Monte-Carlo Sampling for NP-Hard Maximization Problems in the Framework of Weighted Parsing

Jean-Cédric Chappelier and Martin Rajman

DI-LIA – EPFL – CH-1015 Lausanne – Switzerland
{chaps,rajman}@lia.di.epfl.ch

Abstract. The purpose of this paper is (1) to provide a theoretical justification for the use of Monte-Carlo sampling for approximate resolution of NP-hard maximization problems in the framework of weighted parsing, and (2) to show how such sampling techniques can be efficiently implemented with an explicit control of the error probability. We provide an algorithm to compute the local sampling probability distribution that guarantee that the global sampling probability indeed corresponds to the aimed theoretical score. The proposed sampling strategy significantly differs from existing methods, showing by the same way the bias induced by these methods.

1 Motivations

In the framework of Speech Recognition and Natural Language Processing, it is a very common task to search for elements (e.g. sentences, parse trees) a score that depends on the process that was used to produce them. Examples of such tasks include searching a word graph for the most-probable sentence (MPS) according to a Stochastic Context-Free Grammar (SCFG) [11] or finding, for a given sentence, the most probable parse tree (MPP) according to a Stochastic Lexicalized Tree Adjoining Grammar (SLTAG) [16] or a Data-Oriented Parsing (DOP) model [5].

The problem with such maximisation tasks is that they often correspond to an instance of an NP-hard problem [17] and, as such, cannot be solved exactly in an effective way. In such cases, heuristics and/or approximations need to be used instead. Monte-Carlo sampling is an example of an approach that approximates the exact maximization by a search over a reduced random sample that can be generated at reasonable (i.e. polynomial) algorithmic cost [4].

The purpose of the present paper is to provide theoretical justification for the use of Monte-Carlo sampling for such NP-hard maximization problems in the framework of parsing, and to show how such sampling techniques can be efficiently implemented and controlled; i.e. guarantee, with some *a priori* fixed probability error, that the approximate solution indeed corresponds to an exact solution of the problem.

D.N. Christodoulakis (Ed.): NLP 2000, LNCS 1835, pp. 106–117, 2000.

The paper first provides a short overview of some NP-hard maximization problems in the general framework of "*weighted parsing*" (as defined in section 2.1). It then presents a generic implementation for an approximate solution based on Monte-Carlo sampling. The problem of the control of the probability error during sampling is finally discussed.

2 General Framework

2.1 Weighted Parsing

Weighted parsing can be defined as parsing with a phrase-structure grammar, the productions of which (i.e. rules) are scored, i.e. mapped on a subset of the positive real numbers $[0, +\infty)$. Such scores often consist of probabilities, but this is by no means mandatory for the approach described here.

The mapping (denoted by σ) is extended by definition to any derivation as the product of the scores associated with the productions in the derivation.[1] More precisely, the score $\sigma(d)$ associated with a derivation $d = r_1 \circ ... \circ r_q$ [2] is $\sigma(d) = \prod_{r \in d} \sigma(r) = \prod_{i=1}^{q} \sigma(r_i)$.

Moreover, if some equivalence classes are defined over the derivations (for instance the equivalence classes induced by sentences: two derivations are in the same class if they produce the same sentence), the scores can be extended to derivation classes ($\overline{d}$) as the sum of the scores of all the derivations that belong to a given class, finally leading a sum of products: $\sigma(\overline{d}) = \sum_{d \in \overline{d}} \sigma(d) = \sum_{d \in \overline{d}} \prod_{r \in d} \sigma(r)$.

2.2 Some NP-Hard Maximization Problems in the Framework of Weighted Parsing

When only one single class of derivation is considered (for instance the parsing of a unique input sentence with a SCFG), weighted parsing may be achieve in polynomial time using standard parsing techniques [13,18,9].

But parsing may also be used in situations where *several different* classes of derivations are mixed together (i.e. share some components). Examples of such situations include finding the MPS in a word-graph using a SCFG as a language model (speech recognition), or looking for the most-probable parse of a sentence according to a Stochastic Tree Substitution Grammar (STSG, as in DOP for instance [7]), where several derivations may lead to the same parse tree[3].

[1] This is in fact a morphism between the (partial) semi-ring of the derivations in the grammar and the natural semi-ring $\mathbb{R}^+$. Readers interested by this formal aspect are referred to [15].

