A Sample Average Approximation Approach for Event-Driven Probabilistic Constraint Programming*

Roberto Rossi,[1] S. Armagan Tarim,[2] Brahim Hnich,[3] and Steven Prestwich[2]

Centre for Telecommunication Value-Chain Research (CTVR), Ireland[1]
r.rossi@4c.ucc.ie
Cork Constraint Computation Centre, University College, Cork, Ireland[2]
{at,s.prestwich}@4c.ucc.ie
Faculty of Computer Science, Izmir University of Economics, Turkey[3]
brahim.hnich@ieu.edu.tr

Abstract. In this work we augment a known Monte Carlo simulation-based approach to stochastic discrete optimization problem, the so called Sample Average Approximation (SAA) method, with a new criterion to decide when the search has to be stopped. Our approach exploits a well known and effective sampling technique, Latin Hypercube Sampling (LHS), and confidence interval analysis, a well established approximation method in statistics. We apply SAA augmented with LHS and the new stopping criterion we defined to an Event-Driven Constraint Programming model for scheduling under uncertainty. Our computational experience shows how this technique can not only quickly converge to near optimal solutions by analyzing small sample sets, but also promptly decide when the chances of improving the current optimal solution by analyzing larger sample sets are sufficiently low to justify the interruption of the search process.

1 Introduction

In this paper we propose an effective strategy to compute near-optimal solutions for optimization models where uncertainty comes into play. In particular we focus on Event-Driven Probabilistic Constraint Programming (EDP-CP) models. EDP-CP is a modeling framework proposed by Tarim et al. in [12] that is concerned with the computation of reliable solutions for optimization models under uncertainty. EDP-CP models can be easily compiled into standard Constraint Programming (CP) models, nevertheless solving these model is a hard computational task especially when the set of possible values that random variables may

* The authors wish to thank Mustafa Kemal Dogru and Ulas Ozen, from the supply chain group in Alcatel-Lucent, Ireland, for the discussion on confidence interval analysis that motivated this work. This work was supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 00/PI.1/C075.

assume is large. Furthermore most of the time the set of possible values over which random variables are defined is continuous — for instance one may think about a job completion time in a probabilistic scheduling problem — and in this case the authors in [12] proposed to apply a sampling procedure and reduce the domains of random variables to a finite (and possibly small) set. Unfortunately no general guidelines have been given to decide how big the sampled set should be and how good is the solution provided by this set. In this work we try to answer to these kind of questions by providing an effective solution strategy for EDP-CP models and, in general, for optimization models under uncertainty. Our preliminary results show that this technique is promising.

Our work builds on a framework called Sample Average Approximation (SAA) method. Originally proposed by Kleywegt et al. [7], this method is a Monte Carlo simulation-based approach to stochastic discrete optimization problems. The basic idea of this method is that a random sample is generated and an expected value function is approximated by the corresponding sample average function. The obtained SAA optimization problem is solved and the procedure is repeated several times until a stopping criterion is satisfied. The authors proved this method to be asymptotically complete and in their experiments the method also appears effective, in the sense that it quickly converges to a nearly optimal solution. Unfortunately the stopping criteria based on "optimality gap" estimators proposed by the authors are weak, as they remark in their conclusions, therefore even if the method is clearly effective, typically the sample size needs to be substantially increased to decide when the best solution found is good enough. For this reason the search cannot be promptly stopped as soon as a good solution is reached. Obviously a key factor in the SAA method is to decide when the sample size used is large enough so that by taking a larger set our chances of improving our current solution are low.

In this work, we propose a different strategy to decide when the search has to be stopped. Our method relies on confidence interval analysis, a well established technique in statistics originally introduced by Neyman [9]. This technique tries to determine when, during the search process, the chances of improving the current solution by increasing the sample size fall below a given threshold under a certain confidence probability. This approach in addition to an effective sampling strategy known as Latin Hypercube Sampling — also recognized effective in [7]— provides a valid stopping criterion that is able to limit the size of the sample set analyzed and to limit the number of replications needed to decide that the search has to be stopped. Like the method proposed in [7] our method is asymptotically complete, but in contrast to the SAA method, it lets the user decide what is the confidence level the algorithm should adopt to state that the optimal solution found is unlikely to be improved by searching over a larger sample set. Obviously, by setting a low confidence level we will obtain a low quality solution in a few runs, while a higher confidence level will improve the quality of the solution found and will also increase the time required to decide when the search has to be stopped. Nevertheless, our initial experiments suggest that for reasonable confidence levels (95% or so) the method can quickly converge to very good

solutions (about 98% of the real problem optimum in average) by analyzing very small sample sets typically comprising less than 5 samples.

The paper is structured as follows. In Section 2 we recall the EDP-CP framework. In Section 2.1 we describe the EDP-CP model to which we will apply our solution method. In Section 3 we describe our new confidence interval analysis based SAA method. In Section 4 we show the effectiveness of our approach. In Section 5 we draw conclusions and future extensions.

2 Event-Driven Probabilistic Constraint Programming

In this section we recall the EDP-CP modeling framework defined in [12]. In its most general form, event-driven probabilistic CP supports uncertain parameters as well as decision variables. A constraint is said to be *probabilistic*, if it involves both decision variables and uncertain parameters. We will refer to the possible values of an uncertain parameter λ_i as $W(\lambda_i)$ and to the probability of λ_i taking a given value v in $W(\lambda_i)$ as $\Pr(\lambda_i = v)$. As in [1], we refer to a complete assignment of uncertain parameters as a *possible world* and denote by $\mathcal{W}$ the set of all possible worlds. We also assume that the probability of each possible world w is given by the probability function $\Pr : \mathcal{W} \rightarrow [0, 1]$. Given a probabilistic constraint c over decision variables and some uncertain parameters, the reduction of c by world $w \in \mathcal{W}$, denoted by $c_{\downarrow w}$, is the deterministic constraint obtained by setting all its uncertain parameters as in w.

