The Computational Complexity of Stochastic Optimization

Cassio Polpo de Campos[4], Georgios Stamoulis[1,3], and Dennis Weyland[1,2]
(cassio@idsia.ch, stamoulis.georgios@gmail.com,
dennisweyland@gmail.com)

[1]Università della Svizzera italiana – Lugano, Switzerland
[2]Università degli Studi di Brescia – Brescia, Italy
[3]Lamsade, Université Paris Dauphine – Paris, France
[4]Dalle Molle Institute for Artificial Intelligence – Lugano, Switzerland

Abstract. This paper presents an investigation on the computational complexity of stochastic optimization problems. We discuss a scenario-based model which captures the important classes of two-stage stochastic combinatorial optimization, two-stage stochastic linear programming, and two-stage stochastic integer linear programming. This model can also be used to handle chance constraints, which are used in many stochastic optimization problems. We derive general upper bounds for the complexity of computational problems related to this model, which hold under very mild conditions. Additionally, we show that these upper bounds are matched for some stochastic combinatorial optimization problems arising in the field of transportation and logistics.

Keywords: stochastic combinatorial optimization, computational complexity, chance constraints, stochastic vehicle routing

1 Introduction

Stochastic optimization problems have received increasing attention in recent years. While these problems are used extensively in practice, our theoretical understanding of their complexity is far from complete. Hardness results have been obtained in the context of two-stage stochastic linear programming and two-stage stochastic integer linear programming [3]. It has been shown that just the evaluation of the objective function is already #P-hard, where #P denotes the famous class of counting problems originally introduced in [7]. The same hardness results could have been derived for the corresponding decision and optimization variants. Similar lower bounds have been obtained in the context of stochastic combinatorial optimization problems for a widely used stochastic vehicle routing problem [8]. Analogously, it can be shown that the evaluation of stochastic/chance constraints is #P-hard as well. On the other hand, we do not have strong upper bounds for the computational complexity of stochastic optimization problems. Some attempts have been done in [3], but the corresponding

results are controversial, since the equality of $P^{\#P}$ and $NP^{\#P}$ is assumed, which is to the best of our knowledge still an open problem.

In this paper we investigate a very general scenario-based model for stochastic optimization problems. This model includes, among others, the above mentioned classes of two-stage stochastic linear programming and two-stage stochastic combinatorial optimization. Our main results are general upper bounds, which hold under very mild assumptions, and lower bounds derived for a very plausible stochastic vehicle routing problem. We show that the evaluation of the objective function is in $FP^{\#P[1]}$ and that the evaluation of constraints is in PP. Furthermore, the decision variant of such problems resides in $NP^{\#P[1]}$ and the optimization variant can be solved with multiple calls to the corresponding decision variant and is therefore in $FP^{NP^{\#P[1]}}$. We then show that these bounds are actually matched by an existing stochastic vehicle routing problem, where the objective function is #P-hard and the decision and optimization variants are both $NP^{\#P[1]}$-hard.

The remaining part of this paper is organized as follows. We start with a discussion of the model used in this paper in Section 2, and explain why this model captures many important stochastic optimization problems. In Section 3 we derive general upper bounds for some computational tasks related to this model, which hold under very mild conditions. We then continue to show that these upper bounds are actually matched for a (non artificial) stochastic vehicle routing problem (Section 4). Finally, we conclude the paper with a short discussion of the results and their implications in Section 5.

2 The Stochastic Optimization Model

We discuss a scenario-based model for stochastic optimization problems that is very general and captures, among others, two-stage stochastic combinatorial optimization, two-stage stochastic linear programming and two-stage stochastic integer linear programming. Additionally, this model can also handle chance constraints which are used in many stochastic optimization problems.