[2] where $\circ$ denotes the sequential composition of production rules. $r \in d$ will denote the fact that production r occurs in derivation d.

[3] in that case the derivations classes are defined as the set of derivations that lead to the same tree

Let us summarize what are the derivations, the classes and the score to be maximized in two specific cases: SCFG-MPS and DOP-MPP:

	SCFG-MPS	DOP-MPP
input	a word-graph	a sentence
derivations	all the derivations of all sentences in the word-graph	all the derivations of the input sentence
classes	all the derivations of a single sentence	all the derivations leading to the same parse tree
class score	sentence probability	DOP probability of a parse tree

For these two cases, finding the class that maximizes the score σ is an NP-hard problem [17]. This means that finding the (exact) optimal class cannot be achieved in polynomial time (in general).

Parsing usually consists in two phases:

analysis i.e. building a compact representation of all the possible derivations of the input;

extraction of results from the former representation: e.g. displaying all the parse trees or extracting the best parse tree.

The analysis can always be achieved in a time cubic with respect to the input size. However, the extraction, i.e. the "unfolding" of the compact representation produced by analysis, may lead to the NP-hard problem mentioned above.

2.3 Parsing Analysis and Notations

In our case, the analysis phase of the parsing consists in a bottom-up filling of a parse chart with items that represent all the possible subderivations of all possible substrings of the input, a sentence in the DOP-MPP case [9] and a word-graph in the SCFG-MPS case [10].

More precisely, the parse chart consists of a set of *items* $[X, i, j]$ each representing the fact that the substring $w_i...w_j$ of the input can be derived from the non-terminal X (this will be noted $X \Rightarrow^* w_i...w_j$). In the case of DOP-MPP, where the input consists of a single sentence, $w_i...w_j$ is simply the substring from the i-th to the j-th word of that sentence. In the case of SCFG-MPS, where the input is a time-indexed word-graph, $w_i...w_j$ represents the subsequence of words going from time-step i to time-step j. In the latter case, notice that the actual number of words in $w_i...w_j$ may be any value between 1 and $j - i + 1$.

For an item $[X, i, j]$, a *decomposition* of that item is defined as a set of items that explicit the first step in the derivation $X \Rightarrow^* w_i...w_j$. For instance in the case of a binary grammar[4], a decomposition of $[X, i, j]$ is a couple consisting of:

1. a production r of the form $X \rightarrow Y\ Z$ [5] or $X \rightarrow w$ and that is the first step of some $X \Rightarrow^* w_i...w_j$,

[4] i.e. Chomsky Normal Form for SCFG and binary trees grammar for DOP

[5] In the case of DOP, the production $r = X \rightarrow Y\ Z$ is indeed an elementary tree, the root of which is X, the left-most non terminal leave is Y and the other is Z. Notice that the points discussed here are independent of the internal structure of the

2. and a position k where to "split" the production r, i.e. defining the substrings $w_i...w_k$ and $w_{k+1}...w_j$ such that $Y \Rightarrow^* w_i...w_k$ and $Z \Rightarrow^* w_{k+1}...w_j$. By convention $k = 0$ if r is a terminal production $X \rightarrow w$.
   In the case of a non binary grammar, the above definition can be generalized by replacing k by a tuple of indices (see [8] for details.)

A decomposition ξ of $[X, i, j]$ can either be denoted by $\langle r, k\rangle$ defined above or, in a equivalent manner, by the two[6] corresponding items $\xi = \ll [Y, i, k], [Z, k+1, j] \gg$ (or $\xi = \ll w_i...w_j \gg$ when the production r is terminal: $X \rightarrow w_i...w_j$).

In order to simplify the notation, the score σ will be extended to decompositions of items by $\sigma(\langle r, k\rangle) = \sigma(r)$.

Finally, for a decomposition ξ, let $\mathcal{D}(\xi)$ denote the set of all possible decompositions of the item of which ξ is itself a decomposition. This notation is generalized to items: $\mathcal{D}([X, i, j])$ is the set of all possible decompositions of the item $[X, i, j]$.

