# Approximation Algorithms for 2-Stage Stochastic Scheduling Problems

David B. Shmoys[1]* and Mauro Sozio[2]**

[1] School of ORIE and Dept. of Computer Science, Cornell University, Ithaca, NY 14853
shmoys@cs.cornell.edu
[2] Dept. of Computer Science, University of Rome "La Sapienza", Italy.
sozio@di.uniroma1.it

**Abstract.** There has been a series of results deriving approximation algorithms for 2-stage discrete stochastic optimization problems, in which the probabilistic component of the input is given by means of "black box", from which the algorithm "learns" the distribution by drawing (a polynomial number of ) independent samples. The performance guarantees proved for such problems, of course, is generally worse than for their deterministic analogue. We focus on a 2-stage stochastic generalization of the problem of finding the maximum-weight subset of jobs that can be scheduled on one machine where each job is constrained to be processed within a specified time window. Surprisingly, we show that for this generalization, the same performance guarantee that is obtained for the deterministic case can be obtained for its stochastic extension.
Our algorithm builds on an approach of Charikar, Chekuri, and Pál: one first designs an approximation algorithm for the so-called polynomial scenario model (in which the probability distribution is restricted to have the property that there are only a polynomial number of possible realizations of the input that occur with positive probability); then one shows that by sampling from the distribution via the "black box" to obtain an approximate distribution that falls in this class and approximately solves this approximation to the problem, one nonetheless obtains a near-optimal solution to the original problem. Of course, to follow this broad outline, one must design an approximation algorithm for the stochastic optimization problem in the polynomial scenario model, and we do this by extending a result of Bar-Noy, Bar-Yehuda, Freund, Naor, and Schieber.
Furthermore, the results of Bar-Noy et al. extend to a wide variety of resource-constrained selection problems including, for example, the unrelated parallel-machine generalization $R|r_j|\sum w_j U_j$ and point-to-point admission control routing in networks (but with a different performance guarantee). Our techniques can also be extended to yield analogous results for the 2-stage stochastic generalizations for this class of problems.

* Research supported partially by NSF grants CCR-0635121 & DMI-0500263.
** This work was done while this author was a visiting student at Cornell University. The work was partially supported by NSF grant CCR-0430682 and by EC project DELIS.

## 1 Introduction

Consider the following 2-stage stochastic optimization problem: there are $n$ users, each of whom might request a particular communication channel, which can serve at most one user at a time, for a specified length of time within a specified time interval; for a given planning period, it is not known which of the $n$ users will actually make their request – all that is known is a probability distribution over the subsets of users indicating which subset might be active; each user has an associated profit for actually being scheduled on the channel; alternatively, the manager of the channel can redirect the user to other providers, thereby obtaining a specified (but significantly smaller) profit; the aim is to decide which users to defer so as to maximize the expected profit over the two stages (where the expectation is with respect to the probability distribution over subsets of active users). Thus, this is a stochastic generalization of the (maximization version) of the single machine scheduling problem that is denoted in the notation of [4] as $1|r_j|\sum w_jU_j$ and we shall refer to this generalization as the *2-stage stochastic* $1|r_j|\sum w_jU_j$. For the deterministic version of this problem, Bar-Noy, Bar-Yehuda, Freund, Naor, & Schieber give a $\rho$-approximation algorithm for any constant $\rho > 2$; rather surprisingly, we show that the exact same result holds for the stochastic generalization. (A *$\rho$-approximation algorithm* for an optimization problem is a (randomized) polynomial-time algorithm that finds a feasible solution with (expected) cost within a factor of $\rho$ of optimal.)

Recently, there has been a series of results for 2-stage discrete stochastic optimization problems with recourse, starting with the work of Dye, Stougie, and Tomasgard[3] that addressed a knapsack-like single-node network provisioning problem. That paper made the simplifying assumption of the *polynomial scenario model* in which there are (only) a polynomial number of scenarios that can be realized in the second stage, and thereby derived the first worst-case performance guarantees for polynomial-time algorithms for models of this type. Kong & Schaefer [8] gave an 2-approximation algorithm for a 2-stage variant of the the maximum-weight matching problem, again in a polynomial scenario model. Later, Immorlica, Karger, Minkoff, and Mirrokni [7], and also Ravi and Sinha [9] addressed analogous questions based on deterministic problems such as the vertex cover problem, the set covering problem, the uncapacitated facility location problem, and network flow problems. The former paper also considered the situation when the probability distribution conformed to an *independent activation model* which, in our setting for example, would mean that there is a probability associated with each user and the active set is drawn by assuming that these are independent Bernoulli random events. However, for these latter results they introduced the *proportionality assumption* in which the corresponding costs for an element in the two stages had constant ratio $\lambda$ for all elements. Gupta, Pál, Ravi, and Sinha [5] proposed a much more general mechanism for specifying the probability distribution, in which one has access to a *black box* from which to generate independent samples according to the distribution, and thereby make use of a polynomial number of samples in the process of computing the first-stage decisions. They gave constant approximation algorithms for a number of 2-stage stochastic optimization problems in this model, most notably the minimum-cost rooted Steiner tree problem and the uncapacitated facility location problem, but they also require the proportionality assumption.