The basic assumption of our model is that we can describe the objective function and the constraints in a scenario-based way. For a given problem instance, we have a set $\mathcal{X}$ of solutions and a set $\mathcal{S}$ of scenarios. There is a mass function $p : \mathcal{S} \to \mathbb{R}^+$ representing the probabilities of the scenarios and a function $k : \mathcal{X} \times \mathcal{S} \to \mathbb{R}^+$ representing the costs of a solution under a specific scenario. The objective function $f : \mathcal{X} \to \mathbb{R}^+$ is then the expectation of the costs over the scenarios, that is, for a given solution $x \in \mathcal{X}$, we have $f(x) = \sum_{s \in \mathcal{S}} p(s)k(x, s)$. Constraints representing simple predicates on the solution space are allowed and divided into the two sets $\mathcal{C}$ and $\mathcal{D}$. Constraints $c \in \mathcal{C}$ are computable predicates $c : \mathcal{X} \to \{\text{false}, \text{true}\}$ and correspond to non-stochastic constraints. Constraints in $\mathcal{D}$ are defined in a similar way as the objective function and correspond to stochastic constraints. Each $d \in \mathcal{D}$ is associated with two functions $p_d : \mathcal{S} \to \mathbb{R}^+$ and $k_d : \mathcal{X} \times \mathcal{S} \to \mathbb{R}$ and a bound $b_d \in \mathbb{R}$. For a given solution $x \in \mathcal{X}$, the predicate d is simply $\sum_{s \in \mathcal{S}} p_d(s)k_d(x, s) \leq b_d$.

For the model to be meaningful, we need the following additional assumptions. First of all, we assume that the input, including all the functions' specifications, is shortly encoded in the sense that the sets $\mathcal{X}$ and $\mathcal{S}$ are of at most exponential size with respect to the input. Additionally, we require that the size of these sets can be efficiently computed and that these two sets can be efficiently enumerated. Furthermore, we require p and k to provide efficiently computable numbers[1]. Analogously, the functions p_d and k_d associated with the constraints $d \in \mathcal{D}$ are required to provide efficiently computable numbers. Finally, the predicates $c \in \mathcal{C}$ are required to be computable in polynomial time.

This model is very powerful and it is easy to verify that it captures the important classes of two-stage stochastic combinatorial optimization, two-stage stochastic linear programming, two-stage stochastic integer linear programming (with continuous variables in the second stage) and chance-constrained programming. For more information about these classes we refer to [1].

3 General Upper Bounds

The relation between stochastic optimization problems [1] and counting problems [7] has proven to be very useful in order to derive results about the computational complexity of stochastic optimization problems [3, 8]. In this work we use the recently introduced framework of weighted counting [2] to derive upper bounds for the computational complexity of our model. For the sake of clarity, we first give an overview about the framework of weighted counting and the results which are needed in the context of this paper. We then discuss the very mild technical condition to our stochastic optimization model and give formulations for the corresponding computational problems. After that, we derive upper bounds for these computational problems using the results about weighted counting.

Weighted counting problems are a natural generalization of conventional counting problems [2]. The computational variant and the decision variant of weighted counting problems can be defined as follows.

Definition 1 (Weighted Counting Problem). *We are given a polynomial p and a function $w : \{0,1\}^\star \times \{0,1\}^\star \to \mathbb{R}$ that can be approximated by a polynomial time (in the size of the first two arguments and the third argument) computable function $v : \{0,1\}^\star \times \{0,1\}^\star \times \mathbb{N} \to \mathbb{Z}$, such that $|w(x,u) - v(x,u,b)/2^b| \leq 2^{-b}$ for all $x \in \{0,1\}^\star, u \in \{0,1\}^\star, b \in \mathbb{N}$. The weighted counting problem associated with p and w is to compute for $x \in \{0,1\}^\star$ the function*

$$f(x) = \sum_{u \in \{0,1\}^{p(|x|)}} w(x,u).$$

[1] That means p and k have one additional input which specifies the number of output bits that are required. In this way p and k are providing the numbers to any desired accuracy in polynomial time with respect to the input and the number of output bits.