Example: As illustration, let us consider the very simple case of parsing the input sequence "`a a a`" with the following CFG: `S` $\rightarrow$ `S S | a` where `a` is the only terminal and `S` the only non-terminal. In that case, 6 items will be present in the chart and have the following decompositions:

item	number of decompositions	$\langle r, k\rangle$ representation of decompositions	$\ll [Y, i, k], [Z, k+1, j] \gg$ representation
$[S, 1, 1]$	1	$\langle$`S` $\rightarrow$ `a`$, 0\rangle$	$\ll$`a`$\gg$
$[S, 2, 2]$	1	$\langle$`S` $\rightarrow$ `a`$, 0\rangle$	$\ll$`a`$\gg$
$[S, 3, 3]$	1	$\langle$`S` $\rightarrow$ `a`$, 0\rangle$	$\ll$`a`$\gg$
$[S, 1, 2]$	1	$\langle$`S` $\rightarrow$ `S S`$, 1\rangle$	$\ll [S, 1, 1], [S, 2, 2] \gg$
$[S, 2, 3]$	1	$\langle$`S` $\rightarrow$ `S S`$, 2\rangle$	$\ll [S, 2, 2], [S, 3, 3] \gg$
$[S, 1, 3]$	2	$\langle$`S` $\rightarrow$ `S S`$, 1\rangle$, $\langle$`S` $\rightarrow$ `S S`$, 2\rangle$	$\ll [S, 1, 2], [S, 3, 3] \gg$, $\ll [S, 1, 1], [S, 2, 3] \gg$

If ξ is the decomposition $\ll [S, 1, 2], [S, 3, 3] \gg$ of item $[S, 1, 3]$, then $\mathcal{D}(\xi) = \mathcal{D}([S, 1, 3]) = \{\ll [S, 1, 2], [S, 3, 3] \gg, \ll [S, 1, 1], [S, 2, 3] \gg\}$.

2.4 Monte-Carlo Estimation

The purpose of Monte-Carlo estimation in the above described framework is to approximate the extraction of the best class by a search limited to a sample of derivations randomly extracted [4]. This approximated search is controlled by an arbitrary small probability of error *a priori* fixed ("control error"). Such an approach is possible and interesting only if:

1. the approximated score of the sampled classes may be computed from samples;

elementary trees. The internal structure of elementary tree (which are in general of depth greater than 1) is therefore kept apart in another representation not addressed here and that is only used for the final display of the whole result.

[6] g in the general case of non-binary grammars, with g the arity of the production r.

Sample $[X, i, j]$:

choose at random a decomposition $\xi = \langle r, k\rangle$ of $[X, i, j]$
if ξ is terminal (i.e $k = 0$ and $r = X \rightarrow w_i$)
  return r
else
  (here $\xi = \langle r, k\rangle = \ll [Y, i, k], [Z, k + 1, j]\gg$)
  return r, **Sample** $[Y, i, k]$, **Sample** $[Z, k + 1, j]$

Table 1. A simple version of the sampling algorithm in the case of a binary grammar. This algorithm is applied top-down from $[S, 1, m]$ to words on a filled parse chart.

2. the best class (according to the real score) can be found on the basis of the approximated scores;
3. the sampling of classes may be achieved efficiently (polynomial time).

For the sake of simplicity, let us first explain on a simple binary grammar what the sampling method consist of, although it may easily be generalized to any chart parser and any SCFG/STSG [8]. As detailed in table 1, the sampling algorithm (that takes place once the analysis step described in the previous section has been performed) consists in recursively choosing at random, from top $[S, 1, m]$ to bottom (words), a possible decomposition of the current item.

As choosing a decomposition of an item may be $\mathcal{O}(m)$ if not carefully implemented (see also [12] page 177), this sampling technique might be of cost $\mathcal{O}(m^2)$ [7] but can easily be improved to $\mathcal{O}(m)$ if the storage of an item in the parse chart directly points to its decomposition [8].

The main problem of Monte-Carlo estimation however is not the sampling function itself but how to correctly implement the action "*choose at random a decomposition*", so that the three above mentioned conditions are fulfilled (and especially so as to ensure that the sampling probability converges to the appropriate score).

