q1)** and set $c_{i,j,2} = c_{i,j,3} = \cdots = c_{i,j,s} = 0$ in **(Q2)**.
- Identify $x_{j,k,1}$ of **(Q2)** with x_j of **(Q1)**.

This leads to the following approximation hardness result.

Corollary 8. *There exist instances of* Int-Multi-Ssbo *and* Frac-Multi-Ssbo*, with n keywords, $m = n$ scenarios each with equal probability and s slots, such that, unless* ZPP = NP, *any polynomial-time algorithm for solving these problems must have an approximation ratio of $\Omega\left(n^{1-\varepsilon}\right)$ or $\Omega\left(\kappa \log^{1-\varepsilon} d_n\right)$, where $0 < \varepsilon < 1$ is any constant.*

The theorem below shows that Int-Multi-Ssbo is MAX-SNP-hard even when severely restricted.

Theorem 9 (Inapproximability of Int-Multi-Ssbo with two slots)**.**
Int-Multi-Ssbo *is* MAX-SNP*-hard for $s = 2$ even when $\kappa = 1$ and $c_{j,k} = 1$ for all j and k.*

Proof. We reduce the MAX-2SAT-3 problem[16] to our problem. MAX-2SAT-3 is defined as follows. We are given a collection of m clauses $C_1, C_2, \ldots, C_m$ over n Boolean variables $z_1, z_2, \ldots, z_n$, where every clause is a disjunction of exactly two literals and every variable occurs exactly 3 times (and, thus, $m = 3n/2$). The goal is to find an assignment of truth values to variables to satisfy a maximum number of clauses. This problem was shown to be MAX-SNP-hard in [3].

Given an instance of MAX-2SAT-3 we create an instance of Int-Multi-Ssbo (*i.e.*, **(Q2)**) with $s = 2$ as follows. Every variable z_j corresponds to a keyword $\mathcal{K}_j$ with two slots. The variables $x_{j,1}$ and $x_{j,2}$ encode the truth assignments of the variable z_j with $x_{j,1} = 1$ indicating that z_j is true and $x_{j,2} = 1$ indicating that z_j is false; we will say that $x_{j,1}$ and $x_{j,2}$ are the slots corresponding to the literals z_j and $\neg z_j$, respectively. There are exactly m scenarios, each with probability $\frac{1}{m}$, defined in the following manner:

- $B_i = 1$ for $1 \leq i \leq m$.

[16]Our reduction approach should also work if we start with MAX-2SAT-k for any constant k.

- $c_{j,k} = 1$ for $1 \le j \le n$ and $1 \le k \le 2 = s$.
- For the ith clause C_i containing two *literals*, we have the ith scenario of the following form. Let $x_{j,k}$ and $x_{j',k'}$ be the slots corresponding to the two literals of the clause. Then we set $y_{i,j,k} = y_{i,j',k'} = 1$, and $y_{i,j,k} = 0$ if $j \ne j'$ or $k \ne k'$. For example, if $C_i = z_2 \vee (\neg z_3)$ then $y_{i,2,1} = y_{i,3,2} = 1$ and $y_{i,j,k} = 0$ for all other j and k.

An inspection of the construction reveals that it satisfies the following:

- Because this is an instance of Int-Multi-Ssbo, by Equation (2), for every $1 \le j \le n$, either $x_{j,1} = 1$ or $x_{j,2} = 1$ but *not both.* On the other hand, it is always possible to set at least one of the two variables $x_{j,1} = 1$ or $x_{j,2} = 1$ without decreasing the total payoff. Thus setting these variables correspond to a truth assignment.
- A scenario contributes a payoff of 1 if and only if *at least* one of two slots have been selected. Thus, contribution of a scenario correspond to satisfying a clause.

By the above observations, we satisfy m' clauses if and only if the above instance of Int-Multi-Ssbo has a total payoff of m'. □

7 Other Results

7.1 Improved Algorithms for Special Cases of Ssbo and Multi-Ssbo

By the phrase "`within an additive error of` δ" in Lemma 10 we mean that if our solution returns an objective value of x when the optimal value is y then $|x - y| \le \delta$.

