BIOINFORMATICS

Vol. 19 Suppl. 2 2003, pages ii149–ii155
DOI: 10.1093/bioinformatics/btg1072

Finding subtle motifs by branching from sample strings

Alkes Price, Sriram Ramabhadran and Pavel A. Pevzner*

Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0114, USA

Received on March 17, 2003; accepted on June 9, 2003

ABSTRACT

Many motif finding algorithms apply local search techniques to a set of *seeds*. For example, GibbsDNA (Lawrence *et al.*, 1993) applies Gibbs sampling to random seeds, and MEME (Bailey and Elkan, 1994) applies the EM algorithm to selected *sample strings*, i.e. substrings of the sample. In the case of subtle motifs, recent benchmarking efforts show that both random seeds and selected sample strings may never get close to the globally optimal motif. We propose a new approach which searches motif space by branching from sample strings, and implement this idea in both pattern-based and profile-based settings. Our PatternBranching and ProfileBranching algorithms achieve favorable results relative to other motif finding algorithms.

Availability: http://www-cse.ucsd.edu/groups/bioinformatics/software.html

Contact: aprice@cs.ucsd.edu

INTRODUCTION

The goal of motif finding is to find an unknown motif with approximate occurrences at unknown positions in a sample of DNA sequences. Some motif finding algorithms carry out this search in the space of possible starting positions of all motif occurrences in the sample, while others search the space of all possible motifs described by a given model.

Popular algorithms which search the space of starting positions include the greedy CONSENSUS algorithm (Hertz and Stormo, 1999), and the stochastic GibbsDNA algorithm (Lawrence *et al.*, 1993), which applies Gibbs sampling to random seeds. However, the space of starting positions is typically large, and in the case of subtle motifs, greedily or randomly chosen points in this space may never get close to the globally optimal motif.

The alternative is to search in the space of motifs. Recently developed branch-and-bound techniques can find rather subtle motifs by exhaustively searching the space of motif *patterns* (Vanet *et al.*, 2000; Marsan and Sagot, 2000; Pavesi *et al.*, 2001; Apostolico *et al.*, 2002; Eskin and Pevzner, 2002). Though much more efficient than an unbounded exhaustive search, these approaches still have a high computational cost, particularly for longer motifs.

An advantage of searching in motif space is that, because the globally optimal motif has approximate occurrences in the sample, the search can be restricted to small neighborhoods (in motif space) of sample strings. This motivates a *sample-driven* approach in which selected sample strings are used as seeds for a local search. For example, MEME (Bailey and Elkan, 1994) uses a heuristic to choose (profiles derived from) one or more sample strings as seeds to the EM algorithm. However, in the case of subtle motifs, the selected sample strings may not be close to the globally optimal motif, and the sample-driven approach may converge to local optima (Fig. 1a). An alternative is the *extended sample-driven approach*, which searches neighborhoods of all sample strings via exhaustive search Waterman *et al.* (1984); Galas *et al.* (1985); Sagot *et al.* (1995); Sagot (1998). Recently, Keich and Pevzner (2002a) introduced a variant of the extended sample-driven approach which uses *multiprofiles* to restrict this exhaustive search. These approaches find subtle motifs, but have a high computational cost (Fig. 1b). We propose to search by *branching from sample strings* (Fig. 1c). Although our new technique is extremely simple (some would say trivial), it finds subtle motifs far more efficiently than previous methods. We note that there are some similarities between our technique and GibbsDNA (Lawrence *et al.*, 1993), which branches from random seeds. The difference is that we branch deterministically in small subsets of *motif space*, while GibbsDNA branches stochastically in the typically large *space of starting positions*.

Motif finding algorithms model a motif either as a *pattern* of l consensus nucleotides, representing the most likely nucleotide for each position of the motif, or as a *profile*, a $4 \times l$ matrix of nucleotide probabilities for each position of the motif. We refer to *pattern-based* and *profile-based* models, respectively. Recently developed algorithms, such as PROJECTION (Buhler and Tompa,

*To whom correspondence should be addressed.