The two following sections detail the different sampling strategies that are possible for general item-based chart parsers:

rescored sampling: the sampling probability for choosing the items in the parse chart is a simple, *a priori* defined, value. The probability of the sampled classes then needs to be rescored *a posteriori* so as to get the proper (theoretical) value for its score;

exact sampling: the sampling probability of items ensures that the overall sampling probability of a class is the normalized score of that class, i.e. the (theoretical) score of the class divided by the sum of the scores of all classes.

The advantages of the first approach consist in its simplicity and its generality while the advantages of the second one lies in a faster convergence as well as the

[7] where m is the input size

possibility of an *a priori* control over the error probability (and hence over the accuracy of the method).

3 Rescored Sampling

3.1 General Method

Let $d_i, i = 1...N$ be all the possible derivations that can be sampled (i.e. all the derivations of all the classes for the considered problem) and n_i the corresponding number of occurrences in a given sample of size n.[8] Notice that some n_i may be null. Let us define the estimated rescoring $W^{(n)}(\overline{d})$ of a class $\overline{d}$ by: $W^{(n)}(\overline{d}) = \sum_{d_i \in \overline{d}} \frac{n_i}{n} W_i$, where W_i is some rescoring factor for the derivation d_i. The interest of such a definition lies in the fact that, by the law of large numbers (i.e. when n grows to infinity), the rescored estimation $W^{(n)}(\overline{d})$ converges to $W^{(\infty)}(\overline{d}) = \sum_{d_i \in \overline{d}} P_{si} W_i$, where P_{si} the sampling probability of derivation d_i.

This convergence property makes the link between local random choices (P_{si}) and the global score obtained for the sampled classes. Provided that it is possible to compute both W_i and P_{si} for each sampled derivation d_i, this allows to estimate $W^{(\infty)}(\overline{d})$ by the sampled value $W^{(n)}(\overline{d})$. This is of course particularly useful when $W^{(\infty)}(\overline{d}) = \sigma(\overline{d})$, the original score to be maximized.

A first way of sampling, which will be called "*naive* sampling", consists of randomly choosing among the decompositions of an item with a uniform distribution. This corresponds to $P_{si} = \prod_{\xi \in d_i} \frac{1}{|\mathcal{D}(\xi)|}$.[9] Without rescoring (i.e. choosing $W_i = 1$), naive sampling leads, for every class $\overline{d}$, to a score $W^{(\infty)}(\overline{d})$ of the form $\sum_{d_i \in \overline{d}} \prod_{\xi \in d_i} \frac{1}{|\mathcal{D}(\xi)|}$, which is not the score $\sigma(\overline{d})$ to be estimate. Naive sampling can however be corrected, introducing rescoring factors $W_i = \frac{\sigma(d_i)}{\prod_{\xi \in d_i} \frac{1}{|\mathcal{D}(\xi)|}}$, which lead to $W^{(\infty)}(\overline{d}) = \sigma(\overline{d})$. It is important to notice that these rescoring factors are easily computable during sampling. Indeed, as $W_i = \frac{\sigma(d_i)}{\prod_{\xi \in d_i} \frac{1}{|\mathcal{D}(\xi)|}} = \frac{\prod_{\xi \in d_i} \sigma(\xi)}{\prod_{\xi \in d_i} \frac{1}{|\mathcal{D}(\xi)|}} = \prod_{\xi \in d_i} \frac{\sigma(\xi)}{|\mathcal{D}(\xi)|}$, it can be computed during the top-down extraction of each sampled derivation by iterative multiplications of the $\frac{\sigma(\xi)}{|\mathcal{D}(\xi)|}$ scores. Naive sampling may therefore be implemented anyway, therefore providing a very easy estimation method for the class that maximizes $\sigma(\overline{d})$.

[8] $\sum_{i=1}^{N} n_i = n$

[9] with the notation $\xi \in d_i$ as a subscript of a sum or product meaning "*for all decompositions of items chosen during the sampling of derivation* d_i".

However a better way for finding the best class by sampling is to choose $W_i = 1$ for all derivations and $P_{si} = \sigma(d_i)$,[10] leading to $W(\overline{d}) = \sum_{d \in \overline{d}} P_{si} = \sigma(\overline{d})$. Whether $P_{si} = \sigma(d_i)$ can actually be implemented is studied in section 4.