An EDP-CP is a 9-tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, \Pr \rangle$ where:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of decision variables;
- $\mathcal{D} = D_1 \times \ldots \times D_n$, where D_i is the domain of X_i;
- $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$ is a set of uncertain parameters;
- $\mathcal{W} = W_1 \times \ldots \times W_l$, where W_i the domain of λ_i;
- $\mathcal{E} = \{e_1, \ldots, e_m\}$ is a set of event constraints. Each e_i may either be probabilistic (involving a subset of $\mathcal{X}$ and a subset of Λ) or deterministic (involving only a subset of $\mathcal{X}$);
- $\mathcal{C} = \{c_1, \ldots, c_o\}$ is a set of dependency meta-constraints. For each dependency meta-constraint c_i : DEPENDENCY(e, p, f) we have $e \in \mathcal{E}$, where p may be either a probabilistic or a deterministic pre-requisite constraint, and f is a deterministic condition constraint;
- $\mathcal{H} = \{h_1, \ldots, h_p\}$ is a set of hard constraints. Each h_i may either be probabilistic (involving a subset of $\mathcal{X}$ and a subset of Λ) or deterministic (involving only a subset of $\mathcal{X}$);
- Ψ is any expression involving the event realization measures on the event constraints in $\mathcal{E}$;
- $\Pr : \mathcal{W} \rightarrow [0, 1]$ is a probability distribution over uncertain parameters.

An optimal solution to an EDP-CP $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \Lambda, \mathcal{W}, \mathcal{E}, \mathcal{C}, \mathcal{H}, \Psi, Pr \rangle$ is any assignment S to the decision variables such that for each $h \in \mathcal{H}$, for each $w \in \mathcal{W}$, $h_{\downarrow w}$is satisfied, and there exists no other assignment satisfying all the hard constraints with a strictly better value for Ψ, according to the DEPENDENCY constraints introduced in the model.

Maximize:
$\sum_{i \in I} E\{e_i : s_i + \sum_{m \in M} \pi_{im} * \delta_{im} \leq d_i\}$

Given that:
$\text{DEPENDENCY}(e_i, s_j \geq s_i + \sum_{m \in M} \pi_{im} * \delta_{im}, \sigma_{ij} = 1), \forall i, j \in I, i \neq j$

Subject to:
$s_i \geq r_i, \forall i \in I$
$\sigma_{ij} = 1 \Rightarrow s_i < s_j, \forall i, j \in I, i \neq j$
$\sum_{m \in M} \delta_{im} = 1, \forall i \in I$
$\sigma_{ij} + \sigma_{ji} \geq \delta_{im} + \delta_{jm} - 1, \forall m \in M, \forall i, j \in I, i \neq j$
$\sigma_{ij} + \sigma_{ji} \leq 1, \forall i, j \in I, i \neq j$
$\sum_{i \in I} \left(\sum_{m \in M} c_{im} * \delta_{im}\right) \leq B$

$\sigma_{ij} \in \{0, 1\}, \forall i, j \in I \qquad \delta_{im} \in \{0, 1\}, \forall i \in I, \forall m \in M \qquad s_i \in [L_s, L_e], \forall i \in I$

Fig. 1. An EDP-CP model for the Probabilistic Sequencing with Release Times and Deadlines Problem

2.1 An EDP-CP Model for Probabilistic Sequencing with Release Times and Deadlines

We consider a specific scheduling problem similar to the one considered by Hooker et. al [3]. Garey and Johnson [2] also mention this problem in their list of NP-hard problems and they refer to it as "Sequencing with Release Times and Deadlines". An optimization version of this scheduling problem was also described in [6]. The problem consists in finding a least-cost schedule to process a set of orders I using a set of dissimilar parallel machines M. Processing an order $i \in I$ can only begin after the release date r_i and must be completed at the latest by the due date d_i. Order i can be processed on any of the machines. The processing cost and the processing time of order $i \in I$ on machine $m \in M$ are c_{im} and p_{im}, respectively. The model just described is fully deterministic, but we will now consider a generalization of this problem to the case where some inputs are uncertain. For convenience we will just consider uncertain processing times π_{im} for order $i \in I$ on machine $m \in M$. Nevertheless it is easy to see that EDP-CP can be also employed to model more complicated generalizations of this problem where release dates and due dates are uncertain or processing costs are uncertain. Since EDP-CP is meant to model and optimize the reliability of a given plan we will no longer look, as the author do in [6], for a least-cost plan, rather we will optimize a reliability measure expressed in terms of events, as it is usual in EDP-CP. The specific event whose probability we wish to maximize is the successful completion of each job within the given time frame defined by its release and due date. Since jobs are scheduled in sequence on each machine dependencies will arise between subsequent jobs. We adopt a specific policy that unschedules a job if this is not processed within the given due date or before the planned start time of the subsequent job on the respective machine. This policy guarantees that every order will always start at the planned start time, since the respective machine will be free and will start processing it. An EDP-CP model for the problem described is given in Fig. 1. Let us analyze the given