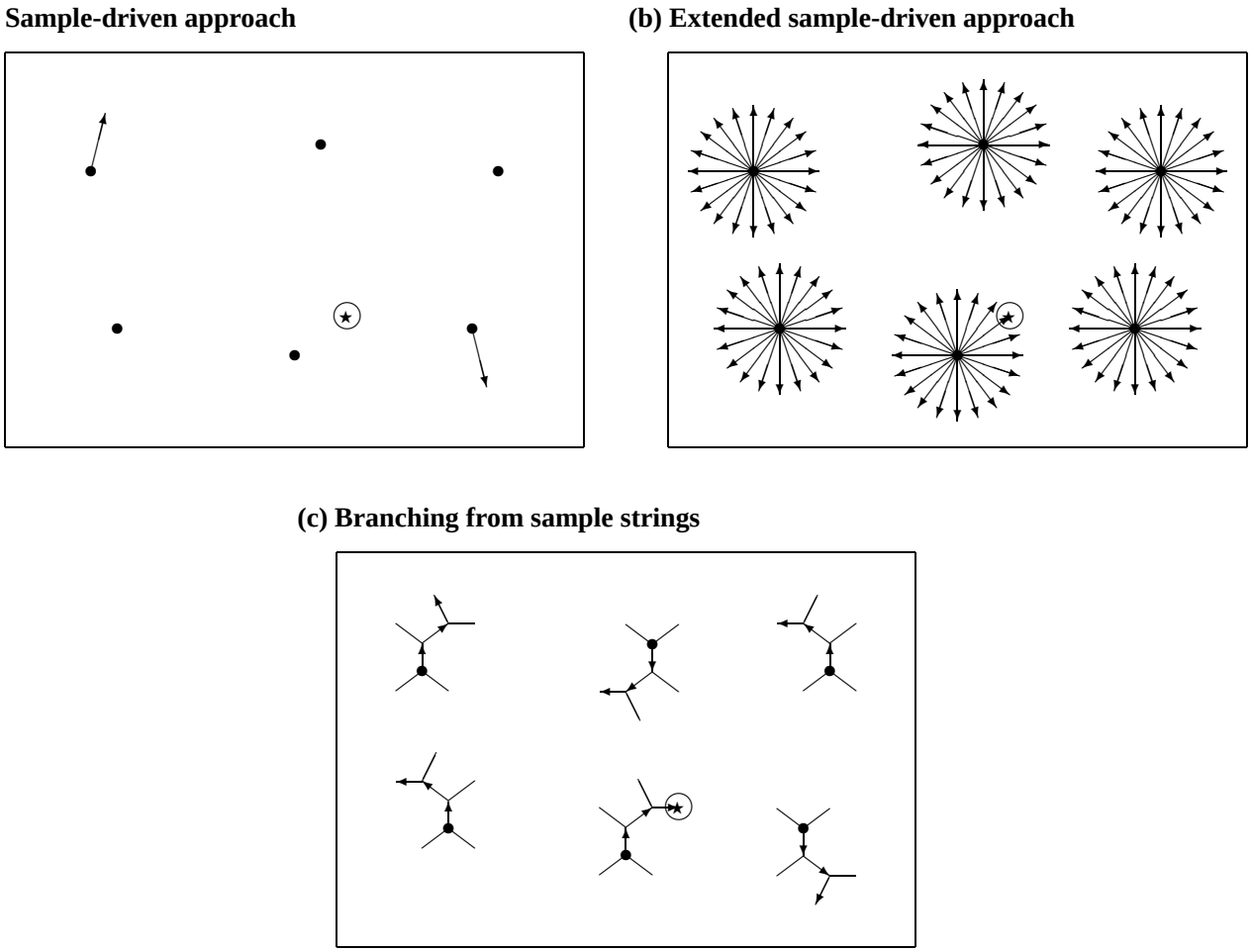

Fig. 1. Comparison of sample-driven, extended sample-driven and branching approaches to searching in motif space. Bullets • represent sample strings, circled stars ⋆ represent the globally optimal motif. (a) Sample-driven algorithms often fail to find the global optimum. (b) Extended sample-driven algorithms typically find the global optimum, but with high computational cost. (c) Branching from sample strings efficiently finds the global optimum.

2001), MITRA (Eskin and Pevzner, 2002) and MULTIPROFILER (Keich and Pevzner, 2002a), can find subtle motifs using the pattern-based model. However, biological motifs do not always fit this model. In particular, positions represented by a purine (R), pyrimidine (Y), weak bond (W) or strong bond (S), with two likely nucleotides instead of one, are common in biological motifs. Pattern-based algorithms have difficulty finding motifs with many degenerate positions of this sort. On the other hand, profile-based algorithms have difficulty finding subtle pattern-like motifs. We refrain from expressing a preference between pattern-based and profile-based algorithms, and implement our approach in each setting.