3.2 R. Bod Sampling Method for DOP

We now have all the theoretical tools to study the sampling method used by R. Bod for DOP [4,5,7,6]. His sampling technique is exactly the one described above but without the correct rescoring. Bod chooses as local random choice a probability such that "*a subderivation*[11] *that has n times as large a* [DOP] *probability as another subderivation should also have n times as large a chance to be chosen as this other subderivation*" [5] i.e. the sampling probability of a decomposition ξ of $[X, i, j]$ is $P_s(\xi) = \dfrac{\sigma(\xi)}{\sum_{\xi' \in \mathcal{D}(\xi)} \sigma(\xi')} = \dfrac{P_{\text{DOP}}(\xi)}{\sum_{\xi'} P_{\text{DOP}}(\xi')}$,[12] leading therefore to a P_{si} for a derivation d_i of the form $P_{si} = \prod_{\xi \in d_i} \dfrac{P_{\text{DOP}}(\xi)}{\sum_{\xi' \in \mathcal{D}(\xi)} P_{\text{DOP}}(\xi')} = P_{\text{DOP}}(d_i) \cdot \dfrac{1}{\prod_{\xi \in d_i} \sum_{\xi' \in \mathcal{D}(\xi)} P_{\text{DOP}}(\xi')}$. The resulting score for classes[13] is therefore $\sum_{d \in \overline{d}} \dfrac{P_{\text{DOP}}(d)}{\prod_{\xi \in d} \sum_{\xi' \in \mathcal{D}(\xi)} P_{\text{DOP}}(\xi')}$ which is <u>not</u> the DOP-probability of that tree in the general case. This shows that Bod's sampling does not at all lead to the right (i.e. DOP) probability, therefore imposing on parse trees a score that as nothing to do with the DOP theory. To have his sampling correct, R. Bod should have rescored the sampled derivations by $W_i = \prod_{\xi \in d_i} \sum_{\xi' \in \mathcal{D}(\xi)} P_{\text{DOP}}(\xi')$, which is indeed computable during sampling (and is not 1 in the general case). Notice also that $P_s(\xi)$ has to be a probability over $\mathcal{D}(\xi)$, i.e. that $\sum_{\mathcal{D}(\xi)} P_s(\xi') = 1$. Therefore $P_s(\xi) = P_{\text{DOP}}(\xi)$ is not a possible choice.

Table 2 resume the three sampling rescoring seen so far.

4 Exact (Controlled) Sampling

The purpose of exact sampling techniques is to sample decompositions so as to get a sampling probability for each class that is equal to the (theoretical) score of the class divided by the sum of the scores of all possible classes without any rescoring. Such sampling techniques ensure that the best class has the best sampling probability. Then, due to the law of large numbers (the sampling frequency

[10] more precisely: a renormalized version of σ in case where σ in not a probability.
[11] i.e. decomposition
[12] $P_{\text{DOP}}(\xi)$ is the product of the probabilities of elementary trees constituting this subderivation.
[13] i.e. parse trees in this case.

method	$P_s(\xi)$	rescoring factor W_i
R. Bod for DOP	$\frac{\sigma(\xi)}{\sum_{\xi' \in \mathcal{D}(\xi)} \sigma(\xi')}$	used: 1 correct: $\prod_{\xi \in d_i} \sum_{\xi' \in \mathcal{D}(\xi)} \sigma(\xi')$
naive	$\frac{1}{\lvert\mathcal{D}(\xi)\rvert}$	$\frac{\sigma(d_i)}{\prod_{\xi \in d_i} \frac{1}{\lvert\mathcal{D}(\xi)\rvert}}$

Table 2. Rescoring factors for non exact sampling methods.

converging towards the sampling probability) the most frequent class in a sample is the best class. One important advantage of exact sampling is the fact that it allows a control of the number of samples to be drawn.

4.1 Computation of the Decomposition Sampling Probability

We here derive the exact form for the sampling probability of the decomposition elements so that the final sampling probability of a class is directly the correct score (divided by the sum of scores) without any rescoring.

The explanation is once again given in the simpler case of binary grammars but still generalizes, using significantly much more notations, to any SCFG/STSG grammar as detailed in [8].