Maximize:
$\sum_{i \in I} \sum_{w \in W} e_{wi} p_w$

Subject to:
$(\sigma_{ij} = 1 \wedge s_i + \sum_{m \in M} \delta_{im} \pi_{im \downarrow w} > s_j) \Rightarrow e_{wi} = 0, \forall w \in W, \forall i, j \in I, i \neq j$
$s_i + \sum_{m \in M} \delta_{im} \pi_{im} > d_i \Rightarrow e_{wi} = 0, \forall w \in W, \forall i \in I$
$s_i \geq r_i, \forall i \in I$
$\sigma_{ij} = 1 \Rightarrow s_i < s_j, \forall i, j \in I, i \neq j$
$\sum_{m \in M} \delta_{im} = 1, \forall i \in I$
$\sigma_{ij} + \sigma_{ji} \geq \delta_{im} + \delta_{jm} - 1, \forall m \in M, \forall i, j \in I, i \neq j$
$\sigma_{ij} + \sigma_{ji} \leq 1, \forall i, j \in I, i \neq j$
$\sum_{i \in I} \left(\sum_{m \in M} c_{im} * \delta_{im} \right) \leq B$

$e_{wi} \in \{0, 1\}, \forall w \in W, i \in I \quad \sigma_{ij} \in \{0, 1\}, \forall i, j \in I \quad \delta_{im} \in \{0, 1\}, \forall i \in I, \forall m \in M \quad s_i \in [L_s, L_e], \forall i \in I$

Fig. 2. A CP equivalent model for the Probabilistic Sequencing with Release Times and Deadlines Problem

model. There are three sets of decision variables: δ_{im} is 1 iff job i is scheduled on machine m; σ_{ij} is 1 iff job i precedes job j; and s_i is the start time of job i. The objective function maximizes the expected number of tasks completed by the respective due dates. DEPENDENCY constraint states that, when two jobs i, j are executed in sequence on the same machine (condition $\sigma_{ij} = 1$), job i has to be completed by its due date (event e_i is satisfied) and before the start time of job j (pre-requisite $s_j \geq s_i + \sum_{m \in M} \pi_{im} * \delta_{im}$). The hard constraints respectively state that: the start time of job i, s_i, must be no less than the release time r_i for this job; if job i is processed before job j ($\sigma_{ij} = 1$), then the start time of i, s_i, must be less than that of j, s_j; each job must be processed on a machine; if two jobs i, j are processed on the same machine m ($\delta_{im} = 1$ and $\delta_{jm} = 1$), either i is processed before j, $\sigma_{ij} = 1$, or j is processed before i, $\sigma_{ji} = 1$; the processing costs must be no greater than the given budget B. We define $L_s = \min_{i \in I} r_i$ and $L_e = L_s + \min_{m \in M} \sum_{i \in I} \lceil \pi_{im} \rceil$ where $\lceil \pi_{im} \rceil$ is the maximum duration of order $i \in I$ on machine $m \in M$ for every possible world $w \in W$. Therefore $\lceil \pi_{im} \rceil = \max_{w \in W} \pi_{im}$. As described in [12], EDP-CP models can be compiled into standard CP model and they can be solved employing standard solvers. A CP equivalent model for the EDP-CP model presented in Fig. 1 is given in Fig. 2. In order to solve the EDP-CP model by employing a standard CP approach we introduced a new set of binary variables, e_{wi}, where $i \in I$, and $w \in \mathcal{W}$, where $\mathcal{W}$ is the set of possible worlds defined in Section 2. We recall that $\pi_{im \downarrow w}$ is the processing time of job i on machine m in world w. The DEPENDENCY meta constraint is translated into the first two hard constraints in the CP equivalent model. These two constraints state that when e_{wi} is equal to 1 this means that in world w job i successfully terminates before the next scheduled job and before its deadline. The remaining hard constraints in the EDP-CP model are retained in the CP equivalent model. The objective function is therefore the weighted sum $\sum_{i \in I} \sum_{w \in W} e_{wi} p_w$, where p_w is the probability associated to world $w \in \mathcal{W}$.

3 Sample Average Approximation

In this section we aim to extend the discussion in [12] on the possible techniques that may be applied to reduce the number of possible worlds considered as an input of an EDP-CP model. We will also discuss how the quality of a given solution can be analyzed with respect to this reduction by using confidence interval analysis and how it is possible to efficiently obtain good solutions by solving problems over a fairly small number of worlds. In order to do so we will employ the approach proposed in [7] by Kleywegt et al. known as Sample Average Approximation (SAA). This approach is extremely powerful due to its generality, efficiency and to the quality of the solutions it provides.

3.1 Latin Hypercube Sampling

First we recall the scenario reduction technique described in [12], *Latin Hypercube Sampling* (LHS) [8]. Latin Hypercube Sampling is a stratified random sampling technique in which a sample of size N from multiple random variables is drawn such that for each individual variable the sample is (marginally) maximally stratified. A sample is maximally stratified when the number of strata equals the sample size N and when the probability of falling in each of the strata equals N^{-1}. The number of categories per variable equals the sample size (6), each row or column contains one element, and the width of rows and columns is 1/6. To draw a Latin hypercube sample of size N for K independent variables, the i^{th} sample element for variable j is obtained as $z_{ij} = F_j^{-1}((p_{ij} - \xi_{ij})/N)$, where F_j is the cumulative distribution function of variable j, where, for each $j = 1, \ldots, K$, p_{ij} $(i = 1, \ldots, N)$ is a random permutation of $1, \ldots, N$; ξ_{ij} is $U[0, 1]$, a uniformly distributed random number between 0 and 1; and the K permutations of the NK uniform variates ξ_{ij} are mutually independent. This sampling procedure has been originally chosen for its effectiveness. For instance Stein [11] showed that LHS yields an asymptotic variance that is smaller than those produced by other techniques like Simple Random Sampling, typically used in Monte Carlo analysis. Pebesma and Heuvelink [10] point out that for the input to the Monte Carlo analysis, the usual way to create multiple realizations of a spatial random field is to draw a simple random sample, subsequent realizations are typically drawn independently. Taking a larger sample of realizations usually allows more accurate estimation of the probability distribution function being analyzed. If the run time of each Monte Carlo run is short, taking a larger sample suffices to make the sampling error negligibly small. On the other hand, as it is the case for our approach, when each model run is relatively expensive[1], the sample size has to be kept small for practical reasons. In this case, more accurate assessment of the output probability distribution function can be obtained when more efficient sampling methods, such as stratified random sampling or its multivariate version, LHS, are used. Furthermore a method