Shmoys & Swamy [10] gave an LP-rounding technique, and showed that one could derive a polynomial-time approximation scheme for the exponentially-large linear programming relaxations in order to derive the first approximation algorithms in the black box model without the proportionality assumption, in particular for a variety of set covering-related problems, the uncapacitated facility location problem, and multi-commodity flow problems. Swamy & Shmoys [11] extend this to constant-stage models, and also show that the so-called sample average approximation yields a polynomial approximation scheme for the LP relaxations. Charikar, Chekuri, and Pál [2] gave a general technique based on the sample average approximation that, for a broad class of 2-stage stochastic minimization problem with recourse, in effect reduced the problem of obtaining a good approximation algorithm for the black box model, to the problem of obtaining the analogous result in the polynomial scenario setting.

We build on these results, by first constructing an approximation algorithm for our maximization problem in the polynomial scenario model, and then derive a maximization variant of the result of [2] (but still specialized to our class of problems) to obtain approximation algorithms in the black box probability model.

We focus on the central model in the class proposed by Bar-Noy, Bar-Yehuda, Freund, Naor, and Schieber [1], who gave primal-dual algorithms for a rich class of deterministic resource allocation and scheduling problems. In their terminology, there is a set of activities, $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$; let $\mathcal{N} = \{1, \ldots, n\}$ index this set. For each activity $\mathcal{A}_j$, $j \in \mathcal{N}$, there is a set of possible instances $A_j$ that specify the various ways in which the activity might be handled (so, in the description above, assuming integer data for the input times, for each user we have one instance for each possible integer starting time that would have it complete by the deadline). This approach appears to convert the original input to a new input in which there are a pseudopolynomial number of instances for each activity. However, Bar-Noy et al. also show how to convert their pseudopolynomial-time algorithm into a polynomial-time one, while losing only a $1+\epsilon$ factor in the performance guarantee.

Our algorithm is a rather natural extension of the approach of Bar-Noy et al. We first run their algorithm on each of the polynomially many scenarios, where the profit of selecting an instance is its contribution to the overall expected second stage profit. For each scenario (which is, after all just an ordinary deterministic input), this generates a feasible dual solution. The deterministic dual variables are of two types: those that are dual to the constraint that says that each activity is scheduled in at most one way (that is, at most one instance of each activity is selected); and those that correspond to the constraint that at each time at most one instance (over all activities) is active. The usual interpretation of dual variables leads us to view the former as providing the marginal expected profit attainable by having this activity on hand in a particular scenario. Thus, we decide to defer an activity $\mathcal{A}_j$, if the total of the corresponding dual variables, summed over all scenarios, is less than the profit collected by actually deferring that activity. This gives the stage I actions. The stage II actions for each scenario are computed by adapting the algorithm of Bar-Noy et al.; we first compute a dual solution that includes even the deferred activities, but then does not select any instance of a deferred activity in constructing the primal solution.

The analysis of our algorithm is also surprisingly simple, and is based on a primal-dual approach using an integer programming formulation of the 2-stage problem. We show that the dual solutions constructed in each scenario can be pieced together to yield a feasible solution for the dual to the linear programming relaxation, and can then show that the expected profit of the primal solution constructed is at least half the value of the feasible dual solution found. This yields that the resulting algorithm is a 2-approximation algorithm. Like the algorithm of Bar-Noy et al., this is a pseudopolynomial-time algorithm, but an approach identical to the one they employed yields a polynomial-time algorithm, while losing a factor of $1 + \epsilon$ in the performance guarantee. Although we focus on this single-machine scheduling model, our approach can be generalized to yield analogously strong results for 2-stage stochastic generalization of the class of problems for which the framework of Bar-Noy et al. applies. This will be discussed in detail in the full version of this paper.

There are other potential 2-stage stochastic extensions of the problem of computing a maximum-weight subset of jobs that can be feasible scheduled. One other natural approach is to use the first stage to make initial decisions about which users to service (but to commit to serve them if they are active), and then to allow the possibility of serving additional users in the second stage, once the probabilistic choice of scenario has been made (with correspondingly lesser profit). We show that the maximum independent set problem can be reduced to an extremely restricted special case of this model in an approximation-preserving way, and hence we cannot hope to obtain a good approximation algorithm for this setting (unless $\mathcal{P} = \mathcal{NP}$). There are few (if any) such strong inapproximability results known for stochastic optimization problems for which their deterministic analogue is relatively easily approximable.

## 2 IP & LP formulations: 2-stage stochastic models

We start by giving a natural integer (linear) programming formulation (and its dual) for the 2-stage stochastic version of $1|r_j|\sum_j w_j U_j$, in its pseudopolynomial-sized variant.