Definition 2 (Weighted Counting Problem, Decision Variant). *We are given a weighted counting problem defined by a polynomial p and a function $w : \{0,1\}^\star \times \{0,1\}^\star \to \mathbb{R}$ as well as a threshold value $t \in \mathbb{Q}$. The corresponding decision problem is to decide for $x \in \{0,1\}^\star$ whether $f(x) \geq t$ or not.*

Here the variable x is the "input" and w is a function which assigns an efficiently computable weight to each of the exponentially many values of u. At a first glance, the relationship to our scenario-based model is apparent. The important observation is that we are able to describe the objective function and the stochastic constraints of stochastic optimization problems using the scenario-based model in terms of weighted counting. In fact, the computation of the objective function and the evaluation of the stochastic constraints can be seen as weighted counting problems themselves. By exploiting this fact, we derive upper bounds for the complexity of computational tasks related to stochastic optimization problems. The (slightly adapted) results regarding weighted counting [2] that are important for this work can be stated as follows.

Theorem 1. *We are given a weighted counting problem defined by a polynomial p and a function $w : \{0,1\}^\star \times \{0,1\}^\star \to \mathbb{R}$. If the size of the output (eventually encoded as a fraction) is bounded by a polynomial $q(|x|)$, then the given weighted counting problem is in $FP^{\#P[1]}$.*

Theorem 2. *We are given the decision variant of a weighted counting problem defined by a polynomial p, a function $w : \{0,1\}^\star \times \{0,1\}^\star \to \mathbb{R}$ and a threshold value $t \in \mathbb{Q}$. If the size of the output (of the computational variant, eventually encoded as a fraction) is bounded by a polynomial $q(|x|)$, then the problem is in PP.*

It is evident that in order to use these results we have to make an additional, mild condition to our scenario-based model: We assume that the size of the output of the objective function and the (computational variants of the) constraints is polynomially bounded. That means that we require the output to be limited to a polynomial number of bits and this is indeed an extremely mild condition[2]. With this assumption, we can immediately obtain the following results.

Theorem 3. *The following statements hold for stochastic optimization problems using the scenario-based model.*

(i) The evaluation of the objective function is in $FP^{\#P[1]}$.
(ii) Deciding whether a given solution has costs of at most $t \in \mathbb{Q}$ is in PP.
(iii) The evaluation of a stochastic constraint is in PP.
(iv) Checking the feasibility of a given solution is in PP.

[2] In fact, we can even handle problems which do not fulfill this assumption by truncating the objective function and the constraints after a certain number of at most polynomially many bits. The resulting problem is then a slight perturbation of the original problem which should not incur any difference for practical purposes.

(v) The task of computing the objective function in case the given solution is feasible, or returning some arbitrarily fixed value in case the given solution is not feasible is in $FP^{\#P[1]}$.

Proof. (i), (ii) and (iii) follow directly from the fact that we can write the objective function and the stochastic constraints as weighted counting problems. Since PP is closed under multiple non-adaptive/independent calls [4] and since the constraints can be checked independently, (iv) follows. For (v) we have to combine the computation of the objective function and the verification of the constraints. It is clear that we can perform this task in polynomial time with 2 calls to a #P-oracle. Since these oracle calls are independent, they can be combined into a single call, which shows that the task is in $FP^{\#P[1]}$. □

Using this result for the evaluation of solutions, we can derive the following upper bounds for the optimization and decision variants of stochastic optimization problems using the scenario-based model.

Theorem 4. *We have given a stochastic optimization problem using the scenario-based model and a bound* $t \in \mathbb{Q}$. *The problem to decide whether a solution with costs at most* t *exists or not is in* $NP^{\#P[1]}$.

Proof. Here the idea is to create all possible solutions for the given problem in a nondeterministic way. According to Theorem 3 the objective function and the constraints can be evaluated within the NP machine using only a single call to a #P-oracle. Finally, a solution is accepted if it is feasible and has costs of at most t. That means our $NP^{\#P[1]}$ accepts the input if and only if there is at least one feasible solution which obeys the cost bound. □

Theorem 5. *We have given a stochastic optimization problem using the scenario-based model. The task to compute an optimal solution (if it exists) or to return some arbitrarily fixed value in the case where no feasible solution exists can be solved in polynomial time using an oracle for the decision version. In other words, this problem is in* $FP^{NP^{\#P[1]}}$.