Lemma 10.

(a) (Fixed number of scenarios) *If m is fixed,* Frac-Multi-Ssbo *admits a* **pseudo-polynomial** *time solution with an absolute error of δ for any fixed $\delta > 0$,* Int-Ssbo *admits a* **pseudo-polynomial** *time $O(1)$-approximation and* Int-Multi-Ssbo *admits a* **pseudo-polynomial** *time $O(\log^2 n)$-approximation.*

(b) (Fixed number of keywords) *If ns is fixed, then* Frac-Multi-Ssbo *admits a* **pseudo-polynomial** *time solution with an absolute error of δ for any fixed $\delta > 0$.*

(c) (Logarithmic number of keywords) *if $ns = O(\log m)$ then* Int-Multi-Ssbo *admits a polynomial time exact solution.*

(d) (Fixed number of scenarios and polynomial bids) *If m is fixed and the maximum size of all the numbers, namely* $\max\left\{ \max_{i,j,k}\{y_{i,j,k}\}, \max_i\{B_i\}, \max_i\left\{\frac{1}{\varepsilon_i}\right\}, \max_{i,j,k}\{c_{i,j,k}\} \right\}$*, is at most* $\mathsf{poly}(n)$ *then* Int-Multi-Ssbo *admits a polynomial time solution with an absolute error of δ for any fixed $\delta > 0$.*

Proof.
(a) and (b) We prove part **(a)** as follows (the proof for part **(b)** is similar). Consider the Frac-Multi-Ssbo problem; let $y = \max_{\substack{1 \le i \le m \\ 1 \le j \le n \\ 1 \le k \le s}} \{y_{i,j,k}\}$.

Proposition 4. *Let* $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_m^*)$ *and* $\mathbf{x}^* = (x_{1,1}^*, \ldots, x_{1,s}^*, x_{2,1}^*, \ldots, x_{2,s}^*, \cdots\cdots, x_{n,1}^*, \ldots, x_{n,s}^*)$ *be the solution vectors for an optimal solution of value* $\mathbb{E}[\text{payoff}^*] = \sum_{i=1}^m \alpha_i^* \left(\sum_{j=1}^n \sum_{k=1}^s y_{i,j,k}\, x_{j,k}^*\right)$. *Suppose that we approximate the vector* $\boldsymbol{\alpha}^*$ *by a vector* $\boldsymbol{\alpha}_\varepsilon = (\alpha_{1,\varepsilon}, \ldots, \alpha_{m,\varepsilon})$ *such that* $|\,\alpha_i^* - \alpha_{i,\varepsilon}\,| \leq \varepsilon$ *for each* i. *Then, if* $\varepsilon \leq \dfrac{\delta}{nsy}$ *we can compute a solution with a total expected payoff of at least* $\mathbb{E}[\text{payoff}] - \delta$.

Proof. Our algorithm is simple. Plugging the values of this $\boldsymbol{\alpha}_\varepsilon$ in **(Q2)** reduces it to a linear program, which can be solved optimally in polynomial time giving a solution vector, say $\mathbf{x}_\varepsilon$. Our solution vectors are $\boldsymbol{\alpha}_\varepsilon$ and $\mathbf{x}_\varepsilon$. Obviously, all the constraints are satisfied, so we just need to check the total expected payoff of our solution. For notational convenience, let $\mathsf{F}(\boldsymbol{\alpha}, \mathbf{x}) = \sum_{i=1}^m \alpha_i \left(\sum_{j=1}^n \sum_{k=1}^s y_{i,j,k}\, x_{j,k}\right)$ for two vectors $\mathbf{x} = (x_{1,1}, \ldots, x_{1,s}, x_{2,1}, \ldots, x_{2,s}, \cdots\cdots, x_{n,1}, \ldots, x_{n,s})$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$; thus $\mathsf{F}(\boldsymbol{\alpha}^*, \mathbf{x}^*) = \mathbb{E}[\text{payoff}^*]$. Then,

