Improvement of the PROJECTION Motif Finding Algorithm

Mohammed Alam, Warren Cheung,
Juan Estrada, James King
{malam,wcheung,estrada,king}@cs.ubc.ca

December 8, 2003

Abstract

In 2001 Buhler and Tompa [BT01] introduced their PROJECTION algorithm for solving the motif discovery problem for motifs of fixed length l. The algorithm finds good candidates for local optimisation by projecting all l-mers present in the input onto a smaller subspace and analysing dense points in that subspace. We have developed AGGREGATION, a modified version of PROJECTION that analyses dense regions of the subspace rather than dense points. Our subtle modification more than doubles the speed of the algorithm for many problem spaces while maintaining the same level of accuracy.

1 Introduction

Given a number of DNA sequences, the motif finding problem is the task of discovering a particular base sequence that appears (perhaps in a slightly mutated form) in every given sequence. We are considering the *challenge problem* as defined by Pevzner and Sze[PS00], and as stated by Buhler and Tompa[BT01]:

> **Planted (l, d)-Motif Problem:** Let M be a fixed but unknown nucleotide sequence (the motif consensus) of length l. Suppose that M occurs once in each of t background sequences of common length n, but that each occurrence of M is corrupted by exactly d point substitutions in positions chosen independently at random. Given the t sequences, recover the motif occurrences and the consensus M.

Much work has been done on the problem, but there are certain variations of the problem for which satisfactory solutions have not yet been found. The problem becomes harder if, for example, insertions/deletions are considered to be legal mutations or the motif does not occur in all of the t background sequences.

2 Related Work

Pevzner and Sze [PS00] posed their challenge problem for motif discovery and introduced two new algorithms that were more successful than previous attempts had been. WINNOWER treats every occurring l-mer as a node in a graph, with nodes adjacent if and only if they differ in at most $2d$ positions and they occur in different sequences. It then finds cliques of size t and works from those starting points. SP-STAR tries testing every occurring l-mer in turn and considers the possibility that it is a mutated occurrence of the motif consensus. Using this technique it essentially does an enumerative search, but only over the l-mers occurring within data rather than the entire space of 4^l l-mers.

More recently there have typically been two approaches used by motif discovery algorithms. Profile-driven approaches make use of weight matrices to calculate the score of different candidate motifs, returning the highest scoring motif as an answer. Pattern-driven approaches search for a motif of length l by assigning each of all possible 4^l l-letter patterns a score based on its number of occurrences in the sample (or another more sophisticated function). Clearly the latter is impractical for large l. Therefore a variation of the pattern approach — the sample-driven approach — limits the set of possible candidates but may have difficulty detecting subtle motifs. Sinha and Tompa [ST00] designed an enumerative search algorithm for yeast (YMF, Yeast Motif Finder), which illustrates the reliability of the pattern-driven approach. However, this program requires as input the specific characteristics of the motif that is being sought and is only intended for finding short motifs.

More recently, Keich and Pevzner [KP02a] designed MULTIPROFILER, a motif discovery algorithm that balances the pattern-driven and sample-driven approaches to yield a better ability to detect subtle motifs while avoiding the complexity of pure pattern-driven approaches. They also propose an objective tool for analysing the performance of motif finding algorithms [KP02b], with particular focus on hard to detect, or 'subtle' motifs. They suggest that in these cases even the most reliable algorithms are more prone to picking up randomly occurring spurious motifs simply due to the overwhelming presence of spurious motifs that, according to the scoring models used, are as good or better than the planted motif.

Rocke [Roc00] has proposed a modified version of the Gibbs sampling algorithm (GibbsDNA) of Lawrence et al. [LAB$^+$93] that uses suffix trees for the purpose of gapped motif discovery. Pevzner and Sze also provide extensions to their SP-STAR algorithm to handle gapped and variable length motifs.