Proof. We have to show that we can solve the optimization variant in polynomial time with oracle access to the decision variant. We start with a binary search to determine the costs of an optimal solution. Since the size of the output of the objective function is polynomially bounded, we can do this with polynomially many oracle calls. Once we know the value of an optimal solution, we perform a second binary search on the set of solutions. This can be done efficiently by dividing the set of solutions that is enumerated in the NP part of the oracle into two parts. We then continue with any of the two sets of solutions that still contains an optimal solution. Since the number of possible solutions is at most exponentially large, this step can also be performed with at most polynomially many oracle calls. □

4 Hardness Results for the Dependent PTSPD

In this section we will complement the general upper bounds for our model of stochastic optimization problems with lower bounds for a specific stochastic vehicle routing problem. It seems fairly easy to prove such lower bounds for artificially created stochastic optimization problems. The strength of our results is based on the fact that we are able to show strong lower bounds for a practical stochastic combinatorial optimization problem.

We focus on the Dependent Probabilistic Traveling Salesman Problem with Deadlines (Dependent PTSPD, [8]). As a generalization of the Probabilistic Traveling Salesman Problem with Deadlines, the objective function is #P-hard [8] and therefore basically matches the lower bound derived in the previous section. The Dependent PTSPD also inherits the #P-hardness for the decision and optimization variants from the PTSPD. We strengthen these hardness results and show that the optimization and decision variants of the Dependent PTSPD are both in fact $NP^{\#P[1]}$-hard. For this purpose we use reductions from the $NP^{\#P[1]}$-complete problem E-MINORITY-SAT (explained later on). We first give the formal definitions of the problems used in this section. After that we present the reduction from E-MINORITY-SAT to the decision variant of the Dependent PTSPD in detail. At the end, we show how this reduction can be modified for the optimization variant of the Dependent PTSPD.

4.1 Problem Definitions

In [6] it has been shown that the problem E-MAJ-SAT is $NP^{\#P[1]}$-complete. For this problem we have given a formula in conjunctive normal form. The variables are partitioned into two sets. The question is if there exists an assignment for the first set of variables such that at least half of the assignments for the second set of variables (together with the assignment for the first set of variables) satisfy the given formula. For our proof we use a variant of this problem called E-MINORITY-SAT, which is defined analogously but asks for *at most* half of the assignments to satisfy the formula.

Problem 1 (E-MINORITY-SAT). We have given a boolean formula over n variables $x_1, x_2, \ldots, x_n$ in conjunctive normal form with m clauses and a number $k \in \{0, 1, \ldots, n\}$. The task is to decide if there exists an assignment of the first k variables, such that at most half of the assignments of the remaining variables (together with the assignment of the first k variables) satisfy the given formula.

E-MINORITY-SAT is $NP^{\#P[1]}$-complete. The proof is analogous to the one for E-MAJ-SAT [6] and makes use of PP closure properties [4].

The formal definition of the Dependent Probabilistic Traveling Salesman Problem (Dependent PTSPD, [8]) is more intricate. Here we refer as V a set of n locations, including the special starting point $v_0 \in V$. We have given distances / travel times between the locations which are represented by a function $d : V \times V \to \mathbb{Q}^+$. Deadlines for different customers are modeled using a function

$t : V \to \mathbb{Q}^+$ and penalty values for different customers are modeled using a function $h : V \to \mathbb{Q}^+$. For simplicity we also define these values for the starting point v_0, although we meet the deadline at v_0 nevertheless, since we start the tour there. Additionally, the presence of customers is modeled in a stochastic way. We allow certain kinds of dependencies between the presence of different customers. Two customers $v_i, v_j \in V$ can be bonded in the following way: (1) the presence of v_i and v_j is independent, (2) v_i is present if and only if v_j is present, or (3) v_i is present if and only if v_j is absent. These dependencies can be efficiently modeled by defining sets of paired customers and their associated bonds. We avoid further details for the sake of clarity, since we explicitly point out the necessary dependencies used in our reduction. The probabilities for the customers' presence are represented by a function $p : V \to [0, 1]$. Obviously, p is assumed to respect the dependencies.