The paper is organized as follows: we begin by introducing a pattern-based algorithm, PatternBranching. Then, mindful of the greater generality of profiles, we present a profile-based variant, ProfileBranching. In each case, we compare results of our approach to other algorithms in finding subtle implanted motifs. Finally, we test our method on biological samples with known motifs.

THE PATTERNBRANCHING ALGORITHM

Pevzner and Sze (2000) stimulated interest in pattern-based motif finding algorithms by observing that other approaches were unable to solve the following Motif Challenge Problem: given a sample of $n = 20$ sequences, each $N = 600$ nucleotides long with an implanted pattern of length $l = 15$ with $k = 4$ mutations, find the pattern. Since then, various approaches have solved the Motif Challenge Problem, and even harder implantation problems, with increasing probability of success (Pevzner and Sze, 2000; Buhler and Tompa, 2001; Eskin and

Pevzner, 2002; Keich and Pevzner, 2002a). We will use the PatternBranching algorithm to solve the Motif Challenge Problem, and its harder versions, far more efficiently than other algorithms.

Let M be a motif, viewed as a pattern of length l, and let A_0 be an occurrence of M in the sample with exactly k mutations. Given A_0, how do we determine M? Since the Hamming distance $d(M, A_0) = k$, we have $M \in D_{=k}(A_0)$, defined as the set of patterns of distance exactly k from A_0. The extended sample-driven approach scores each of the $\binom{l}{k} \cdot 3^k$ patterns in $D_{=k}(A_0)$ (Waterman *et al.*, 1984; Galas *et al.*, 1985; Sagot *et al.*, 1995; Sagot, 1998), or in a carefully selected subset of $D_{=k}(A_0)$ (Keich and Pevzner, 2002a). However, since we will ultimately need to apply this procedure to all sample strings A_0 of length l, this is somewhat slow. We propose instead to construct a path of patterns

$$A_0 \longrightarrow A_1 \longrightarrow \ldots \longrightarrow A_k$$

by iteratively applying the `BestNeighbor` function, which maps a pattern A to its best neighbor in $D_{=1}(A)$, thus changing a single nucleotide of A. We then score A_k as a guess of M. More generally, if A_0 is a putative occurrence of M in the sample with at most k mutations, as opposed to exactly k mutations, we score A_j as a guess of M at each iteration j. This branching approach greatly reduces the number of neighbors to analyze, as compared to extended sample-driven approaches.

Two questions must be addressed: given a pattern A, how do we score it, and how do we define `BestNeighbor`(A)? First, we score the pattern A using its *total distance* from the sample. For each sequence S_i in the sample $\mathcal{S} = \{S_1, \ldots, S_n\}$, let $d(A, S_i) = \min\{d(A, P) \mid P \in S_i\}$, where P denotes an l-mer (i.e. a pattern of length l). Then the total distance of A from the sample is $d(A, \mathcal{S}) = \sum_{S_i \in \mathcal{S}} d(A, S_i)$. Second, we define `BestNeighbor`$(A)$ to be the pattern $B \in D_{=1}(A)$ with lowest total distance $d(B, \mathcal{S})$. The resulting algorithm is shamefully simple, but extremely powerful:

```
    PatternBranching(S, l, k)
Motif ← arbitrary motif pattern
   For each l-mer A_0 in S
      For j ← 0 to k
    If d(A_j, S) < d(Motif, S)
         Motif ← A_j
   A_{j+1} ← BestNeighbor(A_j)
         Output Motif
```

If we wish to conduct a more thorough search of $D_{=k}(A_0)$, we can keep a set $\mathcal{A}$ of r patterns at each iteration instead of a single pattern, defining `BestNeighbors`$(\mathcal{A})$ to be the set of r patterns $B \in D_{=1}(\mathcal{A})$, i.e. $B \in D_{=1}(A)$ for some $A \in \mathcal{A}$, with lowest total distance $d(B, \mathcal{S})$. Letting $\mathcal{A}_0 = \{A_0\}$, we thus have $|\mathcal{A}_0| = 1$ and $|\mathcal{A}_j| = r$ for $j > 0$.

