Refining motif finders with E-value calculations

Niranjan Nagarajan, Patrick Ng, Uri Keich
Department of Computer Science,
Cornell University, Ithaca, NY, USA

Abstract

Motif finders are an important tool for searching for regulatory elements in DNA. Popular existing programs optimize the entropy score to efficiently search for motifs. While E-values are commonly used for assigning significance to the optimal reported motifs they are not directly optimized for. This raises the question whether optimizing for E-values instead of entropy could improve the finders' ability to detect weak motifs. We first present an efficient algorithm to accurately compute multiple E-values which changes the nature of the above question from a hypothetical to a practical one. Incorporating this method into CONSENSUS- and Gibbs-based finders we then demonstrate on synthetic data that the answer to our question is positive. In particular, E-value based optimizations show significant improvement over existing tools for finding motifs of unknown width.

1 Introduction

The problem of motif-finding can be summarized as scanning a given set of sequences for short, well-conserved ungapped alignments. Most of the interest in this problem comes from its application to identification of transcription factor binding sites, and of *cis*-regulatory elements in general. These in turn are important to the fundamental problem of understanding the regulation of gene expression. This motivated the design of several popular motif-finding tools that search for short sequence motifs given only an input set of sequences (see [11] for a recent comparative review).

Most existing motif finders can be divided into two classes depending on whether they model a motif with a consensus sequence or with a position weight matrix (PWM or profile). Commonly used motif finders that fall in this latter category include MEME [1], CONSENSUS [4] and the various approaches to Gibbs sampling (e.g. [7], [9], [5]). This paper concentrates on improving this class of finders.

Profile-based motif finding algorithms typically try to optimize the entropy score, or information content of the reported alignment which is defined as [10]:[1]

$$I := \sum_{i=1}^{w} \sum_{j=1}^{A} n_{ij} \log \frac{n_{ij}/N}{b_j},$$

where w is the motif width, n_{ij} denotes the number of occurrences of the jth letter in the ith column of the alignment, b_j is the background frequency of the jth letter[2], N is the number of sequences in the alignment, and A the alphabet size. In order to assign statistical significance to the reported motifs as well as to be able to compare alignments of different widths and depths

[1] Strictly speaking, relative entropy is defined as I/N.
[2] Typically estimated from the entire sample.

Hertz and Stormo introduced the notion of a motif E-value. Introduced originally in this context as the "expected frequency" [4], the E-value is the expected number of random alignments of the same dimension that would exhibit an entropy score that is at least as high as the score of the given alignment. When the E-value is high, one can have little confidence in the motif prediction, and conversely when the E-value is low, one can have more confidence in the prediction. It is computed by multiplying the number of possible alignments by the p-value of the alignment. The latter is defined as the probability that a single *given* random alignment would have an entropy score $\geq$ the observed alignment score.

While the E-value is the chosen figure-of-merit for evaluating motifs in popular motif finders such as MEME and CONSENSUS it is not directly optimized for. For example, in MEME E-values are only computed after the EM-algorithm completes its optimization and are only used for significance evaluation and possibly for comparing motifs of different widths. Similarly, when CONSENSUS looks to extend a sub-alignment (matrix) in its greedy search strategy, it chooses the one that optimizes the entropy rather than the E-value[3]. One of the main reasons for this separation between optimization and significance analysis is that E-values are significantly more expensive to compute than entropy scores. Even the relatively fast (and potentially inaccurate [8]) large-deviation method that CONSENSUS employs for computing the E-value can tax an optimization procedure at an unacceptable level.

The discussion above raises two questions:

- Cost aside, can a more direct optimization of the E-value improve our results?
- Can we compute the E-values efficiently so that they can be optimized for?

This paper lays out arguments advocating a positive answer for both questions.