[1] Note that in each "run" we need to solve a model involving N worlds, where N is sample size

for drawing Latin hypercube samples from dependent variables was presented by Stein [11], therefore this technique can be effectively applied even when the random variables considered are not mutually independent.

3.2 Stochastic Discrete Optimization with LHS

We consider optimization problems of the form

$$\min_{x \in S} \{g(x) := \mathbb{E}_P G(x, W)\} \tag{1}$$

where W is a random vector having probability distribution P, S is a *finite* set, $G(x, w)$ is a real valued function of two (vectory) variables x and w, and $\mathbb{E}_P G(x, W) = \int G(x, w) P(dw)$ is the corresponding expected value. $g(x)$, the expected value function, is assumed to be well defined.

Equipped with the sampling procedure described in the former section we will now describe how it is possible to efficiently obtain good solutions for this class of problems by using EDP-CP and by employing a technique similar to the one discussed by Kleywegt et al. in [7]. The technique is simple. A Latin hypercube sample of W is generated for the problem considered. The EDP-CP model is solved for the given sample and the procedure is repeated several times until a stopping criterion is satisfied. An asymptotically exact method consists in incrementally increasing the sample size N until a stopping criterion is satisfied, while an approximate approach consists in solving the EDP-CP model for several samples of a given size. Obviously these two alternative methods can be integrated as shown in [7].

Let $\widehat{W} = [W^1, \ldots, W^N]$ be an independently and identically distributed (i.i.d.) random sample of N realizations of the random vector W. The sample average function is

$$\widehat{g}(x) := \frac{1}{N} \sum_{j=1}^{N} G(x, W^j) \tag{2}$$

and the associated problem

$$\min_{x \in S} \widehat{g}(x). \tag{3}$$

We refer to 1 and 3 as the "true" problem and SAA problems, respectively. Let v^* and $\widehat{v}_N$ denote the optimal values

$$v^* := \min_{x \in S} g(x) \quad \text{and} \quad \widehat{v}_N := \min_{x \in S} \widehat{g}(x), \tag{4}$$

of the respective problems. We also consider sets of ϵ-optimal solutions. That is, for $\epsilon > 0$, we say that $\overline{x}$ is an ϵ-optimal solution of 1 if $\overline{x} \in S$ and $g(\overline{x}) \leq v^* + \epsilon$. The sets of all ϵ-optimal solutions of 1 and 3 are denoted by S^ϵ and $\widehat{S}_N^\epsilon$, respectively. For $\epsilon = 0$ the two sets are identical.

The following propositions in [7] state that with probability approaching 1 exponentially fast in the increase of the sample size, an optimal solution of the sampled problem provides an optimal solution of the "true" problem.

Proposition 1 (Kleywegt et al. [7]). *The following two properties hold: (i)* $\widehat{v}_N \to v^*$ *with probability one (w.p.1) as* $N \to \infty$*, and (ii) for any* $\epsilon \geq 0$ *the event* $\{\widehat{S}_N^\epsilon \subset S^\epsilon\}$ *happens w.p.1 for* N *large enough.*

Proposition 2 (Kleywegt et al. [7]). *The probability of event* $\{\widehat{S}_N^\epsilon \subset S^\epsilon\}$ *approaches* 1 *exponentially fast as* $N \to \infty$ *under a mild assumption.*

As the authors remark in [7] this suggests that an effective sampling procedure, combined with an efficient method for solving the deterministic SAA problem, can effectively solve the type of problems considered. Nevertheless if the computational complexity of solving the SAA problem increases faster than linearly in the sample size N, it may be more efficient to choose a small sample size N and solve several SAA problems with i.i.d. samples. One can view such a procedure as Bernoulli trials with probability of success $p = p(N)$. Here "success" means that a calculated optimal solution $\widehat{x}_N$ of the SAA problem is an optimal solution for the true problem. Obviously, from the propositions above this probability p tends to 1 and $N \to \infty$, and, moreover, it tends to 1 exponentially fast under a mild assumption. However, for a finite N, p can be small or even zero. The probability of producing an optimal solution of the true problem at least once in M trials is $1 - (1 - p)^M$, and this probability tends to 1 as $M \to \infty$, provided p is positive. Unfortunately the authors also point out that choosing a minimum size N that guarantees $p > 0$ is a hard problem specific task. In [7] the authors also describe some possible "optimality gap" estimators that can be used to define a stopping criterion for the optimization process both in the exact and in the approximate case. They emphasize that finding good optimality gap estimators is again a hard problem dependent task. Moreover the stopping criteria based on "optimality gap" estimators proposed by the authors are weak, as they remark in their conclusions, therefore even if their SAA method is clearly effective[2], typically the sample size needs to be substantially increased to decide when the best solution found is good enough. For this reason the search cannot be promptly stopped as soon as a good solution is reached. Here we suggest a different approach to decide when a solution is good enough. Our approach relies on confidence interval analysis, which we now introduce.