A solution can now be represented by a permutation $\tau : [n] \to V$ with $\tau_1 = v_0$. For a specific realization of the customers' presence we use this solution to derive a second-stage solution just by skipping customers which are absent. The costs for the second-stage solution are then the sum of the travel times plus the penalties for missed deadlines. We assume that a customer specific fixed penalty of $h(v)$ for customer $v \in V$ occurs in case the deadline is missed, independently of the actual delay. The costs for a solution τ are the expected costs of the second-stage solutions derived from τ over the different realizations of the customers' presence.

Let $\tau : [n] \to V$ with $\tau_1 = v_0$ be a solution. For all $v \in V$, let A_v be a random variable indicating the arrival time at customer v. Since the travel times of the second-stage solutions are identical to those of the Probabilistic Traveling Salesman Problem (PTSP, [5, 9]), the costs of τ can be expressed as

$$f_{\text{ptspd}}(\tau) = f_{\text{ptsp}}(\tau) + \sum_{i=1}^{n} \Pr(A_{\tau_i} \geq t(\tau_i))\, h(\tau_i).$$

The first part of the costs is the (polynomial time computable) objective function of the PTSP and represents the expected travel times over the second-stage solutions. The second part represents the penalties for missed deadlines. With this expression for the costs of a solution, we define the decision and optimization variants of the Dependent PTSPD in the following way.

Problem 2 (Dependent PTSPD - Decision Variant). Given a set V of size n with a special element $v_0 \in V$, a function $d : V \times V \to \mathbb{Q}^+$, sets of pairs of V defining the customers' bonds, a function $p : V \to [0, 1]$ respecting the dependencies imposed by the partition, a function $t : V \to \mathbb{Q}^+$, a function $h : V \to \mathbb{Q}^+$, and a bound $b \in \mathbb{Q}$, the problem is to decide if there exists a solution $\tau : [n] \to V$ with $\tau_1 = v_0$ such that $f_{\text{ptspd}}(\tau) \leq b$.

Problem 3 (Dependent PTSPD - Optimization Variant). Given a set V of size n with a special element $v_0 \in V$, a function $d : V \times V \to \mathbb{Q}^+$, sets of pairs

of V defining the customers' bonds, a function $p : V \to [0, 1]$ respecting the dependencies imposed by the partition, a function $t : V \to \mathbb{Q}^+$ and a function $h : V \to \mathbb{Q}^+$, the problem is to compute a permutation $\tau^\star : [n] \to V$ with $\tau_1^\star = v_0$ such that $f_{\text{ptspd}}(\tau^\star) \leq f_{\text{ptspd}}(\tau)$ for any permutation $\tau : [n] \to V$ with $\tau_1 = v_0$.

4.2 Hardness of the Decision Variant

We now present a reduction from E-MINORITY-SAT to the decision variant of the Dependent PTSPD. First, we give the general idea behind the reduction, and then we show step by step how for any given instance of E-MINORITY-SAT an instance of the Dependent PTSPD can be constructed. Based on this construction we conclude that the Dependent PTSPD is $NP^{\#P[1]}$-hard. Later we show how this reduction can be modified to obtain the same hardness result for the optimization variant of the Dependent PTSPD.

The overall idea is to construct, for a given E-MINORITY-SAT instance, a highly restricted instance for the decision variant Dependent PTSPD. By highly restricted we mean that deadlines are used to allow only certain paths to appear in an optimal solution. We then simulate the NP decision process, that is, we simulate the assignment for the first set of variables, by using a gadget for each variable which allows an optimal solution to take one of two possible paths. By using customers which are present with a probability of 1/2 and with properly defined dependencies, we create "copies" of these assignments. The second set of variables from the E-MINORITY-SAT instance is modeled by customers which are present with a probability of 1/2. In this case proper dependencies can be used to directly create "copies" of these customers. We then build a new collection of gadgets for the clauses of the formula. The idea is that an assignment which satisfies the formula leads to a certain (controlled) delay. At the end, a special customer is added. The probability for a deadline violation at this customer in an optimal solution enables us to infer the minimum (over the assignments of the first set of variables) number of assignments for the second set of variables that satisfy the clauses. Finally, the cost of an optimal solution for the constructed instance allows us to infer the probability with which the deadline is actually violated in an optimal solution, and this solves the given E-MINORITY-SAT instance.