We begin by describing a new technique, memo-sFFT, that allows us to accurately and efficiently compute multiple E-values. We then present the Conspv program that uses the memo-sFFT system to implement a CONSENSUS style motif finder that directly optimizes E-values. The Conspv program generalizes readily to the problem of finding motifs of unknown widths and is functionally equivalent to a combination of CONSENSUS and WCONSENSUS [4]. We show based on experiments on synthetic data that Conspv can significantly improve over WCONSENSUS for finding motifs of unknown widths. As further evidence to the advantage of a more direct optimization of the E-values, we introduce Gibbspv. This new variant of the Gibbs-sampling algorithm is especially effective when searching for motifs of unknown width by incorporating memo-sFFT to efficiently consider E-values in its optimization procedure. In our experiments on synthetic datasets, Gibbspv clear outperforms other motif finders for finding motifs of unknown width.

It should be noted that GLAM [3] is conceptually quite similar to Gibbspv as both rely on a Gibbs sampling procedure to optimize an overall measurement of statistical significance. However GLAM uses a different significance analysis and as we show below in our tests it is less successful than both Conspv and Gibbspv.

2 Efficiently computing E-values

In a typical application of CONSENSUS in the experiments described in section 6 about 10^8 alignments are compared. CONSENSUS compares them using entropy scores that can be computed in $O(wn + wA)$ time from scratch, where w is the width of the motif, n is the number of sequences and A is the alphabet size (for our purposes a DNA alphabet of 4 letters). Note that

[3]These two approaches would generally differ if the lengths of the sequences are not identical.

```
MEMO-sFFT(n, w, I)
1  if accuracy[n][w][I] < B
2     then (pvalue sFFT , accuracy sFFT ) ← sFFT(n, w, I)
3          for each I
4          do if accuracy[n][w][I] < accuracy sFFT [I]
5                then pvalue[n][w][I] ← pvalue sFFT [I]
6                     accuracy[n][w][I] ← accuracy sFFT [I]
7
8  return pvalue[n][w][I]
```

Figure 1: **The memo-sFFT algorithm.** Here I is the latticized entropy score [6], B is a desired upper-bound on the relative error (that we set to 10^{-2}) and each entry of array *accuracy* is initialized to a value $\geq B$. Note that we use the term accuracy here to refer to the rigorous bound on the relative roundoff error that can be computed for p-values computed using sFFT [6].

the typical case in CONSENSUS is actually when the score is updated while extending a sub-alignment and this takes $O(w)$ time. In comparison, computing E-values reliably can be many orders of magnitude more expensive if done naively. An efficient algorithm for reliably computing a single p-value (a crucial time-limiting step for computing E-values, see [4]) can typically take $\approx 0.01s$ for the test sets in section 6. This can be prohibitively expensive if incorporated into Conspv (see Table 1).

A partial solution to this problem is to memoize the results. However, we can do even better by relying on algorithms that can compute p-values for a range of scores ([4], [6]). While a single application of these algorithms can be more than 10 times slower, this is compensated for by the fact that they compute a range of p-values that can be stored and reused. We exploit this feature to extend the sFFT algorithm in [6][4] to the memo-sFFT algorithm shown in Figure 1.

In addition we also implemented the following optimizations to memo-sFFT for its use in Conspv and Gibbspv:

- sFFT computes an array p_δ (the pmf of a single column) as the first step in its calculations and this array is independent of the value of w. We utilize this fact and modify sFFT to save and reuse this array across runs.
- The sFFT algorithm requires a lattice size Q (or equivalently a step size δ) that acts as a knob to trade accuracy for speed. We found that setting δ to 0.02 provides good accuracy[5] while being efficient for the experiments in section 6.
- As observed in [6] the sFFT algorithm can typically be used to recover the entire range of p-values (for a given n and w) in a small number (≤ 3) of invocations. In particular, we found that a single well-chosen call to sFFT ($\theta = 1$) can provide a good starting point for memo-sFFT and we implemented this as part of our system.

As can be seen from the results in Table 1, Conspv based on memo-sFFT is indeed much more efficient than a version that computes E-values based on the large-deviation method in CONSENSUS. For the sets described in section 6, we found that less than half a minute is spent in pre-computing p-values in Conspv and the amortized cost of a call to memo-sFFT is essentially that of a table-lookup. The memo-sFFT system therefore opens up the possibility of designing better motif finders that directly optimize the E-value and we present two such algorithms in the next two sections.

[4] As shown there, the sFFT algorithm is much more efficient than the numerical method in [4].