$$
\begin{aligned}
|\,\mathsf{F}(\boldsymbol{\alpha}^*, \mathbf{x}^*) - \mathsf{F}(\boldsymbol{\alpha}_\varepsilon, \mathbf{x}^*)\,| &\leq \varepsilon \sum_{j=1}^n \sum_{k=1}^s y_{i,j,k} \leq \varepsilon\, n\, s\, y \\
\implies \mathsf{F}(\boldsymbol{\alpha}_\varepsilon, \mathbf{x}_\varepsilon) \geq \mathsf{F}(\boldsymbol{\alpha}_\varepsilon, \mathbf{x}^*) &\geq \mathsf{F}(\boldsymbol{\alpha}^*, \mathbf{x}^*) - \varepsilon\, n\, s\, y \geq \mathsf{F}(\boldsymbol{\alpha}^*, \mathbf{x}^*) - \delta
\end{aligned}
$$

□

To get such a $\boldsymbol{\alpha}_\varepsilon$, for every $\alpha_{i,\varepsilon}$ we try out all rational numbers between 0 and 1 of the form $\frac{j\,\delta}{2\,n\,s\,y}$ for $j = 0, 1, \ldots, \dfrac{2\,n\,s\,y}{\delta}$ until we succeed. The total number of choices is at most $\left(\frac{2nsy}{\delta} + 1\right)^m$, which is **pseudo-polynomial**[17] in the size of the input since m is fixed.

The result for Int-Multi-Ssbo follows by using the above proof with Lemma 4.

(c) When $ns = O(\log m)$ then we can try out all possible $\mathsf{poly}(m)$ assignments of keywords to slots. For each assignment, we can directly calculate the values of $\alpha_1, \alpha_2, \ldots, \alpha_m$. We take the best of all such solutions.

(d) Let $p_1(n)$ be a polynomial in n such that $\max\left\{ y,\ \max_i\{B_i\}\ ,\ \max_{i,j,k}\{w_{i,j,k}\} \right\} < p_1(n)$. By the proof in part **(a)**, to ensure an absolute error of δ, it suffices to try all vectors $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ in which each α_i is a non-negative rational number with numerator and denominator at most $p_2(n)$ for some polynomial $p_2(n)$, and provide a solution of Int-Multi-Ssbo for this $\boldsymbol{\alpha}$ in polynomial time. We will refer to B_i as the "expected budget" for the ith scenario. Let $\mathbb{E}[\text{payoff}\,(j, k, b_1, \ldots, b_m)\,]$ be the optimal value of the expected payoff when *no* slot was selected after the kth slot of the jth keyword and the expected budget for the ith scenario was b_i. It is easy to see that the following recurrence holds:

$$
\mathbb{E}\,[\,\text{payoff}(j, k, b_1, \ldots, b_m)\,] = \max \left\{ \begin{array}{l} \displaystyle\sum_{i=1}^m y_{i,j,k} + \mathbb{E}\,[\,\text{payoff}(j-1, s, b_1 - \alpha_1 w_{1,j,k}, \ldots, b_m - \alpha_m w_{m,j,k})\,]\,, \\ \qquad\qquad\qquad \mathbb{E}\,[\,\text{payoff}(j, k-1, b_1, \ldots, b_m)\,] \end{array} \right\}
$$

Based on the above recurrence, it is easy to design a polynomial time dynamic programming algorithm to compute the optimal solution $\mathbb{E}\,[\,\text{payoff}(n, s, B_1, \ldots, B_m)\,]$ of Int-Multi-Ssbo. □

[17] The running time is *not* strongly polynomial since the input size depends polynomial on $\log_2 y$ (see Section 2.5).