Simulating the assignment of the first k variables: For the simulation of the NP decision process, we use the gadget depicted in Figure 1. The distance function used here (and also throughout the whole instance) is the Euclidean distance. The starting customer here is v_0. The presence probabilities are 1 for all customers except x_1 and $\bar{x_1}$. The presence probabilities for these two customers are 1/2 and additionally x_1 is present if and only if $\bar{x_1}$ is absent[3].

[3] We use the following convention for all the x customers in the reduction: A group of customers with the same label is either completely present or none of those customers is present. Customers with a label of x_i are present if and only if the customers with the label $\bar{x_i}$ are absent.

By using large enough penalty values and appropriate deadlines, we can force a solution to take one of the following two paths: $v_0, \text{true}, v_1, x_i, \bar{x_i}, \text{false}, v_2$ or $v_0, \text{false}, v_1, x_i, \bar{x_i}, \text{true}, v_2$[4]. The first path corresponds to an assignment of *true* for the corresponding x_i from the E-MINORITY-SAT formula, while the second corresponds to an assignment of *false* for x_i. The goal of the stochastic customers is to guess such chosen assignment. In case the guess is wrong, the travel time within this gadget is by a value of $d(x_i, \bar{x_i}) = \varepsilon$ larger than if the guess would be correct. The reason why we are guessing the chosen assignment is that we can use dependencies between stochastic customers to create "copies" which we require at later stages of the construction to check the clauses. The drawback is that we have to handle the case in which the guess is wrong, but this is not an issue as we will see soon.

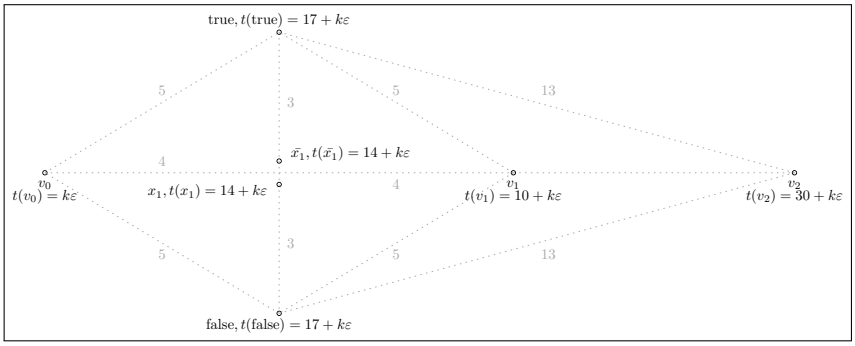

Fig. 1. The gadget used to simulate the variable assignment.

It is clear that we can put k of these gadgets in a row to simulate the assignment of the first k variables. Since we might have some additional delay of ε for each of the gadgets, the deadlines all contain some sort of slack. As long as this distance is small enough compared to the other distances in the gadget, this does not change the overall characteristic of possible optimal paths.[5]

The situation now is as follows. A solution in which the gadgets are visited one after another and in which one of the two paths is chosen for each of the gadgets is always better (of lower cost) than a solution which does not visit the customers in this order. This just follows from the fact that we can use sufficiently large penalty values for the involved customers. Additionally, a realization of the presence for the customers $x_1, x_2, \ldots, x_k$ (and therefore also for the customers $\bar{x_1}, \bar{x_2}, \ldots, \bar{x_k}$) might result in a delay. If the realization correctly guesses the assignment chosen by the path, the delay is 0, otherwise the delay is at least ε.

[4] In fact, there are four and not two possible paths, since the order of x_i and $\bar{x_i}$ can be changed without affecting the quality of the solution, but they are analogous.