We now describe two algorithmic details which speed up the algorithm. First, letting n be the number of sequences in the sample and N be the length of each sequence, we compute $d(A_0, \mathcal{S})$ in time $O(nN)$, instead of time $O(nNl)$, by sharing computations across different sample strings A_0. Second, in the computation of `BestNeighbor`(A_j), we efficiently approximate the total distance $d(B, \mathcal{S})$ of each pattern $B \in D_{=1}(A_j)$ by estimating $d(B, S_i) = \min\{d(B, P) \mid P \in S_i\}$ using only patterns $P \in S_i$ which satisfy two conditions: $d(A_j, P) \leq 2k - j$, which will be satisfied in the important case where P is an occurrence of the correct motif M with at most k mutations and the path of best neighbors $A_j \rightarrow \ldots \rightarrow A_k$ leads to M; and P agrees with B at the nucleotide changed from A_j, which will likely be true for the pattern $P \in S_i$ minimizing $d(B, P)$. By storing the values $d(A_j, P)$, we can quickly compute this estimate of $d(B, S_i)$ for all $B \in D_{=1}(A_j)$ with a single loop through l-mers $P \in S_i$.

We have also implemented the following optional speedups to the PatternBranching algorithm, typically reducing the running time by about a factor of 5. First, at iteration j, instead of fixing the number of patterns to keep, we can define `GoodNeighbors`$(\mathcal{A}_j)$ to be the set of all patterns $B \in D_{=1}(\mathcal{A}_j)$ with $d(B, \mathcal{S}) \leq \beta_j$ for some threshold β_j. Modeling β_j as a linear function of j, appropriate values of β_j can be heuristically determined in negligible computation time relative to the running time of the algorithm. Because the set `GoodNeighbors`$(\mathcal{A}_j)$ is often empty, this approach is faster. Second, following CONSENSUS (Hertz and Stormo, 1999) and MULTIPROFILER (Keich and Pevzner, 2002a), instead of performing the above branching steps for each l-mer A_0 in the sample $\mathcal{S}$, we can branch only from l-mers A_0 in selected *reference sequences* of the sample. This approach should be avoided in the case of corrupted samples unless we can choose reference sequence(s) which are known to contain a motif occurrence.

THE PROFILEBRANCHING ALGORITHM

The ProfileBranching algorithm is similar to PatternBranching, but since we will search in the space of motif profiles instead of the space of motif patterns, we make the following changes:

1. convert each sample string A_0 to a profile $X(A_0)$
2. generalize the scoring method to score profiles
3. modify the branching method to apply to profiles
4. use the top-scoring profile we find as a seed to the EM algorithm

To convert each sample string A_0 to a profile $X(A_0)$, we follow MEME (Bailey and Elkan, 1994). Let $A_0 = a_1 \ldots a_l$, where $a_w \in \{A, C, G, T\}$. Then $X(A_0)$ is defined to be the $4 \times l$ profile matrix (x_{vw}) which in column w has probability $x_{vw} = \frac{1}{2}$ for nucleotide $v = a_w$ and $x_{vw} = \frac{1}{6}$ for each other nucleotide v. The probability $\frac{1}{2}$ for nucleotide $v = a_w$ reflects both the uncertainty in how well the pattern A_0 describes the correct motif, and the fact that the correct motif itself is stochastic.

We replace the total distance score for patterns with the following entropy score for profiles: given a profile $X = (x_{vw})$ and a pattern $P = p_1 \ldots p_l$, let $e(X, P)$ be the log probability of sampling P from X, i.e. $e(X, P) = \sum_{w=1}^{l} \log(x_{p_w w})$. For each sequence S_i in the sample $\mathcal{S} = \{S_1, \ldots, S_n\}$, let $e(X, S_i) = \max\{e(X, P) \mid P \in S_i\}$. Then the entropy score of X is $e(X, \mathcal{S}) = \sum_{S_i \in \mathcal{S}} e(X, S_i)$. Intuitively, $e(X, \mathcal{S})$ describes how well X matches its best occurrence in each sequence of the sample. For a pattern A, $e(X(A), \mathcal{S})$ is equivalent to the total distance score $d(A, \mathcal{S})$, up to a linear transformation.

For a pattern A, the set $D_{=1}(A)$ is a natural choice of candidates for branching. For a profile $X = (x_{vw})$, we define $\mathcal{D}_{=1}(X)$ to be the set of profiles obtained from X by amplifying a single nucleotide in a single position w of X to create a profile $\bar{X} = (\bar{x}_{vw})$ with relative entropy equal to ρ, where ρ is an implicit parameter. The relative entropy is defined as $\sum_v x_{vw} \log(\bar{x}_{vw}/x_{vw})$, and we use the value $\rho = -0.3$. (We have made no effort to optimize the parameter ρ). For example, given nucleotide probabilities $(\frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6})$, by amplifying the second nucleotide we obtain $(0.27, 0.55, 0.09, 0.09)$. At a given position w, we will only amplify a nucleotide v if $x_{vw} < 0.5$.