For each decomposition, an intermediate score σ_0 is introduced. For $\xi = \langle r, k \neq 0 \rangle = \ll [Y, i, k], [Z, k+1, j] \gg$, σ_0 is defined by

$$\sigma(r) \cdot \sum_{\xi_Y \in \mathcal{D}([Y,i,k])} \sigma_0(\xi_Y) \cdot \sum_{\xi_Z \in \mathcal{D}([Z,k+1,j])} \sigma_0(\xi_Z) \ ,$$

and by $\sigma(r)$ for terminal decomposition $\xi = \langle r, 0 \rangle$.

Then for any decomposition ξ, the sampling probability is set to $P_s(\xi) = \frac{\sigma_0(\xi)}{\sum_{\xi' \in \mathcal{D}(\xi)} \sigma_0(\xi')}$ which actually is a probability over all the possible choices at a given point of the derivation extraction (i.e. on $\mathcal{D}(\xi)$). In the case of DOP, notice how this differs from Bod's sampling explained in the former section: σ_0 is indeed the inside probability of a subderivation whereas, in Bod's case, it is replaced by P_{DOP}, the probability of the subderivation *in the corresponding derivation currently being extracted* (which, in the most general case, is different from its inside probability). Rephrasing what R Bod said, the correct sentence would have been "*a subderivation that has n times as large an* inside *probability as another subderivation should also have n times as large a chance to be chosen as this other subderivation*".

Coming back to the general case, what is the sampling probability of a given derivation d? Each decomposition being chosen independently (cf the sampling algorithm given in table 1), the sampling probability of a derivation is the product of the sampling probabilities of the chosen decomposition: $P_s(d) = \prod_{\xi \in d} P_s(\xi)$.

It can be shown by induction [8] that $P_s(d)$ is equal to $\frac{1}{\sum_{\overline{d}} \sigma(\overline{d})} \prod_{r_j \in d} \sigma(r_j)$ which actually is $\frac{\sigma(d)}{\sum_{\overline{d}} \sigma(\overline{d})}$.

The sampling probability of a class $\overline{d}'$, being the sum of sampling probabilities of its derivations, is then $P_s(\overline{d}') = \sum_{d \in \overline{d}'} \frac{\sigma(d)}{\sum_{\overline{d}} \sigma(\overline{d})} = \frac{1}{\sum_{\overline{d}} \sigma(\overline{d})} \sigma(\overline{d}')$, i.e. the score of the class over the sum of all scores of all classes.

This sampling method therefore behaves as a multinomial random variable with K modalities (K being the number of possible classes) whose parameters are $\frac{\sigma(\overline{d})}{\sum_{\overline{d}'} \sigma(\overline{d}')}$. This is precisely the reason why the method can furthermore be controlled.

4.2 Control for Exact Sampling Method

The whole story consist in *controlling* the convergence of the sampling, i.e. determining the number of samples so as to **ensure** that the most frequent class in the sample actually is the best one.[14] This is a classical problem of statistical ordering for this problem is exactly the same as finding the most probable modality in a K-modal binomial law.

The control method proposed by R. Bod for his sampling in the DOP-MPP framework is an illustration of such a mechanism. However (and regardless of the fact that the sampling probability he used does not converge to the right score) the estimation of sampling probabilities themselves is wrong, estimating the error $\sum_{i>1}(1 - (\sqrt{p_{[1]}} - \sqrt{p_{[i]}})^2)^n$ by $\sum_{i>1}(1 - (\sqrt{f_{[1]}} - \sqrt{f_{[i]}})^2)^n$.[15] It is very difficult to evaluate the impact of such an error on the results obtained. Moreover the purely sequential aspect of his methods does not permit to *a priori* compute the size of the sample needed for right estimation.

For all the reasons, it is important to use more sophisticated methods as, for instance, the ones existing in the statistic literature. We will explain only one of these, the most powerful: the Bechhofer-Kiefel-Sobel truncated methods (BKST) [2] which is a sequential truncated sampling method that combines two other methods: BEM and BKS sampling.

BEM Sampling. It is known that for any multinomial random variable with K modalities (as it is the case for classes of derivations) such as $p_{[1]} \geq \theta\, p_{[2]}$

[14] even if P_s corresponds to the correct score, this does not ensure that for a given sample the most frequent class in that sample is actually the best one since for a given sample the most probable (with respect to P_s) class in not necessary the most frequent in that sample.