[5] Additionally, we save some travel time going from v_1 over x_1 / $\bar{x_1}$ to true / false. Again, this does not incur in relevant changes, for the very same reasons.

Verifying the clauses: To verify the clauses' satisfiability, we build gadgets as depicted in Figure 2. Here we illustrate the construction for the clause $x_1 \vee \bar{x_3} \vee \bar{x_4} \vee x_7$. The two customers w_0 and w_1 on the bottom part have to be always visited, while the customers on the top part correspond to the literals used in this particular clause. A separate customer is used for each literal, sharing the same position with the other literal customers. Their presence depends on the actual variable assignment. We set the deadlines of all the customers in such a way that they have to be visited from left to right. In case one of the top customers is present and requires to be visited, an additionally travel time of ε/m is required. This corresponds to the case in which the clause is satisfied. Please note that the realization of the x-variables might not correspond to the assignment defined by the paths that were chosen in the first part of the constructed instance, because it might be that the x-variables have not guessed this assignment correctly. We will handle this case soon.

It is clear that we can put m of these gadgets next to each other to check all the clauses given in the E-MINORITY-SAT instance. Here the deadlines of subsequent gadgets have to be adapted accordingly. If the whole formula is satisfied by the realization of the x-variables, we get a delay of ε/m for each of the clauses for a total delay of ε. This means that the total delay after all clause gadgets is at least ε if the realization of the x-variables is not correctly guessing the assignment defined by the chosen paths, or if the guess is correct and the x-variables satisfy the formula. Otherwise, the delay is at most $\varepsilon - \varepsilon/m$.

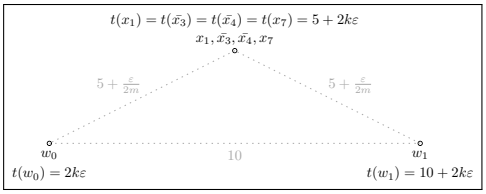

Fig. 2. The gadget used to verify if a clause is satisfied or not.

Putting everything together: We will now place the two sets of gadgets next to each other. Additionally, we will add a special customer z after the clause gadgets. Let T be the arrival time at z without any of the delays. We now use for z a deadline of $t(z) = T + \varepsilon - \varepsilon/m$. In an optimal tour, z is the last customer visited. After visiting z, the vehicle returns to the depot. z is also the only customer whose deadline is violated in an optimal solution. We set the penalty value for z to be $h(z) = 2^{2n}$.

Let us analyze the costs of a solution which does not violate any deadlines except at customer z and which therefore corresponds to an assignment of the first k variables of the original E-MINORITY-SAT instance. There is no need to care about travel times, since the costs are completely dominated by the

penalty at customer z. We are late at customer z if the guess of the decisions that have been made in the first part of the constructed instance is wrong. This happens with a probability of $1 - 2^{-k}$. If our guess is correct, we arrive at the clause part without any additional delay. The deadline at customer z is then violated if and only if all the clauses are satisfied by the correctly guessed first k variables and the randomly assigned remaining $n-k$ variables. Each assignment of the remaining $n-k$ variables occurs with the same probability of 2^{-n+k}. Let r denote the number of satisfying assignments of the remaining $n-k$ variables for the given assignment of the first k variables. We can now write the expected penalty as $(1 - 2^{-k} + 2^{-k}2^{-n+k}r)h(z) = 2^{2n} - 2^{2n-k} + 2^n r$.

If there exists an assignment of the first k variables of the original E-MINORITY-SAT instance such that the formula is satisfied for at most half of the assignments of the remaining $n-k$ variables, then there exists a solution for the Dependent PTSPD instance with costs of at most $2^{2n} - 2^{2n-k} + 2^n 2^{n-k}/2 = 2^{2n} - 2^{2n-k-1}$. On the other hand, if no such solution exists, then every solution for the Dependent PTSPD instance has higher cost. Considering the relatively small travel times, we can set the bound of the constructed instance to $2^{2n} - 2^{2n-k-1} + 2^n - 1$. In this way we are able to solve the original E-MINORITY-SAT problem with the decision version of the Dependent PTSPD.