The algorithm proceeds as follows. For each l-mer A_0 in the sample $\mathcal{S}$, we let $X_0 = X(A_0)$ and construct a path of profiles

$$X_0 \longrightarrow X_1 \longrightarrow \ldots \longrightarrow X_k$$

by iteratively applying the `BestNeighbor` function for profiles, which maps a profile X to a local improvement of its best neighbor in $\mathcal{D}_{=1}(X)$. The best neighbor is the profile $Y \in \mathcal{D}_{=1}(X)$ with highest entropy $e(Y, \mathcal{S})$, and we locally improve Y using its close matches in the sample in order to gain further information from the choice of Y. After branching for k iterations from each l-mer A_0 in the sample, we run the EM algorithm to convergence on the top-scoring profile we have found:

```
    ProfileBranching(S, l, k)
Motif ← arbitrary motif profile
    For each l-mer A_0 in S
        X_0 ← X(A_0)
        For j ← 0 to k
            If e(X_j, S) > e(Motif, S)
                Motif ← X_j
            X_{j+1} ← BestNeighbor(X_j)
        Run EM algorithm with Motif as seed
```

We now address the issue of running time. For a given sample string A_0, we can compute $d(A_0, P)$ for all $P \in S_i$, $S_i \in \mathcal{S}$ in time $O(nN)$, which is a lower bound on the work which must be done by any sample-driven algorithm. In the computation of `BestNeighbor`(X_j), we efficiently approximate the entropy $e(Y, \mathcal{S})$ of each profile $Y \in \mathcal{D}_{=1}(X_j)$ by estimating $e(Y, S_i) = \max\{e(Y, P) \mid P \in S_i\}$ using only patterns $P \in S_i$ which satisfy two conditions: $d(A_0, P) \leq K$, for a fixed parameter K; and P agrees with Y at the nucleotide which was amplified from X_j. The overall proportion of l-mers P which satisfy $d(A_0, P) \leq K$ is equal to the binomial probability $B[l, l-K, 0.25]$;[†] we typically choose K so that $B[l, l-K, 0.25] \approx 0.05$. By storing the values $e(X_j, P)$, we can quickly compute this estimate of $e(Y, S_i)$ for all $Y \in \mathcal{D}_{=1}(X)$ with a single loop through l-mers $P \in S_i$. Finally, we note that the time to run the EM algorithm to convergence on a single profile is very small. Thus, our algorithm takes time $O(n^2N^2) \cdot (1 + O(klB[l, l-K, 0.25]))$, versus a lower bound of $O(n^2N^2)$ for any sample-driven algorithm. For example, for $l = 15$, $k = 4$, $K = 8$ we have roughly an extra factor of 5 in running time. We admit that this is somewhat slow, but we believe that the optional speedups described at the end of the previous section, which we have implemented in PatternBranching but have not yet implemented in ProfileBranching, would alleviate some of the increase in running time.

RESULTS ON IMPLANTED MOTIFS

We begin by presenting results on the Motif Challenge Problem introduced by Pevzner and Sze (2000), who solved it using the WINNOWER and SP-STAR algorithms. However, those algorithms were unable to solve slightly harder implantation problems in reasonable time. This motivated the more powerful algorithms PROJECTION (Buhler and Tompa, 2001), MITRA (Eskin and Pevzner, 2002) and MULTIPROFILER (Keich and Pevzner, 2002a), which are able to solve much harder implantation problems. In Table 1, we list the success rate and running time of PatternBranching on the Motif Challenge Problem, versus each of these algorithms. We define success to mean that the algorithm outputs the implanted motif pattern. We omit from this comparison the algorithms from Pevzner and Sze (2000), for which results were not reported in this form. For each algorithm except MITRA, the choice of parameters leads to a trade-

[†] $B[n, m, p] = \sum_{i=m}^{n} \binom{n}{i} p^i (1-p)^{n-i}$.