Let $\mathcal{S}$ be a collection of explicitly given scenarios $\{S_1, \ldots, S_m\}$ that occur with positive probability; in each scenario $S$, for each activity $\mathcal{A}_j$, there is an associated set of available instances $A_j(S) \subseteq A_j$. For each instance $I$, there is an associated starting time $s(I)$, and an associated ending time $e(I)$. For each scenario $S \in \mathcal{S}$, there is an associated probability $q(S)$, where $q(S) \geq 0$ and $\sum_{S \in \mathcal{S}} q(S) = 1$. In stage I, we must decide which activities to defer, and thereby obtain a (small) profit of $p_j^{\mathrm{I}}$, or else retain for stage II, in which for each scenario $S$ we can obtain a profit $p_j^{\mathrm{II}}(I, S)$ for assigning this activity using instance $I \in A_j(S)$. We give an integer programming formulation of this problem. For each activity $\mathcal{A}_j$, we have a 0-1 variable $x_j$ that indicates whether activity $\mathcal{A}_j$ is deferred in the first phase or not (where $x_j = 1$ means that it is deferred). For each instance $I$ of activity $A_j(S)$, we have a variable $y_j(I, S)$ whose value is 1 if and only if instance $I$ of this activity is scheduled. Let $\mathcal{T}$ be the set of all start-times and end-times of all instances belonging to all activities and let $\mathcal{T}_I = \{t \in \mathcal{T} | s(I) \leq t < e(I)\}$ for each instance $I$. Moreover, let $f(I) \in \mathcal{T}$ be maximal such that $f(I) < e(I)$.

We can formulate the 2-stage problem of maximizing the total expected profit as follows:

$$\max \sum_{j\in\mathcal{N}} p_j^{\mathrm{I}} x_j + \sum_{j\in\mathcal{N}}\sum_{S\in\mathcal{S}}\sum_{I\in A_j(S)} q(S)p_j^{\mathrm{II}}(I,S)y_j(I,S) \tag{SIP}$$

$$\text{s.t. } x_j + \sum_{I\in A_j(S)} y_j(I,S) \le 1 \qquad \forall j\in\mathcal{N}, S\in\mathcal{S}, \tag{1}$$

$$\sum_{j\in\mathcal{N}}\sum_{I\in A_j(S):t\in\mathcal{T}_I} y_j(I,S) \le 1 \qquad \forall S\in\mathcal{S}, t\in\mathcal{T}, \tag{2}$$

$$x_j, y_j(I,S) \in \{0,1\}, \qquad \forall j\in\mathcal{N}, S\in\mathcal{S}, I\in A_j(S). \tag{3}$$

Let (SLP) be the LP obtained by replacing (3) by non-negativity constraints for these variables. If we let $u_j(S)$ be the dual variables corresponding to the constraints (1), and let $v_t(S)$ denote the dual variables corresponding to the constraints (2), then we can write the LP dual of (SLP) as:

$$\min \sum_{j\in\mathcal{N}}\sum_{S\in\mathcal{S}} u_j(S) + \sum_{S\in\mathcal{S}}\sum_{t\in\mathcal{T}} v_t(S) \tag{SD}$$

$$\text{s.t. } \sum_{S\in\mathcal{S}} u_j(S) \ge p_j^{\mathrm{I}}, \qquad \forall j\in\mathcal{N}, \tag{4}$$

$$u_j(S) + \sum_{t\in\mathcal{T}_I} v_t(S) \ge q(S)p_j^{\mathrm{II}}(I,S), \qquad \forall j\in\mathcal{N}, S\in\mathcal{S}, I\in A_j(S), \tag{5}$$

$$u_j(S), v_t(S) \ge 0. \tag{6}$$

It is important to note that our algorithm will not need to solve any of these linear programs! We will simply apply an algorithm for the deterministic variant (for which a performance guarantee relative the optimal value of the deterministic LP is known) to an input based on each scenario $S \in \mathcal{S}$, and then use the linear programs to analyze the performance of the resulting algorithm.

## 3 An algorithm for the polynomial scenario model

We shall show how to adapt the primal-dual algorithmic framework of Bar-Noy, Bar-Yehua, Freund, Naor, & Schieber [1] to yield an approximation algorithm with the identical performance guarantee for the 2-stage stochastic variant of $1|r_j|\sum w_j U_j$, in the polynomial scenario model. For this model, it is straightforward to derive a constant approximation algorithm. The simplest approach is to randomize, and with probability 1/2 to defer all jobs, and otherwise, to run the 2-approximation algorithm of Bar-Noy et al. on the active jobs in the second stage; this is a randomized 4-approximation algorithm. In the polynomial scenario model, one can improve upon this by comparing the benefit of deferring all users with the expected profit obtained by the Bar-Noy algorithm based on not deferring anyone, and then selecting the better of the two. This is easily shown to be a 3-approximation algorithm (and can be extended to the black box model while losing only a factor of $1+\epsilon$). Thus, the surprising aspect of our result is that it is in fact possible to obtain an algorithm for the 2-stage generalization without degrading the performance guarantee at all.

The framework of Bar-Noy et al. works in two phases: a pushing phase in which a dual solution is constructed along with a stack of instances that might be selected to be scheduled; and a popping phase in which elements of the stack are popped off, and accepted for scheduling provided that they do not conflict with activities already scheduled by this procedure.