Theorem 6. *The decision version of the Dependent Probabilistic Traveling Salesman Problem with Deadlines is $NP^{\#P[1]}$-hard, even for Euclidean instances.*

4.3 Hardness of the Optimization Variant

To obtain the same hardness result for the optimization variant of the Dependent PTSPD, we have to modify the construction slightly. The same technique has been used in [8] and therefore we only describe the main ideas at this point. Instead of a single final customer z, we add three customers z_1, z_2, z_3. z_1 is located at the position of z and is forced to be visited immediately after the clause part by imposing proper deadlines and penalties. The three new customers then form an equilateral triangle. This triangle is now placed in a way such that z_2 is closer to the depot than z_3. The deadline which was formerly imposed on z will now be prolonged by the sidelength of the triangle and imposed on z_2. No deadline is imposed on z_3. In this way we offer an optimal solution two choices: visiting at the end of the tour z_1, z_2, z_3 or z_1, z_3, z_2. In the first case the probability that the deadline at z_2 will be violated is lower than in the second case. On the other hand, the travel times are larger in the first case. Placing the triangle in a proper way and using adequate sidelengths, we are able to solve the original E-MINORITY-SAT instance by computing an optimal solution for this modified Dependent PTSPD instance. In case the optimal solution finishes with z_1, z_2, z_3 a solution for the E-MINORITY-SAT instance exists, if the optimal solution finishes with z_1, z_3, z_2, no such solution exists.

Theorem 7. *The optimization version of the Dependent Probabilistic Traveling Salesman Problem with Deadlines is $NP^{\#P[1]}$-hard, even for Euclidean instances.*

5 Discussion and Conclusions

In this paper we have investigated a very powerful scenario-based model of stochastic optimization problems. Using the framework of weighted counting we prove upper bounds for the complexity of various computational tasks related to these problems. Additionally, we show that these upper bounds are matched for a practical existing stochastic vehicle routing problem.

Many stochastic optimization problems inherit NP-hardness from their non-stochastic counterparts, which are usually contained as a special case. We believe that it is possible to obtain much stronger hardness results for a large number of these problems and we hope that our work can help in obtaining such stronger hardness results.

It would also be very interesting to better understand which properties of stochastic optimization problems are actually responsible for their hardness. In particular, it would be interesting to understand if the same hardness results can be obtained for (non artificial) stochastic optimization problems without any dependencies among the random variables. Our work also motivates the usage of approximation algorithms (for both, the objective function and the optimization variant). Objective functions which are #P-hard might still allow for efficient approximations. In this case, the class NP would be an upper bound for the complexity of an approximative version of the decision variant (the upper bound for the optimization variant would change accordingly). On the other hand, inapproximability results for certain stochastic optimization problems could even further strengthen the already existing hardness results.

References

1. J.R. Birge and F.V. Louveaux. *Introduction to stochastic programming*. Springer, 1997.
2. C. de Campos, G. Stamoulis, and D. Weyland. A structured view on weighted counting with relations to quantum computation and applications. Technical report, Electronic Colloquium on Computational Complexity, 2013. TR13-133.
3. M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. *Mathematical Programming*, 106(3):423–432, 2006.
4. L. Fortnow and N. Reingold. PP is closed under truth-table reductions. *Information and Computation*, 124(1):1–6, 1996.
5. P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36(6):929–936, 1988.
6. M.L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9(1):36, 1998.
7. L.G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.
8. D. Weyland, R. Montemanni, and L.M. Gambardella. Hardness results for the probabilistic traveling salesman problem with deadlines. In *Proceedings of ISCO 2012 - The 2nd International Symposium on Combinatorial Optimization*, 2012.
9. D. Weyland, R. Montemanni, and L.M. Gambardella. An improved heuristic for the probabilistic traveling salesman problem with deadlines based on GPGPU. In *Computer Aided Systems Theory-EUROCAST 2013*, pages 332–339. Springer, 2013.