Table 1. Results of various algorithms on the Motif Challenge Problem

Algorithm	Success Rate	Running Time
PROJECTION	about 100%	2 minutes
MITRA	100%	5 minutes
MULTIPROFILER	99.7%	1 minute
PatternBranching	99.7%	3 seconds

PROJECTION results from Buhler and Tompa (2001), MITRA results from Eskin and Pevzner (2002), MULTIPROFILER results from our own experiments[‡]. The success rate of about 100% for PROJECTION is estimated, based on 20 trials (more trials are required to estimate the success rate more accurately). The success rate of 100% for MITRA reflects the fact that MITRA is an exhaustive search algorithm. PROJECTION run on 667 MHz processor (Buhler and Tompa, 2001), MITRA on 750 MHz processor (Eskin and Pevzner, 2002), MULTIPROFILER and PatternBranching on 1.0 GHz processor. PatternBranching results incorporate the optional speedups described above

off between success rate and running time, and results in the table reflect selected parameter choices.

The Motif Challenge Problem can be made harder in two ways: first, one can increase the sequence length N; second, one can reduce the motif length l from 15 to 14, which increases the mutation rate. However, a very subtle implanted motif may be *dim* with respect to the total distance score $d(M, \mathcal{S})$, i.e. random motifs M may score better than the implanted motif (Keich and Pevzner, 2002b). Following PROJECTION and MULTIPROFILER, in these cases we instead use the sequence count score $SQ(M, \mathcal{S})$, which counts the number of sequences $S_i \in \mathcal{S}$ with $d(M, S_i) > k$. Examples of very subtle motifs include $(15, 4)$-motifs with $N = 2000$, and $(14, 4)$-motifs with $N = 800$. On these very hard implantation problems, PROJECTION's success rate drops to 80% or lower (Buhler, 2001), MITRA results were not reported in Eskin and Pevzner (2002), and MULTIPROFILER succeeds, with success rate arbitrarily close to 1 depending on parameter choices; for example, on $(15, 4)$-motifs with $N = 2000$, Keich and Pevzner (2002a) report that MULTIPROFILER achieves a success rate above 99% in 1.25hr on a 500Mz processor. PatternBranching succeeds much more quickly than MULTIPROFILER, taking a few minutes or less on a 1.0GHz processor to achieve a success rate above 99% on each of these problems. In summary, PatternBranching solves these pattern implantation problems much more quickly than any other algorithm we are aware of.

We now present results of ProfileBranching on the Motif Challenge Problem, versus other profile-based algorithms. Because profile-based algorithms output a profile instead of a pattern, success rate is not an appropriate figure of merit, and we instead use the *performance coefficient* of Pevzner and Sze (2000). Let $\mathcal{K}$ be the set of $n \cdot l$ implanted motif positions in the sample, and let $\mathcal{P}$ be the set of predicted motif positions. Then the performance coefficient is defined to be $|\mathcal{K} \cap \mathcal{P}|/|\mathcal{K} \cup \mathcal{P}|$. In Table 2, we list the performance coefficient and running time of ProfileBranching on the Motif Challenge Problem, versus CONSENSUS (Hertz and Stormo, 1999), GibbsDNA (Lawrence *et al.*, 1993) and MEME (Bailey and Elkan, 1994), which also model motifs using profiles. We admit that our algorithm is much slower than MEME. However, we believe that thorough software optimization would bring us closer to the factor of 5 in running time described above, and that the optional speedups which we have not yet implemented in ProfileBranching would reduce the running time further.

Table 2. Results of various profile-based algorithms on the Motif Challenge Problem

Algorithm	Perf. Coeff.	Running Time
CONSENSUS	0.20	40 seconds
GibbsDNA	0.32	40 seconds
MEME	0.14	5 seconds
ProfileBranching	0.57	80 seconds

Benchmarking of CONSENSUS, GibbsDNA and MEME kindly provided by Neil Jones. Performance coefficients averaged over 100 trials. All algorithms run on 1.0GHz processor

PatternBranching clearly outperforms ProfileBranching on pattern-like motifs as represented by the Motif Challenge Problem. However, pattern-based algorithms have difficulty finding motifs with many degenerate positions. To demonstrate the greater generality of the profile-based approach, we implant $(15, 5)$-motifs in $n = 20$ sequences of length $N = 600$, with the further restriction that all mutations of a given motif position mutate to a fixed secondary nucleotide value, so that each motif position has only two possible nucleotide values. This implanted motif is dim with respect to pattern-based total distance or sequence count scores (Buhler and Tompa, 2001), thus this implantation problem cannot be solved by pattern-based algorithms. Indeed, the average performance coefficient is only 0.10 for PatternBranching, versus 0.63 for MEME and 0.99 for ProfileBranching. As we gradually make the implanted motif more subtle, ProfileBranching continues to outperform MEME. For example, if in each occurrence of the motif we allow 1 of 5 mutations to mutate to a third nucleotide value for that motif position, average performance coefficients decline to 0.03 for PatternBranching, 0.03 for MEME and 0.62 for ProfileBranching. In summary, ProfileBranching is very successful in finding subtle implanted motifs where motif positions are dominated

[‡] Appropriate parameters settings for MULTIPROFILER were verified through personal correspondence with Uri Keich.