The algorithm for the 2-stage problem proceeds as follows. For each scenario $S \in \mathcal{S}$, the deterministic profit $p_j(I)$ is $q(S)p_j^{\mathrm{II}}(I,S)$ for each $j \in \mathcal{N}$, and each $I \in A_j(S)$. We execute the pushing procedure of the algorithm proposed in Bar-Noy et al. for each scenario $S \in \mathcal{S}$. Algorithm 1 shows the pseudocode for this procedure. We let $u_j(S)$ denote the dual variable corresponding to the deterministic analogue of (1) computed by this procedure. Then, for each activity $\mathcal{A}_j$, $j \in \mathcal{N}$, we check if

$$p_j^{\mathrm{I}} \geq \sum_{S \in \mathcal{S}} u_j(S), \tag{7}$$

and defer each activity $\mathcal{A}_j$ that satisfies this condition. This completes the first stage action. We shall also denote this solution by setting $\bar{x}_j = 1$ for each deferred activity $\mathcal{A}_j$, and setting $\bar{x}_j = 0$ otherwise.

In what follows, we shall say that an instance $I$ is *uncovered* if constraint (5) for instance $I$ is not satisfied and we say that $I$ is *tight* if this constraint is satisfied with equality.

For the second stage, for a given scenario $S \in \mathcal{S}$, we recompute the execution of the pushing procedure. Then we compute a feasible schedule by executing the popping procedure of the algorithm of Bar-Noy et al., but we delete each activity that was deferred in the first phase. We denote this solution by setting $\bar{y}_j(I,S) = 1$ for each scheduled instance $I$, and setting $\bar{y}_j(I,S) = 0$ otherwise. Algorithm 2 shows the pseudocode for the second phase for a given scenario.

The main intuition behind the deferring rule is the following. Suppose at the end of the pushing phase the total value of variables $u$ of an activity $\mathcal{A}_j$ is “small”. There are two possible reasons for this. The total profit of all instances of $\mathcal{A}_j$ is smaller than $p_j^{\mathrm{I}}$. In this case, it is clear that deferring the activity is the best we can do. If the total profit $P$ of instances of $\mathcal{A}_j$ is greater than $p_j^{\mathrm{I}}$, then since $u$ is “small”, there are many other instances of other activities which are in conflict with instances of $\mathcal{A}_j$. Hence, $P$ can be “replaced” by the profit of these instances, and we can gain other profit by deferring $\mathcal{A}_j$. More generally, the value of the sum reflects the total expected marginal value of the activity $A_j$; if this is less than the (sure) profit gained by deferring it, then certainly deferring it is a good thing to do.

We shall prove that the performance guarantee of the two-phase algorithm is 2. The main idea behind this proof is the following. Each instance increases the total value of the dual variables by some amount $2\delta$. For instances that belong to a non-deferred activity, we are able to charge $\delta$ to a scheduled instance. For instances that belong to a deferred activity, we charge this amount to the profit gained by deferring that activity.

Given a scenario $S$ we say that $I \in A_j(S)$ and $\hat{I} \in A_l(S)$ are *incompatible* if $j = l$ or their time intervals overlap. For each instance $I \in A_j(S)$, we refer to the variables which occur in the constraint (5) for $I$, as “the variables of $I$”.

**Algorithm 1** Pushing procedure for the first phase in scenario $S$

```
1: $Stack(S)$=$\emptyset$;
2: $u_j(S) \leftarrow 0 \quad \forall j \in N$;
3: $v_t(S) \leftarrow 0 \quad \forall t \in \mathcal{T}$;
4: **while** no uncovered instance is left **do**
5:     select an uncovered instance $I \in A_j(S),\ j \in \mathcal{N}$ with minimum end-time;
6:     push(I,$Stack(S)$);
7:     let $\delta(I,S) = (q(S)p_j^{\text{II}}(I,S) - u_j(S) - \sum_{t\in\mathcal{T}_I} v_t(S))/2$;
8:     $u_j(S) \leftarrow u_j(S) + \delta(I,S)$;
9:     $v_{f(I)}(S) \leftarrow v_{f(I)}(S) + \delta(I,S)$;
10: **end while**
```

**Algorithm 2** The algorithm for the second phase in scenario $S$

```
1: /* *pushing procedure* */
2: $Stack(S)$=$\emptyset$;
3: $u_j(S) \leftarrow 0 \quad \forall j \in N$;
4: $v_t(S) \leftarrow 0 \quad \forall t \in \mathcal{T}$;
5: **while** no uncovered instance is left **do**
6:     select an uncovered instance $I \in A_j(S),\ j \in \mathcal{N}$ with minimum end-time;
7:     push(I,$Stack(S)$);
8:     let $\delta(I,S) = (q(S)p_j^{\text{II}}(I,S) - u_j(S) - \sum_{t\in\mathcal{T}_I} v_t(S))/2$;
9:     $u_j(S) \leftarrow u_j(S) + \delta(I,S)$;
10:    $v_{f(I)}(S) \leftarrow v_{f(I)}(S) + \delta(I,S)$;
11: **end while**
12: /* *scheduling procedure* */
13: **while** $Stack(S)$ is not empty **do**
14:     I=pop($Stack(S)$);
15:     Let $j \in \mathcal{N} : I \in A_j(S)$;
16:     **if** $A_j$ is not deferred and $I$ is not in conflict with other scheduled instances **then**
17:         schedule $I$ and set $\bar{y}_j(I,S) = 1$;
18:     **end if**
19: **end while**
```

**Theorem 1.** *For the* $2$*-stage stochastic maximization version of* $1|r_j|\sum w_jU_j$*, there is a* $(2+\epsilon)$*-approximation algorithm in the polynomial scenario model.*

*Proof.* We shall consider only the version of the problem in which we have a pseudopolynomial representation of the input: that is, for each activity, we have an explicitly given set of allowed starting times. However, for each scenario, this is exactly the algorithm of Bar-Noy et al. (on a carefully constructed input), who show that it can be converted to run in polynomial time for $1|r_j|\sum w_jU_j$, while losing a factor of $1+\epsilon$ in the performance guarantee. This will thereby yield the theorem in the form stated above.