[15] where n is the number of samples, $f_{[i]}$ is the frequency in the sample of the i-th most frequent class (in that sample) and $p_{[i]}$ its theoretical probability to be sampled.

with $\theta > 1$,[16] the probability that the most frequent modality in a sample if effectively the most probable one is always bigger than the probability $P_{\min}$ of selecting the best one in the case where all but this best one are equal [14]. This lower bound $P_{\min}$ can more be *a priori* computed as a function of K,[17] θ and n [1].

The non-sequential BEM controlled sampling method is then as simple as:

1. choose *a priori* some value for $\theta = \frac{p_{[1]}}{p_{[2]}}$ and a control error probability P,
2. compute (from tables) the smallest sample size n so that $P \leq P_{\min}(K, \theta, n)$,
3. determine the best class as the most frequent one on a sample of n classes.

BKS Sampling. BKS is a sequential sampling method that relies on the following result: for any multinomial random variable with K modalities such as $p_{[1]} \geq \theta\, p_{[2]}$ with $\theta > 1$, the probability that the most frequent modality in a sample if effectively the most probable one is always bigger than $\frac{1}{1+Z}$ with $Z = \sum_{i=2}^{K} (\frac{1}{\theta})^{(n_{[1]} - n_{[i]})}$ [3].[18] The BKS method is then:

1. choose *a priori* some value for $\theta = \frac{p_{[1]}}{p_{[2]}}$ and a control error probability P,
2. keep on sampling, updating the $n_{[i]}$'s[19] and Z, as long as $\frac{1}{1+Z} < P$,
3. determine the best class as the most frequent one.

BKST. BKST is a sequential truncated sampling method that combine BEM and BKS: BEM is included in BKS, adding to the stopping condition of BKS the BEM criterion on the maximal (precomputed) sample size. BKST is therefore guaranteed to stop at the minimal stopping time of BEM and BKS.

If we really have $p_{[1]} \geq \theta\, p_{[2]}$ then the class chosen with any of these methods is really the best one (with a error probability P).

4.3 Convergence of Sampling Methods

In addition to the fact that exact sampling allows perfect control, another advantage of this method is its much faster convergence to the correct distribution of classes. This can be illustrated by the simple simulation results given in figure 4.3 where 20 runs of 5000 sample of a single random variable have been produced using either the rescored sampling method or the exact one. It clearly appears on that figure that the variance (resp. the convergence rate) of the first one is much bigger (resp. smaller) than for the exact (not rescored) one.

[16] $p_{[1]}, ..., p_{[K]}$ being the K parameters of the multinomial in decreasing order: $p_{[1]} \geq ... \geq p_{[K]}$ (with $p_{[1]} + ... + p_{[K]} = 1$).

[17] If K is not known *a priori*, it can be replaced without restriction by some upper bound.

[18] $n_{[i]}$ being the i-th number of occurrence of a class in the sample, in decreasing order.

[19] not only the values but also the order.

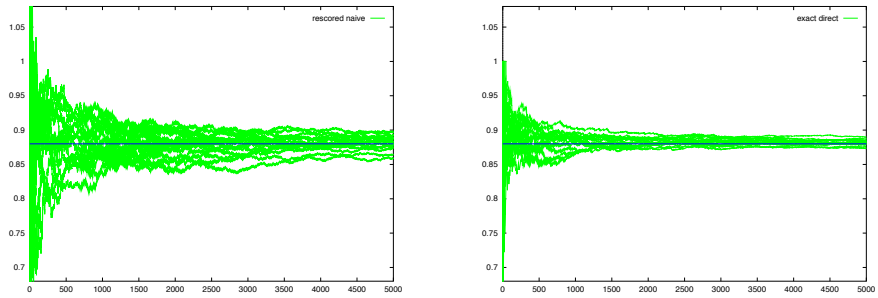

Fig. 1. (score .vs. sample size) Comparison of convergence of the rescored sampling method (left) and the exact method (right) on 20 runs of 5000 samples converging to the known score of 0.88: exact method converges much faster.

This fact is easy to understand intuitively: rescoring is needed when the sampling probability does not lead to the correct score, which means that the most frequent samples are not the correct ones. Therfore, in the case where rescoring is needed, the good exemples are less frequent than they should be (if the sampling probability would have been correct). Therefore more samples are needed to have a good approximation of the (correct) best class.