7.2 Limitations of the Semidefinite Programming Relaxation Approaches for SSBO

(* Vector program (V) *)

$$
\begin{array}{ll}
\text{maximize} & \sum_{i=1}^{m}\sum_{j=p}^{q} y_{i,j}\,\mathcal{U}_i \centerdot \mathcal{V}_j \\
\text{subject to} & \forall\, 1\le i\le m\colon\ \sum_{j=1}^{n} c_{i,j}\, y_{i,j}\,\mathcal{U}_i \centerdot \mathcal{V}_j \le B_i \\
& \forall\, 1\le i\le m\colon\ \forall\, 1\le j\le n\colon\ \mathcal{U}_i \centerdot \mathcal{V}_j \ge 0 \\
& \forall\, 1\le i\le m\colon\ \mathcal{U}_i \centerdot \mathcal{U}_i \le 1 \\
& \forall\, 1\le i\le m\colon\ \mathcal{U}_i \in \mathbb{R}^{m+n} \\
& \forall\, 1\le j\le n\colon\ \mathcal{V}_j \centerdot \mathcal{V}_j \le 1 \\
& \forall\, 1\le j\le n\colon\ \mathcal{V}_j \in \mathbb{R}^{m+n}
\end{array}
$$

Figure 6: SDP-relaxation of **(Q1)**.

A natural Semidefinite programming (SDP) relaxation approach to solve quadratic programs such as **(Q1)**, extensively used in existing literatures for efficient approximations of quadratic programs for MAX-CUT, MAX-2SAT and many other problems [21], is as follows. We first add some redundant inequalities to **(Q1)**. For every i and j we add the inequality $\alpha_i x_j \ge 0$. Clearly, this does not change the solutions of **(Q1)**. Then, **(Q1)** can be relaxed to a vector program **(V)** by replacing the variables by $(m+n)$-dimensional vectors and the product of variables by the inner product (denoted by $\centerdot$) of the corresponding vectors. The resulting vector program is shown in Fig. 6; it is well known that **(V)** is a relaxation of **(Q1)** (*e.g.*, see [21]).

Since the lower bounds in Theorem 5 have $\varepsilon < 1$ and thus leaves a "very small" gap between this lower bound and the upper bound in Theorem 1, one might wonder if the gap can be somewhat narrowed down by designing an approximation algorithm based on the SDP-relaxation approaches whose approximation ratio is, say, $o\left(\frac{m}{\log m}\right)$ or $o\left(\frac{\log d_n}{\log\log d_n}\right)$? However, we show that the large integrality gap of the SDP-relaxation does not allow for such a possibility.

Lemma 11 (Limitations of SDP-relaxation approaches)**.** *Let $\kappa = 1$. Let OPT_{Q1} and OPT_{V} be the total optimal payoff for an instance of* **(Q1)** *and the optimal value of the objective function of* **(V)***, respectively. Then,* $\dfrac{OPT_{\mathrm{V}}}{OPT_{\mathrm{Q1}}} \ge \dfrac{m}{2} = \Theta\left(\dfrac{\log d_n}{\log\log d_n}\right)$.

Proof. We reuse the notations and terminologies used in the proof of Theorem 5. Let the given graph G be a completely connected graph; thus $\Delta_{\mathrm{ind}} = 1$. We construct an instance of SSBO as in Theorem 5. Thus, $\Delta_{\mathrm{Q1}} < 1 + \Delta_{\mathrm{ind}} = 2$. Note that $c_n = d_n = m^{6m}$ and thus $m = \Theta(\log d_n / \log\log d_n)$.

However, we show that $\mathrm{OPT}_{\mathrm{vector}} \ge m$. Let $\mathcal{U}_1, \dots, \mathcal{U}_m$ be a set of mutually orthogonal *unit-norm* vectors in $\mathbb{R}^{m+n}$ and let $\mathcal{V}_i = \mathcal{U}_{m-i+1}$ for $1 \le i \le m$. Thus, $\mathcal{U}_i \centerdot \mathcal{V}_j$ is 1 if $i + j = m + 1$ and is 0 otherwise, and $\mathcal{U}_i \centerdot \mathcal{U}_i = \mathcal{V}_i \centerdot \mathcal{V}_i = 1$ for all i. Obviously, $\sum_{i=1}^{m}\sum_{j=1}^{n} y_{i,j}\,\mathcal{U}_i \centerdot \mathcal{V}_j = m$. We now verify that this is indeed a valid solution of **(V)** by checking that it satisfies all the constraints $\left(\sum_{j=1}^{n} w_{i,j}\,\mathcal{U}_i \centerdot \mathcal{V}_j\right) \le B_i$ for $1 \le i \le n$. It can be seen that $\left(\sum_{j=1}^{n} w_{i,j}\,\mathcal{U}_i \centerdot \mathcal{V}_j\right) = w_{i,m-i+1} = c_{m-i+1} = B_i$. ∎