3.3 Confidence Interval Analysis

Originally introduced by Neyman [9], confidence interval analysis is a well established technique in statistics [13]. In the years several techniques for building confidence intervals for a given sample set have been proposed. Here we will focus on the exact methods to build these intervals for normally distributed random variables. Obviously, under the assumptions stated by the Central Limit Theorem (CLT) [13], this approach can also be adopted when the the distribution of the random variables is unknown. The basic idea underlying confidence

[2] In their experiments it quickly converges to a nearly optimal solution by analyzing only a few samples

interval analysis is as follows. Every confidence interval is build on the top of a condition that gives the satisfaction probability of some inequalities concerning a given estimated value $\tilde{a}$. In the general case the distribution law of $\tilde{a}$ depends on the parameters of the probability distribution function for the unknown random variable X being analyzed, therefore analyzing this distribution law is difficult. Nevertheless in some cases it is possible to switch from the random variable $\tilde{a}$ to another function of the observed realizations $X_1, \ldots, X_N$ whose distribution law depends only on the number of realizations observed N and on the distribution law of X (i.e. normal distribution, poisson distribution etc.), but not on the unknown parameters of such a distribution. The theory of confidence intervals is of great importance in statistics, for instance it has been proved that if X is a random variable normally distributed, then the random variable $T = \sqrt{N}\frac{\widetilde{m}-X}{\sqrt{\widetilde{D}}}$, where $\widetilde{m} = \frac{\sum_{i=1}^{N} X_i}{N}$ and $\widetilde{D} = \frac{\sum_{i=1}^{N}(X_i-\widetilde{m})^2}{N-1}$ is distributed according to Student's distribution with $N-1$ degrees of freedom. We shall now see how this distribution is applied to build the confidence intervals for $\tilde{m}$, that is our sample mean. By computing the confidence interval for the sample mean $\tilde{m}$ we are looking for a value ϵ_α such that $\Pr\{|\widetilde{m} - X| < \epsilon_\alpha\} = \alpha$. Let t_α the value of the Inverse Student's T distribution for a confidence probability α and $N-1$ degrees of freedom. By setting $\epsilon_\alpha = t_\alpha\sqrt{\frac{\widetilde{D}}{N}}$, we find half of the length of the confidence interval I_α and therefore the interval $I_\alpha = \left(\widetilde{m} - t_\alpha\sqrt{\frac{\widetilde{D}}{N}}, \widetilde{m} + t_\alpha\sqrt{\frac{\widetilde{D}}{N}}\right)$. A similar approach can be used to build these intervals also for $\widetilde{D}$, that is the computed sample variance. In this case confidence interval I_α for the sample variance $\widetilde{D}$ can be computed as $I_\alpha = \left(\frac{\tilde{D}(N-1)}{\chi_1^2}, \frac{\tilde{D}(N-1)}{\chi_2^2}\right)$, where χ_1^2 is the value of the inverse χ^2 distribution [13] for $N-1$ degrees of freedom and a probability $\frac{1-\alpha}{2}$, χ_2^2 is the value of the inverse χ^2 distribution for $N-1$ degrees of freedom and a probability $1 - \frac{1-\alpha}{2}$.

3.4 Stopping criteria for SAA based on confidence intervals

It is possible to apply the theory presented in Section3.3 to define effective stopping criteria. Confidence interval analysis can be extremely useful for making dynamic decisions related to the search process for a solution that is optimal under certain criteria. As a consequence of the considerations on Bernoulli trials in Section 3.2 and from the CLT, intuitively it should be possible, in principle, to solve an SAA problem over a very small sample set for several times and to eventually generate an optimal solution. As the authors remark in [7] this is what happens in practice when we look at the experiments. Nevertheless in their conclusions the author remarked that their stopping criterion performs poorly and therefore their approach keeps looking for a solution even when the chances of improving the current one are very low. In what follows we will define an effective approach to decide when to stop the search. Our approach is able to converge to "sufficiently" optimal solutions by using fairly small sample sizes.

(i) solution quality criterion A first stopping criterion that has to be defined is the following. Since solving a SAA problem with large $N = |\widehat{W}|$ is computationally expensive, we usually consider SAA problems where, in the sample average function $\widehat{g}(x)$, the sample set size N is small. It directly follows that the objective function value $\widehat{v}_N$ obtained may not be representative enough for the actual quality of the solution obtained. If the expected value function $g(x)$ is discrete, defined over a finite set of worlds $\mathcal{W}$, and if we have a complete knowledge over these worlds, then $g(x)$ for the optimal solution x of the SAA problem can be easily computed. Nevertheless in general $g(x)$ is a continuous function. In this case in order to obtain a more representative value $\widehat{v}_{N'}$ that estimates $g(x)$, we can efficiently compute the value of the sample average function $\widehat{g}(x)$ for x over a larger sample set $\widehat{W}'$, where $N' = |\widehat{W}'| > N$. The size of $\widehat{W}'$ can be dynamically determined using the technique described in Section 3.3 when a confidence probability α and a threshold size for the confidence interval I_α over $\widehat{v}_{N'}$ are fixed.