5 Conclusion

This paper presents three important results for the approximation of solutions of NP-hard maximization problems in the framework of weighted parsing:

1. We have shown that Monte-Carlo sampling techniques can actually be implemented in such cases. We furthermore derive the relationship between (local) sampling probability and the score of a class;
2. We have computed what the sampling probability of a decomposition has to be so that the sampling probability of a class exactly is the score of that class (among the sum of the scores). This sampling strategy significantly differs from the former existing ones, showing by the same way the bias induced by such methods.
3. Finally we have presented a method that allow to control the sample quality so as to be sure (with some *a priori* known control error) that the most frequent class in the sample is the best one.

These results allow experiments on and practical use of, for instance, SCFG-MPS word-graph extraction (useful in speech recognition) or STSG-MPP extraction (useful for the DOP model), grounded on a more robust and theoretically grounded basis.

References

1. R.E. Bechhofer, S. Elmaghraby, and N. Morse. A single-sample multiple-decision procedure for selecting the multinomial event which has the largest probability. *Ann. Math. Statist.*, 30:102–119, 1959.
2. R.E. Bechhofer and D.M. Goldsman. Truncation of the Bechhofer-Kiefer-Sobel sequential procedure for selecting the multinomial event which has the largest probability. *Communications in Statistics: simulation and computation*, 14(2):283–315, 1985.
3. R.E. Bechhofer, J. Kiefer, and M. Sobel. *Sequential Identification and Ranking Procedures*. University of Chicago Press, Chicago, 1968.
4. R. Bod. Applying Monte Carlo techniques to Data Oriented Parsing. In *Proceedings Computational Linguistics in the Netherlands*, Tilburg (The Netherlands), 1992.
5. R. Bod. *Enriching Linguistics with Statistics: Performance Models of Natural Language*. Academische Pers, Amsterdam (The Netherlands), 1995.
6. R. Bod. *Beyond Grammar, An Experience-Based Theory of Language*. Number 88 in CSLI Lecture Notes. CSLI Publications, Standford (CA), 1998.
7. R. Bod and R. Scha. Data-Oriented language processing: An overview. Technical Report LP-96-13, Departement of Computational Linguistics, University of Amsterdam, 1996. cmp-lg/9611003.
8. J.-C. Chappelier and M. Rajman. Extraction stochastique d'arbres d'analyse pour le modèle DOP. In *Proc. of 5ème conférence sur le Traitement Automatique du Langage Naturel (TALN98)*, pages 52–61, Paris (France), June 1998.
9. J.-C. Chappelier and M. Rajman. A generalized CYK algorithm for parsing stochastic CFG. In *TAPD'98 Workshop*, pages 133–137, Paris (France), 1998.
10. J.-C. Chappelier, M. Rajman, R. Aragues, and A. Rozenknop. Lattice parsing for speech recognition. In *Proc. of 6ème conférence sur le Traitement Automatique du Langage Naturel (TALN'99)*, pages 95–104, July 1999.
11. A. Corazza, R. Demori, R. Gretter, and G. Satta. Optimal probabilistic evaluation functions for search controlled by stochastic context-free grammars. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(10):1018–1027, October 1994.
12. J. Goodman. *Parsing Inside-Out*. PhD thesis, Harvard University, May 1998. `cmp-lg/9805007`.
13. F. Jelinek, J. D. Lafferty, and R. L. Mercer. Basic methods of probabilistic context-free grammars. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding: Recent Advances, Trends and Applications*, volume 75 of *F: Computer and System Science*. Springer, 1992.
14. H. Kesten and N. Morse. A property of the multinomial distribution. *Ann. Math. Statist.*, 30:120–127, 1959.
15. W. Kuich. Semirings and formal power series: Their relevance to formal languages and automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages*, volume 1, chapter 9, pages 609–677. Springer-Verlag, 1997.
16. Yves Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proc. 14th Int. Conf. of Computationnal Linguistics (COLING)*, pages 426–432, Nantes (France), August 1992.
17. K. Sima'an. Computational complexity of probabilistic disambiguation by means of tree grammars. In *Proceedings of COLING'96*, Copenhagen (Denmark), 1996. cmp-lg/9606019.
18. A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201, 1995.