7.3 Combinatorial Dual of SSBO Problems

In DUAL-SSBO, the natural combinatorial dual version of SSBO, we are given a lower bound, say P, on $\mathbb{E}[\text{payoff}]$. Our goal is to compute the *minimum* possible value of the budget B of the advertiser

(* Quadratic program **(Dual-Q1)** *)
minimize B
subject to $\sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_i x_j y_{i,j} \geq P$
$\forall\, 1 \leq i \leq m\colon\ \alpha_i\left(\sum_{j=1}^{n} w_{i,j}\, x_j\right) \leq \varepsilon_i\, B$
$\forall\, 1 \leq i \leq m\colon\ 0 \leq \alpha_i \leq 1$
$\forall\, 1 \leq j \leq n\colon\ 0 \leq x_i \leq 1$

Figure 7: Quadratic program for Dual-Ssbo.

such that his/her total expected payoff is *at least* P. The dual version Dual-Multi-Ssbo of Multi-Ssbo can be defined in a manner analogous to that of Dual-Ssbo. Dual-Ssbo can be reformulated as the quadratic program **(Dual-Q1)** shown in Fig. 7.

Obviously, Dual-Ssbo is NP-hard since Ssbo is NP-hard. For a given required expected profit $\mathcal{P}$, let $\mathcal{B}_{\mathcal{P}}$ be the minimum budget that achieves the expected total profit $\mathcal{P}$. We define a *bi-criteria approximation* for Dual-Ssbo in the following manner:

> a (δ, γ)-approximation for Dual-Ssbo, for $\delta, \gamma \geq 1$, is a solution that achieves an expected total profit of at least $\frac{\mathcal{P}}{\delta}$ with a budget of $\gamma\, \mathcal{B}_{\mathcal{P}}$.

Lemma 12.
(a) (Inapproximability of Dual-Ssbo via inapproximability of Ssbo)

- *If* Frac-Ssbo *cannot be approximated to within a ratio of $\rho > 1$ for some parameter range, then* Dual-Frac-Ssbo *also cannot be approximated to within a ratio of ρ for the same parameter range.*

- *If* Int-Ssbo *cannot be approximated to within a ratio of $\rho > 1$ for some parameter range, then* Dual-Int-Ssbo *also cannot be approximated to within a ratio of $\frac{\rho}{200 \ln m}$ for the same parameter range.*

(b) (Bi-criterion approximation of Dual-Frac-Ssbo via Frac-Ssbo) *If* Frac-Ssbo *can be approximated to within a ratio of $\rho > 1$ for some parameter range, then* Frac-Ssbo *has a $(\rho, 1)$-approximation in the same parameter range.*

Proof. Let $\mathbb{E}\left[\text{payoff}^{\mathcal{B}}\right]$ be the optimal total expected payoff for Ssbo when the budget is $\mathcal{B}$. For any constant $\Delta > 1$, a solution of **(Q1)** with a budget of $\mathcal{B}$ is obviously also a solution of the same instance of **(Q1)** with a budget of $\Delta\mathcal{B}$. This implies $\mathbb{E}\left[\text{payoff}^{\Delta\mathcal{B}}\right] \geq \mathbb{E}\left[\text{payoff}^{\mathcal{B}}\right]$. Let $p = \sum_{i=1}^{m}\sum_{j=1}^{n} y_{i,j}$ and $b = \max_{1 \leq i \leq m}\left\{\sum_{j=1}^{n} a_{i,j}\, c_{i,j}\right\}$; note that both $\log_2 p$ and $\log_2 b$ are polynomial in the size of the input (see Section 2.5).