[5] Note that the p-value is computed as the geometric mean of the bounds returned by sFFT.

Experiment	memo-sFFT	CONSENSUS
CRP-100	3.0	7.5
CRP-500	3.5	32.7
CRP-1000	4.2	65.1
CRP-5000	9.5	316.6

Table 1: **The advantage of using memo-sFFT in Conspv.** The columns memo-sFFT and CONSENSUS report the runtime (in seconds) for Conspv implemented with memo-sFFT and the large-deviation method in CONSENSUS respectively, for the various test sets. The CRP-X sets contain 18 sequences of length 108 and X specifies the number of alignments saved by Conspv in its beam search (corresponding to the `-q` option for CONSENSUS).

Experiment	CONSENSUS TPs	Conspv TPs
COMBO1	174	195
COMBO2	146	153
FIFTY1	62	145
FIFTY2	30	41

Table 2: **Tests on sequences of varied length.** The values reported here are the number of tests in which the reported motif has a significant overlap with the implanted motif (see section 6 for details) out of a total of 200 tests.

3 Optimizing for E-values - Conspv

The Conspv program in its simplest form adapts the CONSENSUS algorithm with the difference being that it uses E-values rather than entropy scores to compare alignments. More specifically, we implemented a version of the CONSENSUS algorithm under the OOPS model [2] and the `-pr2` option (save the best alignment extension). We also employed the memo-sFFT system described in section 2 to compute E-values. While the cost of computing E-values using this system is essentially a constant, this can still be a significant time penalty for Conspv. We therefore optimized its running time further by not computing the E-value for alignments that have too low an entropy score to be worthy of consideration. This is determined by keeping a lower bound for the entropy score based on alignments that do not make it into the list of best alignments[6].

When run on a set of sequences that have identical lengths the CONSENSUS algorithm (using the entropy score to compare alignments) can be seen as a greedy algorithm to optimize the E-value. However, on a set of sequences of varying lengths this is no longer the case. For such a set, CONSENSUS only optimizes the E-value indirectly. To test if this makes a difference to the performance of CONSENSUS, we compared it to Conspv on some of the test sets described in section 6. As can be seen from the results presented in Table 2, Conspv can significantly improve on the results of CONSENSUS. The improvement is most pronounced on the sets `COMBO1` and `FIFTY1` corresponding to sets where the sequence lengths are more diverged.

A major advantage of Conspv is that it lends itself naturally for searching over multiple motif widths. Since alignments are compared using E-values there is no need for heuristics such as the one used in WCONSENSUS [4]. To exploit this, we implemented a version of Conspv that takes a range of widths to search over as input[7]. The single width version of Conspv is then extended as follows: instead of ranking an alignment by its E-value for a given fixed width, we now rank

[6] Recall that CONSENSUS is a beam search algorithm that maintains a list of best alignments seen so far.

[7] A generalization to a set of allowed widths can also be easily implemented.

Experiment	Finders	TPs	Cov	Acc
COMBO3	WCONSENSUS	52	38	29
	WECons	49	40	42
	Conspv	89	76	74
GAP1	WCONSENSUS	57	46	43
	WECons	46	39	38
	Conspv	74	63	60

Table 3: **Comparison of motif finders based on CONSENSUS.** In the "TPs" column we report the number of tests where there is significant overlap with the implanted motif out of a total of 200 tests. Also, the "Cov" and "Acc" columns report the number of tests in which the overlap is a substantial fraction of the implanted (overlap-coverage) and reported (overlap-accuracy) motifs respectively. For details on the experiments and the overlap scores see section 6.

by the optimal E-value for widths in the given range. A naive alternative approach to this (that we refer to as WECons) is to run CONSENSUS for each of the widths in the given range and choose the motif with the optimal E-value.