Table 3. Results of PatternBranching on biological samples

Sample	Sample size (bp)	PatternBranching motif	Reference motif	Ref.
E.coli CRP	1890	TGTGAAATAGATCACATTTT	TGTGANNNNGNTCACA	(A)
preproinsulin	7689	GCAGACCCAGCACCAGGGAA	AGACCCAGCA	(B)
		GAAATTGCAGCCTCAGCCCC	CCTCAGCCCC	(B)
		CCCTAATGGGCCAGGCGGCA	CCCTAATGGGCCA	(B)
DHFR	800	TGCAATTTCGCGCCAAACTT	TTCGCGCCAAACT	(B)
metallothionein	6823	CTCTGCGCCCGGCCCGGTTC	TTGCGCCCGG	(B)
		GGGAGCTCTGCACACCGCAC	AGCTCTGCACTC	(B)
c-*fos*	3695	CCATATTAGGACATCTGCGT	CCATATTAGGACATCTG	(C)
Yeast ECB	5000	GTATTTCCCGTTTAGGAAAA	TTTCCCNNTNAGGAAA	(C)

We list the motif(s) from PatternBranching output which match the reference motif(s), underlining the areas which match. References: (A) (Stormo and Hartzell III, 1989), (B) (Blanchette, 2001), (C) (Buhler and Tompa, 2001). Running times ranged from less than 1 second for DHFR to 6 seconds for preproinsulin on a 1.0GHz processor

by two frequently occurring nucleotide values, a feature of many biological motifs.

RESULTS ON BIOLOGICAL SAMPLES

We tested PatternBranching on the following biological samples with known motifs: a sample containing CRP binding sites in *E.coli* (Stormo and Hartzell III, 1989); four samples of upstream regions in a variety of organisms of each of the following eukaryotic genes: preproinsulin, dihydrofolate reductase (DHFR) and metallothionein (Blanchette, 2001), and c-*fos* (Buhler and Tompa, 2001); and a sample of promoter regions from yeast which are known to contain a shared promoter (Buhler and Tompa, 2001). These samples are also analyzed in Keich and Pevzner, 2002a. We set $l = 20$ and $k = 5$, and modified the algorithm to save 20 motifs with lowest total distance score. Table 3 shows that PatternBranching finds the known reference motif(s) in each sample. We note that, for the preproinsulin sample, some of the motifs returned by PatternBranching have a better total distance score than any of the reference motifs.

We also tested the ProfileBranching algorithm on these biological samples, again using $l = 20$ and $k = 5$. For each sample, the consensus pattern of the motif profile returned by ProfileBranching similarly matches one of the reference motifs from Table 3. Running times ranged from less than 1 second for DHFR to 18 seconds for preproinsulin on a 1.0GHz processor. We have not yet modified the algorithm to output more than one motif; this modification would be necessary to find all reference motifs in the preproinsulin and metallothionein samples.

We admit that the motifs in these biological samples have all been found by popular motif finding algorithms such as MEME (Bailey and Elkan, 1994). We are not aware of any experimentally verified motifs which are sufficiently subtle to demonstrate the advantage of our approach.

DISCUSSION

We have described a new method of finding motifs by branching from sample strings. This approach restricts the search to small neighborhoods (in motif space) of sample strings, and searches these neighborhoods with great efficiency. The PatternBranching and ProfileBranching algorithms implement this idea in pattern-based and profile-based settings, respectively. Both algorithms achieve favorable results in finding subtle implanted motifs, and succeed in finding known motifs in biological samples.

The next step is to apply these algorithms to more challenging biological samples. Our efficient approach is well suited to the analysis of larger samples. A separate question is how it will fare on corrupted samples containing many sequences without a motif occurrence. Preliminary results on implanted motifs indicate that we can find subtle motifs in large samples where most sequences do not contain a motif occurrence.

An intriguing idea which we have not yet explored is to extend the PatternBranching algorithm to an alphabet of motif letters which contains not only the nucleotides A,C,G,T but also letters corresponding to purine (R), pyrimidine (Y), weak bond (W) and strong bond (S), which each represent two likely nucleotide values. This would address the main weakness of the PatternBranching algorithm, namely its inability to find motifs containing many such degenerate positions.