We prove **(a)** by contradiction. Suppose that some version of Dual-Ssbo has a ρ-approximation. Consider an instance of the *same* version of Ssbo and suppose the budget is B. We do a binary search in the range of positive integers $[1, p]$ in polynomial time with the approximation algorithm for Dual-Ssbo to find a $\mathcal{P} \in [1, p]$ such that $\mathcal{B}_{\mathcal{P}-1} < \rho\, B$ but $\mathcal{B}_{\mathcal{P}} \geq \rho\, B$. Consider this solution of Dual-Ssbo and suppose that $\mathcal{B}^*$ is the actual optimal value of the budget corresponding to the total expected payoff $\mathcal{P}$. Thus, $\mathcal{B}^* \geq \frac{\mathcal{B}_{\mathcal{P}}}{\rho} \geq B$ and $\mathbb{E}\left[\text{payoff}^{\mathcal{B}_{\mathcal{P}}}\right] \geq \mathbb{E}\left[\text{payoff}^{\mathcal{B}^*}\right] \geq \mathbb{E}\left[\text{payoff}^{B}\right]$. Suppose that we now divide every x_i by ρ. This provides a valid solution of Frac-Ssbo with a total expected payoff of at least $\frac{\mathbb{E}\left[\text{payoff}^{\mathcal{B}_\mathcal{P}}\right]}{\rho}$. By Lemma 4, from this valid solution of Frac-Ssbo one can obtain a solution of Int-Ssbo with a total expected payoff of at least $\frac{\mathbb{E}\left[\text{payoff}^{\mathcal{B}_\mathcal{P}}\right]}{200\,\rho\,\ln m}$.

To prove **(b)**, suppose that some version of Ssbo with a budget of $\mathcal{B}$ has a ρ-approximation algorithm. Consider an instance of the same version of Dual-Ssbo with a requirement of total expected payoff of $\mathcal{P}$ and let $\mathcal{B}_\mathcal{P}$ be the value of an optimal budget for this instance. Since $\left(1 - \dfrac{1}{B+1}\right) \mathbb{E}\left[\text{payoff}^{B+1}\right] \le \mathbb{E}\left[\text{payoff}^{B}\right] \le \mathbb{E}\left[\text{payoff}^{B+1}\right]$, we do a binary search in the range of positive integers $[1, b]$ in polynomial time with the ρ-approximation algorithm for Ssbo to find a $\mathcal{B} \in [1, b]$ such that $\frac{\mathcal{P}}{\rho} \le \mathbb{E}\left[\text{payoff}^{\mathcal{B}}\right] \le \rho\,\mathcal{P} + 1$. Thus, this provides a solution of the Dual-Ssbo with a total expected payoff of at least $\frac{\mathcal{P}}{\rho}$ and a budget of at most $\mathcal{B}_\mathcal{P}$, giving the desired $(\rho, 1)$-approximation in **(b)**. $\square$

8 Conclusion

We have presented the first known approximation algorithms as well as hardness results for stochastic budget optimization under the scenario model. The scenario model is natural in many areas, and it is particularly apt for internet ad systems. We obtained our results by making the connection between these problems and a special case of bipartite quadratic programs; we exploited this intuition crucially in both approximation algorithms and hardness proofs. These classes of quadratic programs may have independent applications elsewhere.

Our work shows that there are several instances of parameters where stochastic budget optimizations are solvable with reasonable computational resource even with multiple slots. Our hope is that therefore, in practice, one can carefully model particular applications such as sponsored search, so that the parameters are suitable, and advertisers can optimize their campaigns more effectively than is typically done now by applying some of the algorithms in this paper.

Acknowledgements

We thank the reviewers for their detailed comments which improved both the readability and the technical content of the paper.