The advantage of Conspv over WECons derives from the fact that the cost of running Conspv for r different widths (where the largest width is w_{max}) is much less than the cost of r runs of CONSENSUS. This is essentially because a majority of the running time of CONSENSUS is spent in evaluating extensions to alignments and this can be done in $O(w_{max})$ time in both CONSENSUS and Conspv. In practice, Conspv is a bit more than twice slower compared to a single run of CONSENSUS. This improved runtime is exploited by Conspv as follows: when searching for the best motif CONSENSUS maintains a list (of size q specified by the user) of the best motifs seen so far. When searching over w different widths, CONSENSUS would maintain w such lists independently. In Conspv a single, much larger list can be maintained for the same total runtime. This enables Conspv to devote more time looking at the promising motifs regardless of their width and thus do a better search of the motif space.

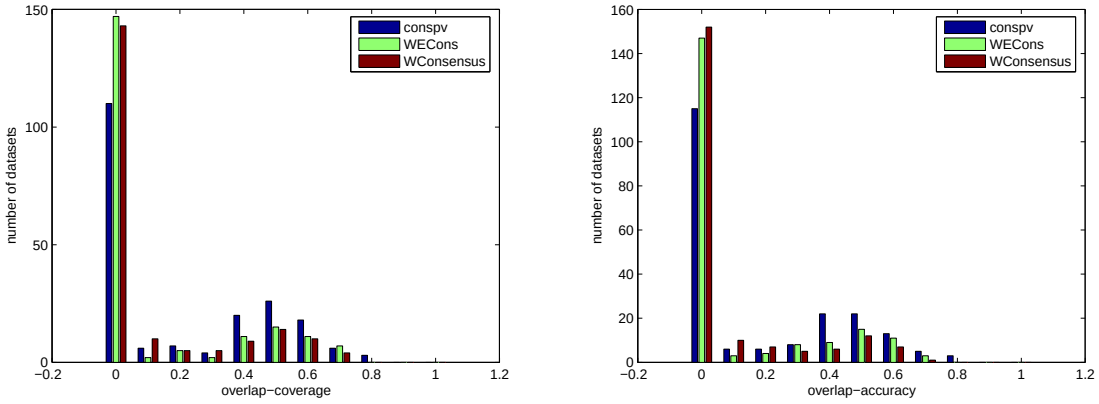
To assess the relative performance of Conspv, WECons and WCONSENSUS[8] we compared them over several synthetic datasets (see section 6 for details). The results were qualitatively similar across the datasets and a couple of them are presented in Table 3. As can be seen in Table 3, Conspv can improve substantially over WCONSENSUS and WECons in finding motifs that overlap the implanted motifs in our datasets. This is also uniformly true across the various overlap scores that we measured and for various thresholds of overlap (as indicated by Figure 2).

4 E-value based improvements of the Gibbs sampler

Having established that consideration of E-values can improve the performance of CONSENSUS we next look at the Gibbs sampler. In particular we look at the problem of unknown motif width. Lawrence et al. [7] considered several criteria for choosing the right width from multiple runs of their sampler, a run for each possible width. The criterion they eventually recommend is termed the "information per parameter" which is the incomplete-data log-probability ratio (22 in [7]) divided by the number of free parameters ($(A-1)w$). Below we refer to this version of Gibbs as WGibbs.

An obvious alternative to WGibbs in the spirit of WECons, which we call WEGibbs, is to choose the run with the width that optimizes the E-value instead of the original information

[8]Since WCONSENSUS does not allow the user to directly specify a range of widths we instead varied the bias parameter (the `-s` option) over the range $\{0.5, 1, 1.5, 2.0\}$ (as suggested in [4]).



(a) (b)

Figure 2: Histogram of the number of datasets as a function of the overlap score for the COMBO3 experiment and the various motif finders in Table 3.

Experiment	Finders	TPs	Cov	Acc
COMBO3	WGibbs	20	15	17
	WEGibbs	125	117	118
	Gibbspv	146	137	136
GAP1	WGibbs	17	13	11
	WEGibbs	77	64	60
	Gibbspv	95	82	79

Table 4: **Comparison of Gibbs samplers.** See Table 3.

per parameter. Using again the tests described in Section 6 we found that WEGibbs does a significantly better job than WGibbs at detecting the implanted motifs (see Table 4 and Figure 3). The next logical step is to ask whether a Gibbs analogue of Conspv that would more intimately link the E-values to the optimization procedure can further improve these results. To answer this question we designed Gibbspv, a new variant of the Gibbs sampling procedure.