An advantage of the ProfileBranching algorithm which we have not yet mentioned is that, like MEME (Bailey and Elkan, 1994), it can apply a prior distribution on the nucleotide probabilities in each position of the motif profile (Bailey and Elkan, 1995). Preliminary results indicate that imposing a Dirichlet mixture prior in this fashion greatly improves the results of ProfileBranching on the Motif Challenge Problem, but does not improve the results of MEME; this merits further investigation. Following MEME, the ProfileBranching algorithm can

also be extended to incorporate models of the underlying background distribution (for example, to handle genomes with different base compositions), and to output multiple motifs by probabilistically erasing occurrences of previously discovered motifs.

ACKNOWLEDGEMENTS

We are grateful to Neil Jones for benchmarking CONSENSUS, GibbsDNA and MEME on the Motif Challenge Problem, and to Uri Keich for helpful discussions and verifying appropriate parameter settings for MULTIPROFILER.

REFERENCES

Apostolico,A., Bock,M. and Lonardi,S. (2002) Monotony of surprise and large-scale quest for unusual words. *Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology (RECOMB-02)*. ACM Press, Washington, DC, pp. 22–31.

Bailey,T. and Elkan,C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology (ISMB-94)*. AAAI Press, Menlo Park, CA, pp. 28–36.

Bailey,T. and Elkan,C. (1995) The value of prior knowledge in discovering motifs with MEME. *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology (ISMB-95)*. AAAI Press, Cambridge, England, pp. 21–29.

Blanchette,M. (2001) Algorithms for phylogenetic footprinting. *Proceedings of the Fifth Annual International Conference on Research in Computational Molecular Biology (RECOMB-01)*. ACM Press, Montreal, Canada, pp. 49–58.

Buhler,J. (2001) *Search Algorithms for Biosequences Using Random Projection*, Ph.D. Thesis, University of Washinton.

Buhler,J. and Tompa,M. (2001) Finding motifs using random projections. *Proceedings of the Fifth Annual International Conference on Research in Computational Molecular Biology (RECOMB-01)*. ACM Press, Montreal, Canada, pp. 69–76.

Eskin,E. and Pevzner,P. (2002) Finding composite regulatory patterns in DNA sequences. *Bioinformatics*, Proceedings of the Tenth International Conference on Intelligent Systems for Molecular Biology(ISMB-02), **S1**, pp. 354–363.

Galas,D., Eggert,M. and Waterman,M. (1985) Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from *Escherichia coli*. *J. Mol. Biol.*, **186**, 117–128.

Hertz,G. and Stormo,G. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, **15**, 563–577.

Keich,U. and Pevzner,P. (2002a) Finding motifs in the twilight zone. *Bioinformatics*, **18**, 1374–1381.

Keich,U. and Pevzner,P. (2002b) Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics*, **18**, 1382–1390.

Lawrence,C., Altschul,S., Boguski,M., Liu,J., Neuwald,A. and Wootton,J. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**, 208–214.

Marsan,L. and Sagot,M. (2000) Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *J. Comput. Biol.*, **7(3-4)**, 345–362.

Pavesi,G., Mauri,G. and Pesole,G. (2001) An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics*, Proceedings of the Ninth International Conference on Intelligent Systems for Molecular Biology (ISMB-01), **S1**, pp. 207–214.

Pevzner,P. and Sze,S. (2000) Combinatorial approaches to finding subtle signals in DNA sequences. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB-00)*. AAAI Press, San Diego, California, pp. 269–278.

Sagot,M. (1998) Spelling approximate or repeated motifs using a suffix tree. *Lecture Notes in Computer Science*, **1380**, 111–127.

Sagot,M., Escalier,V., Viari,A. and Soldano,H. (1995) Searching for repeated words in a text allowing for mismatches and gaps. *Proceedings of the Second South American Workshop on String Processing*. Valparaiso, Chile, pp. 87–100.

Stormo,G. and Hartzell III,G. (1989) Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl Acad. Sci. USA*, **86**, 1183–1187.

Vanet,A., Marsan,L., Labigne,A. and Sagot,M. (2000) Inferring regulatory elements from a whole genome. An analysis of Helicobacter pylori sigma(80) family of promoter signals. *J. Mol. Biol.*, **297(2)**, 335–353.

Waterman,M., Arratia,R. and Galas,E. (1984) Pattern recognition in several sequences: consensus and alignment. *Bull. Math. Biol.*, **46**, 515–527.