(ii) SAA problem replication number criterion We now know how to dynamically compute a good estimate value $\widetilde{a}$ for $g(x)$ when the optimal solution x for a SAA problem over a given sample set is known. As stated in Section 3.2 we typically aim to repeatedly solve several SAA problems for a given sample size N and look for the solution $\widehat{x}$ with the best estimate value $\widetilde{a}$. Therefore in general we will consider M sample sets $\widehat{W}_1, \ldots, \widehat{W}_M$, where $|\widehat{W}_i| = N$ for $i = 1, \ldots, M$ and we will solve a SAA problem over each sample set $\widehat{W}_i$. A key problem now is to decide the number M of problems that need to be solved in order to have a good chance to produce an ϵ-optimal solution. Let $a_1, \ldots, a_M$ be the estimate values computed for $g(x_1), \ldots, g(x_M)$, where x_i is the optimal solution of the SAA problem over $\widehat{W}_i$. The stopping criterion is simple. We build confidence intervals (for a given confidence probability α) as explained in Section 3.3 for the sample mean and the sample variance of the sample set $a_1, \ldots, a_M$. Let $\lceil \widetilde{m} \rceil$ be the upper limit of the confidence interval for the sample mean and let $\lceil \widetilde{\sigma}^2 \rceil$ be the upper limit of the confidence interval for the sample variance. By (roughly) applying the 3σ rule[3] [13], if the best solution value found so far is greater than $\lceil \widetilde{m} \rceil + 3 * \sqrt{\lceil \widetilde{\sigma}^2 \rceil}$ then with about $\alpha\%$ confidence our search over the current sample size cannot lead to a better solution.

(iii) sample set size criterion Finally, we need a criterion to decide when the sample size has to be increased or the search for an ϵ-optimal solution has to be terminated. This criterion is defined as follows. We assume an increase step s for the number of samples considered and a minimum improvement threshold τ. Let U_N be the value $\lceil \widetilde{m} \rceil + 3 * \sqrt{\lceil \widetilde{\sigma}^2 \rceil}$ computed for a sample size N and U_{N-s} the one for sample size $N - s$. If $U_N - U_{N-s} > \tau$ we increase the sample set size to $N + s$ and we solve a new set of SAA problems for this new sample set size. Otherwise we stop searching and we return the solution $\overline{x}$ with the best-so-far

[3] The 3σ rule states that for a normally distributed random variable with mean m and standard deviation σ the event $\{|X - m| < 3\sigma\}$ can be assumed almost certain, in fact $\Pr\{|X - m| < 3\sigma\} = 0.99730$

Algorithm 1: SAA method with confidence based stopping criteria

```
   input : N: the initial sample size; B: minimum extended sample size; α: confidence
           probability; τ: improvement threshold; s: sample set increase step
   output: x̄: best solution so far
1  begin
2      x̄ ← NULL; v̄ ← −∞; U_{N−s} ← −∞; U_N ← 0;
3      while U_N − U_{N−s} > τ do
4          U_{N−s} = U_N;
5          U_N ← runSampleSize(N, α);
6          N ← N + s;
7  end
```

$\widetilde{a}$. The justification for this criterion directly comes from the fact that, if the inequality $U_N - U_{N-s} > \tau$ does not hold, this means that, by using a sample size N, with $\alpha\%$ confidence we can not sufficiently — with respect to the given threshold τ — improve the solution found by using a smaller sample size $N - s$.

3.5 SAA method with confidence based stopping criteria

We now have all the ingredients to define our solution method. In Algorithm 1 the high level pseudo-code for the proposed procedure is presented. The approach proceeds as follows. We first select an initial sample size N; a confidence probability α that will be used in the computation of confidence intervals; a solution improvement threshold τ; and a step s to increase the sample set. Notice that by means of parameters α and τ, in practice, what we ask is that with a probability α our algorithm must return a solution with a value greater than $v^* - \tau$, where $v*$ is the real optimum of the problem. Therefore τ is problem dependent and has to be properly chosen. As explained in Section 3.2 we generate a sample set of size N (`replicateSampleSize(`N, α`)`, line 2), we solve the associated SAA problem (`replicateSampleSize(`N, α`)`, line 4), then we assess the quality of the solution found, x, with respect to a larger sample set whose size N' (possibly greater or equal to a minimum size B indicated by the user) is decided dynamically — criterion (i) — by means of confidence interval analysis (`replicateSampleSize(`N, α`)`, cycle starting at line 7). If the solution x found has the best estimated value $\tilde{a}$ for $g(x)$ we store x and $\tilde{a}$ (`replicateSampleSize(`N, α`)`, line 15). After N' replications we obtain a sufficiently good estimate $\tilde{a}$ (with respect to the chosen confidence probability) of $g(x)$. We replicate this process M times (`runSampleSize(`N, α`)`, cycle starting at line 4). Again the number of replications M is dynamically computed using confidence interval analysis (`runSampleSize(`N, α`)`, line 11, 12 and 13) as explained in criterion (ii). After M replication we obtain an estimate for the average solution value $\widetilde{A}$ associated to a given sample set size; this value has been computed according to the required confidence level α. This process is repeated (Algorithm 1, line 3) until the last increase in the sample set size does not bring a sufficient improvement — criterion (iii) — in the average solution value $\widetilde{A}$.