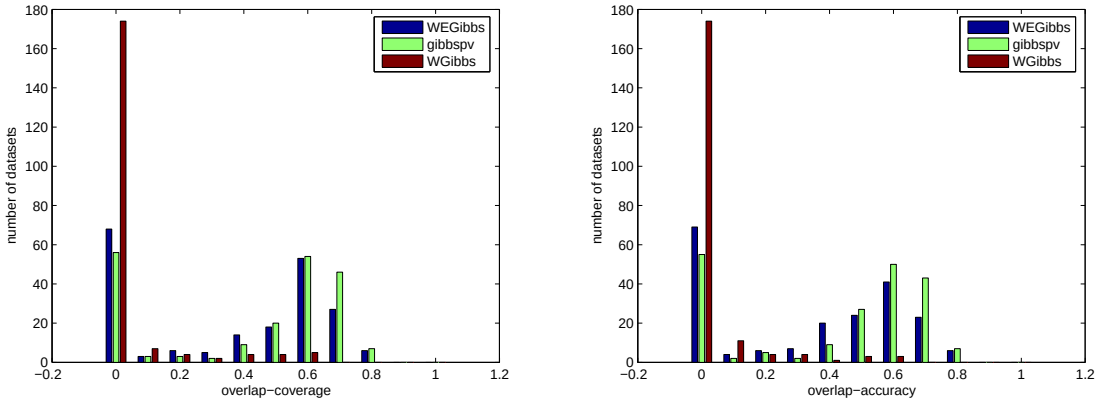
The original Gibbs-sampling motif finder begins each run by picking a random starting position in each sequence in the data set. The algorithm then sequentially applies the following two-step procedure to each of the sample sequences. The predictive update step computes a motif model Θ based on the current chosen set of starting positions[9]. The sampling step in turn randomly selects new candidate starting positions in the current sequence with probability proportional to the likelihood ratio of the position given the current model Θ. Each iteration of the Gibbs sampler consists of applying the aforementioned two-step procedure once to each of the input sequences.

Gibbspv cycles through a user specified number of iterations (default is `-C`=2) at the end of which five E-values are computed corresponding to the following five alignments[10]:

- The alignment of the currently chosen sites (width w)

[9]The model Θ is inferred from the starting positions by the rule $\Theta_{ij} = \frac{c_{ij}+b_i}{N-1+\sum_j b_j}$, where c_{ij} is the count of letter j in the i-th sequence of the alignment and b_j is an *a priori* chosen pseudocount to avoid 0 probabilities.

[10]Subject to the condition that the considered alignment is well defined and within the specified range of widths.



(a) (b)

Figure 3: Histogram of the number of datasets as a function of the overlap score for the COMBO3 experiment and the various Gibbs samplers in Table 4.

Experiment	Finders	TPs	Cov	Acc
COMBO3	Gibbspv	146	137	136
	Conspv	89	76	74
	GLAM	21	16	14
	MEME	39	28	31
GAP1	Gibbspv	95	82	79
	Conspv	74	63	60
	GLAM	17	7	8
	MEME	41	29	26

Table 5: **Comparison of Gibbspv and Conspv with MEME and GLAM.** See Table 3.

- The alignments of first/last $w-1$ columns of the currently chosen sites
- The alignment generated by adding the column to the right/left of the currently chosen sites (width $w+1$)

The algorithm then chooses the alignment with the best (smallest) E-values and continues as before. As in the original Gibbs sampler, if no improvement to the entropy score is detected in a specified number of iterations (-L) the program starts a new run.

Table 4 and Figure 3 confirm that by incorporating the E-values into its sampling strategy Gibbspv is better at detecting the implanted motifs in our experiments. In addition, as can be seen from the results in Table 5, Gibbspv can be substantially better than existing algorithms such as MEME [1] and GLAM [3] for finding motifs with unknown width.

5 Conclusion

In this paper we have demonstrated the utility of E-value calculations for designing better motif finders. For this purpose, memo-sFFT can serve as an accurate tool for efficiently computing a large number of E-values. In particular, for finding motifs of unknown width, the memo-sFFT

based gibbs sampler, Gibbspv can outperform several existing motif finders. Exploring the use of E-values for designing better motif finder for other motif models (such as ZOOPS [2]) can be a fruitful avenue for future research. The programs described in this paper will be available from http://www.cs.cornell.edu/~keich.