Let $\bar{u}_j(S)$ and $\bar{v}_t(S)$ be the value of the dual variables $u$ and $v$ at the end of the algorithm. First consider the constraints (5); the algorithm ensures that these are satisfied by the dual solution computed. This is a consequence of the fact that as long as there exists an uncovered instance, the algorithm pushes an instance in the stack and increases its dual variables making a constraint (5) tight. Hence, at the end of the algorithm, there does not exist an uncovered instance, and each constraint (5) is satisfied. On the other hand, constraint (4) can be violated by any deferred activity. In order to satisfy this constraint, we increase the value of dual variables in the following way. Let

$$\delta_j = p_j^{\mathrm{I}} - \sum_{S \in \mathcal{S}} \bar{u}_j(S) \quad j = 1, \ldots, n$$

and let $\bar{S} \in \mathcal{S}$, be an arbitrarily chosen scenario. For each activity $\mathcal{A}_j$, we increase the value of $\bar{u}_j(\overline{S})$ by $\delta_j$. Clearly, this maintains that the other constraints are satisfied, and ensures that constraint (4) is satisfied now as well.

We now prove the performance guarantee of the algorithm is 2. The essence of the proof is as follows. In each scenario $S$, for each instance $I$ of a non-deferred activity, we charge $\delta(I, S)$ to some scheduled instance. For each instance $I$ of a deferred activity $\mathcal{A}_j$, we charge $\delta_j$ and $\delta(I, S)$ to the profit $p_j^{\mathrm{I}}$. Hence, at the end of the algorithm, all amounts $\delta$ are "charged" to some profit. Moreover, the sum of all these $\delta$, multiplied by 2, gives a bound on the total value of the dual variables. The theorem then follows from weak duality.

Consider a scenario $S$. Let $\hat{I} \in A_j(S)$ be an instance scheduled in $S$ such that $\mathcal{A}_j$ is not deferred, $j \in \mathcal{N}$. Let $B_{\hat{I}}(S)$ be a set which contains $\hat{I}$ and as well as instances that are:

– incompatible with $\hat{I}$ and
– pushed onto $Stack(S)$ before $\hat{I}$.

Consider each instance $I$ in $B_{\hat{I}}(S)$. When $I$ is placed on the stack, there are two dual variables that are increased by $\delta(I, S)$. For each such $I$, one of these two variables are variables of $\hat{I}$. If $I \in A_j(S)$, then the variable $u_j(S)$ occurs in constraint (5) for $\hat{I}$. Otherwise, since $e(\hat{I}) \geq e(I)$, then the variable $v_{f(I)}(S)$ occurs in this constraint. Let $\hat{u}$ and $\hat{v}$ be the value of dual variables $u$ and $v$ at the time $\hat{I}$ is pushed in the stack. We have that:

$$\sum_{I \in B_{\hat{I}}(S)} \delta(I, S) \leq \hat{u}_j(S) + \sum_{t \in \mathcal{T}_{\hat{I}}} \hat{v}_t(S) \leq q_S p_j^{\mathrm{II}}(\hat{I}, S) \tag{8}$$

where last inequality follows from the fact that $\hat{I}$ is uncovered before being pushed on the stack and after that, its variables are increased in order to make constraint (5) tight.

Note that each instance $I$ of a non-deferred activity belongs to the set $B_{\hat{I}}(S)$ for some instance $\hat{I}$. This follows from the fact that either $I$ is scheduled or there is another instance $\hat{I}$ pushed after $I$ in the stack, which has been scheduled instead of $I$. This implies that for each scenario $S \in \mathcal{S}$

$$\begin{aligned} \sum_{\substack{j \in \mathcal{N}: \\ \bar{x}_j = 0}} \sum_{I \in A_j(S)} \delta(I, S) &= \sum_{\substack{j \in \mathcal{N}: \\ \bar{x}_j = 0}} \sum_{\substack{\hat{I} \in A_j(S): \\ y_j(\hat{I}, S) = 1}} \sum_{I \in B_{\hat{I}}(S)} \delta(I, S) \\ &\leq \sum_{\substack{j \in \mathcal{N}: \\ \bar{x}_j = 0}} \sum_{\hat{I} \in A_j(S)} q_S p_j^{\mathrm{II}}(\hat{I}, S) \bar{y}_j(\hat{I}, S) \end{aligned} \tag{9}$$