Function `replicateSampleSize`(N, α)

```
   input  : N, α
   output: ã
 1 begin
 2     generate N i.i.d. samples Ŵ = [W^1, ..., W^N];
 3     let x be the optimal solution of the SAA problem over Ŵ;
 4     N' ← B;
 5     ã ← 0;
 6     ã̃ ← 0;
 7     while true do
 8         generate N' i.i.d. samples Ŵ = [W^1, ..., W^N'];
 9         for each sample W_j do
10             ã ← ã + G(x, W^j);
11             ã̃ ← ã̃ + G(x, W^j)^2;
12         i ← confidenceInterval(ã, ã̃, N'), that is 2t_α √(ã̃/N' − (ã/N')^2)/(N' − 1);
13         if i ≤ ã(1 − α)/N' then
14             ã ← ã/N';
15             if ã > v̄ then
16                 v̄ ← ã;
17                 x̄ ← x;
18             return;
19         else
20             N' ← N' + 1;
21 end
```

4 Experiments

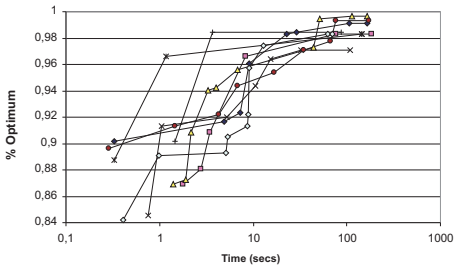

Fig. 4. Comparison among 10 runs for the 7×3 "hard" instance

We will now present some experiments that suggest that our approach can effectively converge to optimal solutions by analyzing sample sets of limited size and by employing only a few replications. In this section we will use a test bed originally presented by Jain and Grossmann for the scheduling problem described in Section 2.1. In [6] the authors consider an optimization version of sequencing with release date and deadlines and provide ten problems, five different sizes with two data sets each (the problems are available in the Online Supplement of the article http://joc.informs.org). As remarked in Section 2.1 we will not optimize processing cost and we will instead optimize the reliability of a plan under a given budget B. Since the input data in [6] are for a deterministic problem we will assume here (without loss of generality) that the job durations provided represent expected values for normally distributed random job lengths, the respective standard deviations can be computed as $\tilde{\pi} c_v$, where $\tilde{\pi}$ is an expected job length and c_v is a given coefficient of variation. In the example here provided we assume $c_v = 0.15$. In order to assess

Function `runSampleSize`(N, α)

```
   input  : N, α
   output: U
 1 begin
 2     S ← {};
 3     M ← 0;
 4     while true do
 5         ã ← replicateSampleSize(N, α);
 6         S ← S ∪ {ã};
 7         M ← M + 1;
 8         if M = 1 then
 9             continue;
10         else
11             [⌊m̃⌋, ⌈m̃⌉] ← sampleMeanConfidenceInterval(S, M)[a];
12             [⌊σ̃²⌋, ⌈σ̃²⌉] ← sampleVarianceConfidenceInterval(S, M);
13             if ⌈m̃⌉ + 3 * √⌈σ̃²⌉ ≤ v̄ then
14                 U ← ⌈m̃⌉ + 3 * √⌈σ̃²⌉;
15                 return;
16 end
```

[a] The confidence intervals for the mean and the variance over the set S of samples can be computed as shown in Section 3.3

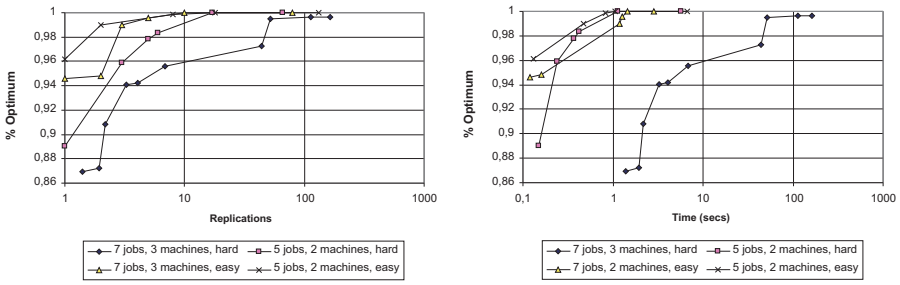

Fig. 3. Solution improvement for the four instances considered

the effectiveness of our approach we consider the following instances based on the set from [6]: two instances having 5 jobs to be scheduled on 2 machines (5×2) and two instances having 7 jobs to be scheduled on 3 machines (7×3). In each pair of instances we consider a "hard" instance and an "easy" one. In the "hard" instance having 7 jobs on 3 machine the release dates for the orders are $[2, 3, 4, 5, 10, 1, 2]$. The due dates are $[16, 13, 21, 28, 24, 28, 23]$. The costs for processing orders on machine M_1 are $[10, 8, 12, 10, 8, 12, 12]$, on machine M_2 they are $[6, 5, 7, 6, 5, 7, 7]$, and on machine M_3 they are $[8, 6, 10, 8, 7, 10, 10]$. The given budget B is 60. The expected processing time of orders on machine M_1 are $[10, 6, 11, 6, 10, 7, 10]$, on machine M_2 they are $[14, 8, 16, 12, 16, 12, 8]$, and on machine M_3 they are $[12, 7, 13, 8, 12, 10, 10]$. The given budget B is 60. In the "easy" instance having 7 jobs on 3 machine the release dates, the budget, the due dates and the costs for processing orders on machines remain the same, while the ex-