References

[1] N. Alon and A. Naor. *Approximating the cut-norm via Grothendieck's inequality*, SIAM Journal on Computing, 35, 787-803, 2006.

[2] G. Aggarwal, A. Goel, and R. Motwani. *Truthful auctions for pricing search keywords*, 7th ACM Conference on Electronic Commerce, 1-7, 2006.

[3] P. Berman and M. Karpinski. *On some tighter inapproximability results, further improvements*, Electronic Colloquium on Computational Complexity, 5 (29), 1998.

[4] C. Borgs, J. Chayes, O. Etesami, N. Immorlica, K. Jain and M. Mahdian. *Dynamics of bid optimization in online advertisement auctions*, 16th International World Wide Web Conference, 531-540, 2007.

[5] T.-M. Bu, X. Deng and Q. Qi. *Forward looking Nash equilibrium for keyword auction*, Information Processing Letters, 105(2), 41-46, 2008.

[6] M. Cary, A. Das, B. Edelman, I. Giotis, K. Heimerl, A. R. Karlin, C. Mathieu and M. Schwarz. *Greedy bidding strategies for keyword auctions*, 8th ACM conference on Electronic commerce, 262-271, 2007.

[7] C. Chekuri and S. Khanna. *A PTAS for the multiple knapsack problem*, SIAM Journal on Computing, 35 (3), 713–728, 2006.

[8] F. Cocco, A. Consiglio and S. Zenios. *Scenario Optimization Asset and Liability Modeling for Endowments with Guarantees*, Wharton Financial Institutions Center WP No. 00-41, June 12, 2000, available at `http://ssrn.com/abstract=253039` or `doi:10.2139/ssrn.253039`.

[9] R. S. Dembo. *Scenario optimization*, Annals of Operations Research, 30 (1), 63-80, 1991.

[10] B. Edelman, M. Ostrovsky, and M. Schwarz. *Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords*, American Economic Review, 97(1), 242-259, 2007.

[11] J. Feldman, S. Muthukrishnan, M. Pál and C. Stein. *Budget optimization in search-based advertising auctions*, 8th ACM Conference on Electronic Commerce, 40-49, 2007.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1979.

[13] M.X. Goemans and D.P. Williamson. *Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming*, Journal of the ACM, 42, 1115-1145, 1995.

[14] J. Håstad. *Clique is hard to approximate within* $n^{1-\varepsilon}$, Acta Mathematica, 182, 105-142, 1999.

[15] H. Mausser and D. Rosen. *Applying Scenario Optimization to Portfolio Credit Risk*, The Journal of Risk Finance, 2 (2), 36-48, 2001.

[16] K. Jansen. *Parameterized approximation scheme for the multiple knapsack problem*, SIAM Journal on Computing, 39 (4), 1392-1412, 2009.

[17] E. L. Lawler. *Fast Approximation Algorithms for Knapsack Problems*, Mathematics of Operations Research, 4(4), 339-356, 1979.

[18] R. Motwani and P. Raghavan. *Randomized Algorithms*, Cambridge University Press, 1995.

[19] S. Muthukrishnan, M. Pál and Z. Svitkina. *Stochastic Models for Budget Optimization in Search-Based Advertising*, Algorithmica, 58 (4), 1022-1044, 2010.

[20] H. Varian. *Position auctions*, International Journal of Industrial Organization, 25(6), 1163-1178, 2007.

[21] V. Vazirani. *Approximation Algorithms*, Springer-Verlag, 2001.

[22] S. A. Zenios (editor). *Financial Optimization*, Cambridge University Press, 1996.

[23] Y. Zhou, D. Chakrabarty, R. Lukose. *Budget Constrained Bidding in Keyword Auctions and Online Knapsack Problems*, 4th International Workshop On Internet And Network Economics, 566-576, 2008.

[24] Y. Zhou and V. Naroditskiy. *Algorithm for Stochastic Multiple-Choice Knapsack Problem and Application to Keywords Bidding*, 17th International World Wide Web Conference, poster paper.