For each deferred activity $\mathcal{A}_j$, we have that:

$$\delta_j + \sum_{S\in\mathcal{S}} \sum_{I\in A_j(S)} \delta(I,S) = \sum_{S\in\mathcal{S}} \bar{u}_j(S) = p_j^{\mathrm{I}} \tag{10}$$

By combining Equation (9) and Equation (10), we obtain

$$\begin{aligned}
\sum_{j\in\mathcal{N}} \left( \delta_j + \sum_{\substack{S\in\mathcal{S}\\ I\in A_j(S)}} \delta(I,S) \right) &= \sum_{S\in\mathcal{S}} \sum_{\substack{j\in\mathcal{N}:\\ \bar{x}_j=0}} \sum_{I\in A_j(S)} \delta(I,S) + \sum_{\substack{j\in\mathcal{N}:\\ \bar{x}_j=1}} \left( \delta_j + \sum_{\substack{S\in\mathcal{S}\\ I\in A_j(S)}} \delta(I,S) \right) \\
&\leq \sum_{S\in\mathcal{S}} \sum_{\substack{j\in\mathcal{N}:\\ \bar{x}_j=0}} \sum_{I\in A_j(S)} q_S p_j^{\mathrm{II}}(I,S) \bar{y}_j(I,S) + \sum_{\substack{j\in\mathcal{N}:\\ \bar{x}_j=1}} p_j^{\mathrm{I}} \\
&\leq \sum_{j\in\mathcal{N}} p_j^{\mathrm{I}} \bar{x}_j + \sum_{j\in\mathcal{N}} \sum_{\substack{S\in\mathcal{S}\\ I\in A_j(S)}} q(S) p_j^{\mathrm{II}}(I,S) \bar{y}_j(I,S) \qquad (11)
\end{aligned}$$

Since the initial value of each dual variable is zero, and each instance $I \in A_j(S)$ increases the total value of the dual variables by at most $2\delta(I,S)$, we can sum over all such $\delta$ to bound the total value of the dual variables:

$$\sum_{j\in\mathcal{N}} \sum_{S\in\mathcal{S}} \bar{u}_j(S) + \sum_{S\in\mathcal{S}} \sum_{t\in\mathcal{T}} \bar{v}_t(S) \leq 2 \left( \sum_{j\in\mathcal{N}} \left( \delta_j + \sum_{S\in\mathcal{S}} \sum_{I\in A_j(S)} \delta(I,S) \right) \right) \tag{12}$$

Equations (11) and (12), together with the weak duality theorem, immediately imply the claimed result.

## 4 An algorithm for the black box model

We show next that we can adapt the algorithm derived in the previous section for the polynomial scenario setting to the black box model, where the probability distribution is specified only by allowing access to an oracle from which independent samples according the distribution can be drawn. We show that applying the previous algorithm to an approximate version of the distribution based on sampling can be shown to still yield the same performance guarantee. Our analysis uses the structure of the analysis used for the previous algorithm, and builds on the general result for minimization 2-stage stochastic problems derived by Charikar, Chekuri, and Pál [2].

We shall make use of the following version of the Chernoff bound.

**Lemma 1.** *Let $X_1, \dots X_N$ be independent random variables with $X_i \in [0,1]$ and let $X = \sum_{i=1}^N X_i$. Then, for any $\epsilon \geq 0$, we have $Pr\left[|X - E[X]| > \epsilon N\right] \leq 2\, exp(-\epsilon^2 N)$.*

We assume that there is an inflation factor $\lambda \geq 1$ such that $p_j^{\mathrm{II}}(I,S) \leq \lambda p_j^{\mathrm{I}}$, $\forall j \in \mathcal{N}$, $\forall S \in \mathcal{S}$, $\forall I \in A_j(S)$.

The algorithm first takes a polynomial-sized sample from the set of scenarios and then proceeds just as the Algorithm 1 in Section 3 while using a slightly different deferring rule.

More precisely, it takes $N = \Theta(\frac{\lambda^2}{\epsilon^2} \log \frac{n}{\gamma})$ independent random samples $S_1, \dots, S_N$ from the black box, where $n$ is the number of activities, $\epsilon$ will be the allowed additional relative error, and $\gamma$ is the confidence parameter (that is, we shall obtain that the desired approximation is found with probability at least $1 - \gamma$). Then the algorithm executes the pushing procedure (see Algorithm 1) for each scenario that occurs in the polynomial sample. Observe that the data used by this algorithm for scenario $S$ is described to be $q(S) p_j^{\mathrm{II}}(I, S)$. At first glance, this might be worrying, but of course the value $q(S)$ is just a uniform scalar multiple for all profits, and so it makes sense to define $\tilde{u}$ and $\tilde{v}$ as the dual variables computed after executing this algorithm with inputs $p_j^{\mathrm{II}}(I, S)$. Observe that the values $\bar{u}$ and $\bar{v}$ for a scenario $S$ from our exact distribution are equal to $q(S)\tilde{u}$ and $q(S)\tilde{v}$, respectively. Given $\epsilon > 0$, we shall defers an activity $\mathcal{A}_j$, $j \in \mathcal{N}$, if and only if:

$$(1 + \epsilon) p_j^{\mathrm{I}} \geq \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_j(S_i) \tag{13}$$

This is the deferring rule for the black box model.