pected processing time of orders on machine M_1 are now $[5, 3, 2, 3, 2, 1, 1]$, on machine M_2 they are $[7, 4, 4, 6, 4, 3, 2]$, and on machine M_3 they are $[6, 3, 3, 4, 3, 2, 1]$. In the "hard' instance and in the "easy "instances having 5 jobs on 2 machine we simply employ data from the former instances for the first 5 jobs and for the first 2 machines. The given budget is now reduced to 40. For each instance we generated 100 Latin hypercube samples according to the job length distributions (normal distribution with $c_v = 0.15$) and to the expected job lengths provided. We assume that these samples constitute our universe of possible events. Since this universe has a finite number of scenarios we can obviously solve an instance over these 100 scenarios and obtain an optimal solution for the problem. In order to solve the proposed scheduling problem, for efficiency reasons instead of using the model produced by the EDP-CP compilation presented in [12] as it is, we reformulated the first two nonlinear constraints in a straightforward manner into linear ones. Since in the resulting model all the constraints are linear, we solved it using CPLEX 9.0 [4] using OPL Studio 3.7 [5] on an Intel(R) Centrino(TM) CPU 1.50GHz with 2Gb RAM. We maintained the default search settings. The hard 7×3 instance could not be solved to optimality in a 12 hours run and the best solution found in this time span had a value of 5.88. The hard 5×2 instance was solved in 11247.79 seconds and the optimal solution found had a value of 3.64. The easy 7×3 instance was solved in 1.16 seconds and the optimal solution found had a value of 7. The easy 5×2 instance was solved in 0.76 seconds and the optimal solution found had a value of 5. We will now assess the quality of our approach against these instances. As search parameters we set $\alpha = 0.95$, $\tau = 0.1$ (this means we want to be 95% confident that the solution found with the last sample size before the search stops can not be improved by a factor higher than 0.1 by taking a larger sample set), $s = 1$. The initial sample size is 2. In Fig. 3 we show two graphs. In the graphs on the left for each instance considered, on the X-axis we show the number of replications performed and on the Y-axis we report the % over the optimal solution value (computed by the complete MIP model) achieved. In the graph on the right we consider instead the time on the X-axis and again the % achieved over the optimal solution value on the Y-axis. In all the cases the iterative approach solved replications of instances comprising at most 4 samples, in fact the chance of finding better solutions with higher sample sizes was dynamically ruled out by our criteria. The last data point for each instance represents respectively the last run performed or the end time of the search. In Fig. 4 we consider the "hard" 7×3 instance and we solved it 10 times using our approach. Again the last dot for each instance represents the end time of the search. Notice that our approach is robust in the sense that both the quality of the solution and the required time to converge do not vary much from one run to another. We also performed a larger set of 100 runs for this instance and, according to the α and τ parameters that have been chosen, respectively 0.95 and 0.1, in 98% of the runs our method returned a solution with a value greater than $5.88 - \tau = 5.78$. Finally we will shortly present an interesting practical application for our solution method. We refer to the "easy" 7×3 instance presented in the former section. This instance, for the determin-

istic case, has been solved in [6]. In the deterministic case the problem consists in finding a feasible schedule that minimizes the processing cost. The optimal solution found has a processing cost of 44. We now set the budget B in our model to 44 and we solve the problem when the job lengths are assumed to be normally distributed with a coefficient of variation $c_v = 0.15$. The solution found with our approach has a reliability of 5.81. This means that, when uncertainty comes into play, if we want to preserve the optimal cost that we pay when jobs lengths are deterministic, then we can only schedule at most 5.81 jobs over 7.

5 Conclusions

We presented a novel stopping criterion for the SAA method based on confidence interval analysis. Although confidence interval analysis is a tool often used in conjunction with Monte Carlo simulation, to the best of our knowledge, this is the first time that this method is used to determine the maximum sample size that has to be considered in order to be confident, with probability α, that searching over a larger sample will not provide a better solution. Our preliminary results suggest that this can be really effective. In the future we aim to extend our tests over a wider class of problems.

References

1. H. Fargier, R. Martin-Clouaire J. Lang, and T. Schiex. A constraint satisfaction framework for decision under uncertainty. In *Proc. of the 11th Int. Conf. on Uncertainty in Artificial Intelligence, Montreal, Canada*, 1995.
2. M. R. Garey and D. S. Johnson. *Computer and Intractability. A guide to the theory of NP-Completeness.* Bell Laboratories, Murray Hill, New Jersey, 1979.
3. J. N. Hooker, G. Ottosson, E. S. Thorsteinsson, and H. J. Kim. On integrating constraint propagation and linear programming for combinatorial optimization. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 136–141. AAAI, The AAAI Press/MIT Press, Cambridge, Ma., 1999.
4. Inc. ILOG. *CPLEX 9.0 Users Manual.* ILOG, Inc., Incline Village, NV, 2007.
5. Inc. ILOG. *OPL Studio 3.7 Users Manual.* ILOG, Inc., Incline Village, NV, 2007.
6. V. Jain and I. E. Grossmann. Algorithms for hybrid milp/cp models for a class of optimization problems. *INFORMS Journal on computing*, 13:258–276, 2001.
7. A. J. Kleywegt, A. Shapiro, and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization*, 12(2):479–502, 2001.
8. M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.
9. J. Neyman. Fiducial argument and the theory of confidence intervals. *Biometrika*, 32:128–150, 1941.
10. E. J. Pedezma and G. B. Heuvelink. Latin hypercube sampling of gaussian random fields. *Technometrics*, 41:303–312, 1999.
11. M. L. Stein. Large sample properties of simulation using latin hypercube sampling. *Technometrics*, 29:143–151, 1987.
12. S. A. Tarim, B. Hnich, S. Prestwich, and R. Rossi. Finding reliable solutions: Event-driven probabilistic constraint programming. *Annals of Operations Research, Special Issue for CPAIOR 2006*, accepted for publication.
13. E. S. Ventsel. *Theory of Probability.* Moscow, Nauka, (In Russian), 1979.