6 Methods

To test the various motif finders we constructed synthetic datasets with implanted motifs as follows: independent sequences with the specified lengths were sampled by choosing symbols at random from the four letter DNA alphabet according to a uniform, independent background frequency. A position was chosen uniformly at random from each sequence and an instance of a given profile Θ, generated as described below, was inserted in that position. The profiles used (see Table 6) are represented as a position weight matrix, a $4 \times w$ array of numbers where Θ_{ij} denotes the frequency of letter i in column j in all aligned instances of Θ. Since we wanted to have control over the implanted motifs the instances were essentially generated by permuting the columns of the alignment. Each column of the alignment matched the corresponding column of the profile up to discretizing effects.

For each of the experiments that we conducted, 200 datasets were generated for a given profile. The various motif finders were then run with parameter settings that allowed them to take from 9-10 minutes, to place them on an equal footing. Note that we were unable to do this for MEME as it does not employ any parameters that allow the control of running time. In all the experiments, MEME ran for much less than 9 minutes. This factor should be taken into account when judging the generally poor performance of MEME compared to the other motif finders. The details for the experiments that we conducted can be found in Table 8. Also, the various profiles used are shown in Table 6.

In general, the length of the sequences and the implanted profiles were chosen such that the motif finders we considered would have a non-trivial percentage of failures (i.e. datasets where they pick motifs with no overlap with the implants). These hard motif finding problems provide good test sets for discriminating between the various motif finders. Finally, an estimate of overlap for each data set and for each motif finder was computed in the following manner: Let a_n be the position of the implanted motif instance in the n^{th} sequence, let $\widehat{a_n}$ be the position of the motif reported by a motif finder and let w and $\widehat{w}$ be the respective widths of the motifs. Then we define the following overlap scores: overlap-coverage = overlap-x$(a, \widehat{a}, w)$, overlap-accuracy = overlap-x$(a, \widehat{a}, \widehat{w})$ and overlap = $\min\{\text{overlap-coverage}, \text{overlap-accuracy}\}$ where

$$\text{overlap-x}(a, \widehat{a}, x) = \max_{|i|<\frac{x}{2}} \left\{ \frac{x - |i|}{x} \cdot \frac{|\{n : a_n = \widehat{a_n} + i\}|}{N} \right\} \tag{1}$$

and N is the number of sequences in the dataset. To report a significant overlap between the implanted and the reported motif (true positives or TPs) we used a threshold of 0.1 for the overlap score. Also, for overlap-coverage and overlap-accuracy (corresponding to the columns "Cov" and "Acc" in Tables 3, 4 and 5) we used a threshold of 0.3.

7 Acknowledgements

This research uses computational resources funded by NIH grant 1S10RR020889.

Table 6: **The profiles used in our experiments.**

	COMBO				FIFTY				GAP			
Pos.	A	C	G	T	A	C	G	T	A	C	G	T
1	0.95	0.00	0.00	0.05	0.50	0.00	0.00	0.50	0.70	0.10	0.10	0.10
2	0.00	0.50	0.50	0.00	0.00	0.50	0.50	0.00	0.00	0.70	0.30	0.00
3	0.70	0.10	0.10	0.10	0.50	0.50	0.00	0.00	0.10	0.00	0.90	0.00
4	0.00	0.70	0.30	0.00	0.50	0.00	0.50	0.00	0.10	0.10	0.10	0.70
5	0.50	0.00	0.00	0.50	0.50	0.50	0.00	0.00	0.00	0.70	0.00	0.30
6	0.25	0.25	0.25	0.25	0.00	0.50	0.50	0.00	0.30	0.20	0.30	0.20
7	0.95	0.00	0.00	0.05	0.00	0.50	0.00	0.50	0.25	0.25	0.20	0.30
8	0.25	0.25	0.25	0.25	0.00	0.50	0.00	0.50	0.00	0.50	0.50	0.00
9	0.70	0.10	0.10	0.10	0.50	0.00	0.50	0.00	0.10	0.10	0.70	0.10
10	0.00	0.50	0.00	0.50	0.00	0.50	0.50	0.00	0.00	0.70	0.30	0.00
11	0.00	0.70	0.00	0.30	0.50	0.50	0.00	0.00	0.10	0.10	0.10	0.70
12	0.70	0.10	0.10	0.10	0.00	0.50	0.50	0.00	0.00	0.90	0.10	0.00
13	0.00	0.50	0.50	0.00	0.00	0.50	0.00	0.50	0.30	0.00	0.70	0.00