This concludes the description of the first stage action. For the second stage, for a given scenario $S \in \mathcal{S}$, we execute Algorithm 2 for scenario $S$. (Again, note that the linearity effect of $q(S)$ implies that we can run the algorithm with inputs $p_j^{\mathrm{II}}(I, S)$ instead.)

Let us analyze the performance guarantee of this algorithm. The proof proceeds by showing that, under the assumption that there is an inflation factor $\lambda$, equation (13) is a good approximation for equation (7). This approach is inspired by the proof in [2] for "low scenarios".

**Theorem 2.** *For any $\epsilon > 0$ and $\gamma > 0$, with probability at least $1 - \gamma$, the proposed deferring rule is a $(2 + \epsilon)$-approximation algorithm for the 2-stage stochastic variant of the problem $1|r_j|\sum w_j U_j$ in the black box model.*

*Proof.* Suppose we run Algorithm 1 in each of the exponentially-many scenarios and let $\bar{u}$ and $\bar{v}$ be the value of dual variables computed in this way. Consider activity $\mathcal{A}_j$. Let

$$r = \sum_{S \in \mathcal{S}} \bar{u}_j(S) = \sum_{S \in \mathcal{S}} q(S) \tilde{u}_j(S) \qquad \hat{r} = \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_j(S_i).$$

We will prove that, with "high" probability, $\hat{r}$ is "close" to $r$. We can view $\hat{r}$ as the arithmetic mean of $N$ independent copies $Q_1, \dots, Q_N$ of the random variable $Q$ defined as

$$Q = \tilde{u}_j(S).$$

Note that $E[Q] = r$. Let $Y_i$ be the variable $Q_i/M$ where $M = \lambda p_j^{\mathrm{I}}$ and let $Y = \sum_i Y_i$. Note that for each activity $\mathcal{A}_j$ and for each scenario $S \in \mathcal{S}$, there exists some $I \in A_j(S)$ such that $\tilde{u}_j(S) \leq p_j^{\mathrm{II}}$. This implies that $Y_i \in [0, 1]$. Moreover, $Y = \sum_i Q_i/M = \frac{N}{M}\hat{r}$

and $E[Y] = \sum_i E[Q_i]/M = \frac{N}{M}r$. By applying the Chernoff bound, we obtain the following:

$$Pr\left[|Y - E[Y]| > \frac{\epsilon}{\lambda}N\right] \leq 2\exp\left(-\frac{\epsilon^2}{\lambda^2}N\right) \Leftrightarrow Pr\left[|r - \hat{r}| > \epsilon p_j^{\mathrm{I}}\right] \leq \frac{\gamma}{n}, \qquad (14)$$

where the last inequality follows from the choice of the value of $N$. By taking the union bound over all activities, we obtain that $r$ is "close" to $\hat{r}$ for all activities, with probability at least $1 - \gamma$.

We use the same argument as we used in the polynomial scenario model to show that constraint (5) is satisfied. Consider constraint (4) for some scenario; it may be violated by any activity. We show that it is satisfied, with high probability, by a non-deferred activity. For a deferred activity, we shall increasing the value of its dual variables, as we did in the polynomial scenario model so that the corresponding constraint is also satisfied with high probability. (It is important to note that this increase in the dual variables is not performed by the algorithm; it is only used for the analysis.)

For each deferred activity $\mathcal{A}_j$, let

$$\delta_j = p_j^{\mathrm{I}} - \sum_{S \in \mathcal{S}} \bar{u}_j(S) \quad j = 1, \ldots, \mathcal{N}$$

and let $\overline{S} \in \mathcal{S}$ be an arbitrarily selected scenario. We increase the value of $\bar{u}_j(\overline{S})$ by $\delta_j$ for each deferred activity $\mathcal{A}_j$. From the fact that $r$ is a good approximation of $\hat{r}$, it follows that, for each activity $\mathcal{A}_j$, if

$$\frac{1}{N}\sum_{i=1}^{N} \tilde{u}_j(S_i) \leq (1+\epsilon)p_j^{\mathrm{I}},$$

then with probability at least $1 - \gamma$,

$$\sum_{S \in \mathcal{S}} \bar{u}_j(S) \leq (1 + 2\epsilon)p_j^{\mathrm{I}}. \qquad (15)$$

This implies that with high probability, for each deferred activity $\mathcal{A}_j$

$$\delta_j + \sum_{S \in \mathcal{S}} \sum_{I \in A_j(S)} \delta(I, S) = \sum_{S \in \mathcal{S}} \bar{u}_j(S) \leq (1 + 2\epsilon)p_j^{\mathrm{I}} \qquad (16)$$

In a similar way, if for an activity $\mathcal{A}_j$

$$\frac{1}{N}\sum_{i=1}^{N} \bar{u}_j(S_i) > (1+\epsilon)p_j^{\mathrm{I}}$$

then with probability at least $1 - \gamma$, it follows that

$$\sum_{S \in \mathcal{S}} \bar{u}_j(S) > p_j^{\mathrm{I}}.$$

Hence, the new solution is dual feasible with high probability. Note that Equation (16) is an approximation to Equation (10). This implies that by replacing this new equation in the previous proof we obtain

$$\sum_{j\in\mathcal{N}}\sum_{S\in\mathcal{S}}\bar{u}_j(S)+\sum_{S\in\mathcal{S}}\sum_{t\in\mathcal{T}}\bar{v}_t(S)\leq 2(1+2\epsilon)\sum_{j\in\mathcal{N}}p_j^{\mathrm{I}}\bar{x}_j+ \\ +2(1+2\epsilon)\sum_{j\in\mathcal{N}}\sum_{S\in\mathcal{S}}\sum_{I\in A_j(S)}q(S)p_j^{\mathrm{II}}(I,S)\bar{y}_j(I,S), \tag{17}$$

which completes the proof.

## 5 An $\mathcal{NP}$-hardness of approximation result

We show that, in contrast to the results of the previous sections, another natural 2-stage stochastic generalization of the problem $1|r_j|\sum w_jU_j$ (even in a very simple case) can not be approximated. Suppose that in the first phase, we select a set of activities that we are committed to serve. In the second phase, for a given scenario, we must schedule exactly one instance of each activity selected in the first phase, and we may augment this solution by scheduling other instances of additional activities. We wish to maximize is the total expected profit (where it is now natural to assume that the profit obtained for an instance in the second phase is less than the corresponding profit in the first). We will refer to this problem as the *augmentation* 2*-stage stochastic* $1|r_j|\sum w_jU_j$.

An integer programming formulation for this problem is obtained by changing (SIP) in the following way: a 0-1 variable $x_j$ indicates (with value 1) that activity $\mathcal{A}_j$ is selected in the first phase; constraint (1) is replaced by the following two constraints:

$$\sum_{I\in A_j(S)} y_j(I,S)\geq x_j \qquad \forall S\in\mathcal{S}, j\in\mathcal{N}: A_j(S)\neq\emptyset \tag{18}$$

$$\sum_{I\in A_j(S)} y_j(I,S)\leq 1 \qquad \forall j\in\mathcal{N}, S\in\mathcal{S} \tag{19}$$

Unfortunately, it is straightforward to show that selecting a feasible set of activities in the first phase can be used to model the maximum independent set problem. This is formalized in the following lemma.

**Lemma 2.** *If there is a $\rho$-approximation algorithm for the augmentation* 2*-stage stochastic* $1|r_j|\sum w_jU_j$*, then there is a $\rho$-approximation algorithm for maximum independent set problem.*

**Proof Sketch.** We give an approximation-preserving reduction from the maximum independent set problem. Given a graph $G$, we build the following input for the augmentation 2-stage stochastic $1|r_j|\sum w_jU_j$. For each vertex $v_j$, there is an activity $\mathcal{A}_j$, $j=1,\ldots,n$, each activity is always released at time 0, has deadline time 1, and takes one time unit to complete; each activity has first-stage profit 1, and second-stage profit

0. For each edge $e_i = (v_j, v_k)$, there is a scenario $S_i$ in which only the activities $\mathcal{A}_j$ and $\mathcal{A}_k$ are active. Each scenario $S_i$ occurs with positive probability, and hence our first stage selection must contain at most one of the endpoints of $e_i$. Thus, there is a one-to-one correspondence between independent sets in $G$ and feasible first-stage decisions. Furthermore, the objective function value of any first-stage decision is exactly the number of activities selected (since the second stage does not contribute any expected profit). Hence, we see that the two optimization problems are identical. ■

From Lemma (2) and the result in [6] we obtain the following theorem.

**Theorem 3.** *For any $\epsilon > 0$, there does not exist a polynomial-time algorithm that approximates the augmentation $2$-stage stochastic $1|r_j|\sum w_j U_j$ within a factor $n^{1/2-\epsilon}$, unless $\mathcal{P} = \mathcal{NP}$.*

## References

1. A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48:1069–1090, 2001.
2. M. Charikar, C. Chekuri, and M. Pál. Sampling bounds for stochastic optimization. In *Proceedings of APPROX-RANDOM 2005*, pages 257–269, 2005.
3. S. Dye, L. Stougie, and A. Tomasgard. The stochastic single resource service-provision problem. *Naval Research Logistics*, 50:869–887, 2003.
4. R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discrete Math.*, 5:287–326, 1979.
5. A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 265–274, 2004.
6. J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.
7. N. Immorlica, D. Karger, M. Minkoff, and V. S. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 691–700, 2004.
8. N. Kong and A. J. Schaefer. A factor 1/2 approximation algorithm for two-stage stochastic matching problems. *European Journal of Operational Research*, 172:740–746, 2006.
9. R. Ravi and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In D. Bienstock and G. Nemhauser, editors, *Integer Programming and Combinatorial Optimization: 10th International IPCO Conference*, number 3064 in Lecture Notes in Computer Science, pages 101–115. Springer-Verlag, 2004.
10. D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science*, pages 228–237, 2004.
11. C. Swamy and D. B. Shmoys. The sampling-based approximation algorithms for multi-stage stochastic optimization. In *Proceedings of the 46th Annual Symposium on Foundations of Computer Science*, pages 357–366, 2005.