Table 7: **The parameter sets used in our experiments.** For the MULTI-WIDTH tests the motif-finders were used to search for motifs with widths in the range $[9, 17]$.

Parameter Set	Finder	Parameters
SINGLE-WIDTH	CONSENSUS	`-L 13 -c0 -q 4000`
	Conspv	`13 6000`
MULTI-WIDTH	WCONSENSUS	`-c0 -q 200`
	WECons	`-c0 -q 120`
	Conspv	`4000`
	GLAM	`-n8000 -r55 -z -a9 -b17`
	MEME	`-mod oops -nmotifs 1 -minw 9 -maxw 17 -dna -text -maxsize 1000000`
	WGibbs	`-d -n -t80 -L150`
	WEGibbs	`-d -n -t80 -L150`
	Gibbspv	`-t350 -L400`

Table 8: **Experiment details.** See Table 6 and 7 for details about the profiles and parameter sets used.

Experiment	Profile	Parameter Set	Sequences
COMBO1	COMBO	SINGLE-WIDTH	20 of length 500 & 20 of length 2500
COMBO2	COMBO	SINGLE-WIDTH	20 of length 1000 & 20 of length 2000
FIFTY1	FIFTY	SINGLE-WIDTH	20 of length 500 & 20 of length 2500
FIFTY2	FIFTY	SINGLE-WIDTH	20 of length 1000 & 20 of length 2000
COMBO3	COMBO	MULTI-WIDTH	30 of length 1000
GAP1	GAP	MULTI-WIDTH	30 of length 1000

References

[1] T.L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, Menlo Park, California, 1994.

[2] T.L. Bailey and C. Elkan. The value of prior knowledge in discovering motifs with meme. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 21–29, Menlo Park, California, 1995. AAAI Press.

[3] Martin C Frith, Ulla Hansen, John L Spouge, and Zhiping Weng. Finding functional sequence elements by multiple local alignment. *Nucleic Acids Res*, 32(1):189–200, 2004.

[4] GZ Hertz and GD Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7-8):563–77, 1999.

[5] JD Hughes, PW Estep, S Tavazoie, and GM Church. Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J Mol Biol*, 296(5):1205–14, Mar 2000.

[6] U. Keich. Efficiently computing the p-value of the entropy score. *J Comput Biol*, 12(4), 2005.

[7] CE Lawrence, SF Altschul, MS Boguski, JS Liu, AF Neuwald, and JC Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–14, Oct 1993.

[8] Niranjan Nagarajan, Neil Jones, and Uri Keich. Computing the P-value of the information content from an alignment of multiple sequences. *Bioinformatics*, 21 Suppl 1(ISMB 2005):i311–i318, Jun 2005.

[9] AF Neuwald, JS Liu, and CE Lawrence. Gibbs motif sampling: detection of bacterial outer membrane protein repeats. *Protein Sci*, 4(8):1618–32, Aug 1995.

[10] GD Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, Jan 2000.

[11] Martin Tompa, Nan Li, Timothy L Bailey, George M Church, Bart De Moor, Eleazar Eskin, Alexander V Favorov, Martin C Frith, Yutao Fu, W James Kent, Vsevolod J Makeev, Andrei A Mironov, William Stafford Noble, Giulio Pavesi, Graziano Pesole, Mireille Rgnier, Nicolas Simonis, Saurabh Sinha, Gert Thijs, Jacques van Helden, Mathias Vandenbogaert, Zhiping Weng, Christopher Workman, Chun Ye, and Zhou Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotechnol*, 23(1):137–44, Jan 2005.