GuhaThakurta and Stormo [GS01] have noted that many of the regulatory signals that appear in unaligned DNA sequences are composite patterns that are groups of monad (single motif) patterns occurring near each other. Eskin and Pevzner [EP02] have noted that it is difficult to find composite patterns using the current monad-based approaches, as such patterns are often composed of one or more monad signals that are 'too weak' (i.e. not statistically meaningful enough). They proposed a monad pattern discovery algorithm, MITRA (MIsmatch TRee Algorithm), for discovering composite patterns by first considering the problem as a larger monad pattern discovery problem through preprocessing of the sample to be analysed.

3 The Projection Algorithm

In 2002, Buhler and Tompa [BT01] introduced their Projection algorithm. This algorithm 'projects' every l-mer from the given data onto a smaller space by hashing. The hash function is based on k of the l positions that are selected at random when the algorithm begins. l-mers are hashed into the same bucket if they have the same bases in those k positions. In this way, l-mers are hashed to k-mers; these are the fingerprints that correspond to the buckets.

The idea behind Projection is that background l-mers (essentially random noise) will be distributed fairly evenly between the buckets. Meanwhile, the 'planted bucket' (i.e. the bucket into which the motif consensus would be hashed) will have additional l-mers because some occurrences of the planted motif will not be mutated in any of the k hashable positions. Projection performs this hashing, then performs refinement on each sufficiently full bucket to find the best motif in the vicinity of the l-mers in that bucket. The algorithm is likely to find the consensus motif if it ends up refining the planted bucket. Projection also sometimes finds the consensus motif by refining a bucket that is close to the planted bucket. By running the algorithm for a sufficient number of iterations, Projection will refine the planted bucket in at least one run with high probability.

Projection performs significantly better than many other motif finding algorithms, especially for harder instances of the problem such as $(14, 4)$, $(16, 5)$, and $(18, 6)$. For this reason, we have decided to concentrate on improving Projection, which has found great success in the l-mer hashing technique.

4 Modification of Projection

The majority of Projection's running time is taken up by the refinement stage that finds the best motif in the neighbourhood of a given bucket. For this reason, it would be especially advantageous to refine as few buckets as possible in our search for the planted bucket. The difficulty is that refining more buckets is currently advantageous in that it increases the probability that we refine the planted bucket at some point.

We have modified the projection process in a way that makes projection more complicated and time consuming, but allows us to refine few enough buckets that the time gained in refinement more than makes up for the time lost in projection. For the refinement stage of the algorithm, Buhler and Tompa used the expectation maximisation (EM) algorithm developed by Lawrence and Reilly [LR90]. We have left this stage of the algorithm completely unmodified.

A major problem with the projection method is that most planted occurrences of the motif consensus will be hashed to buckets other than the planted bucket, essentially being thrown away and considered as noise. The probability that many of these occurrences will be hashed to the planted bucket is small. However, a significant number of motif occurrences will land near, but not in, the planted bucket. Rather than simply refining any bucket that contains a sufficiently large number of l-mers, we refine any bucket *whose neighbourhood* contains a sufficiently large number of l-mers. We define the neighbourhood of a bucket B to be the bucket itself, plus all buckets whose corresponding fingerprints are at Hamming distance 1 from the fingerprint corresponding to B. We could extend this neighbourhood to include buckets further from B, thereby further increasing the probability of a planted occurrence falling into the neighbourhood of the planted bucket, but this would make our threshold tests prohibitively expensive.

We will refer to the number of l-mers hashed to a bucket's neighbourhood as that bucket's *score*. Because this value is obtained by aggregating the scores of individual buckets, we will refer to our modified method as AGGREGATION. By performing this score aggregation we essentially focus the effect of the significant data (planted motif occurrences) in relation to the noise (randomly occurring l-mers). By doing this, we improve the chances of sending the planted bucket to refinement. This means that fewer total refinements are required and the algorithm therefore runs faster.

5 Analysis

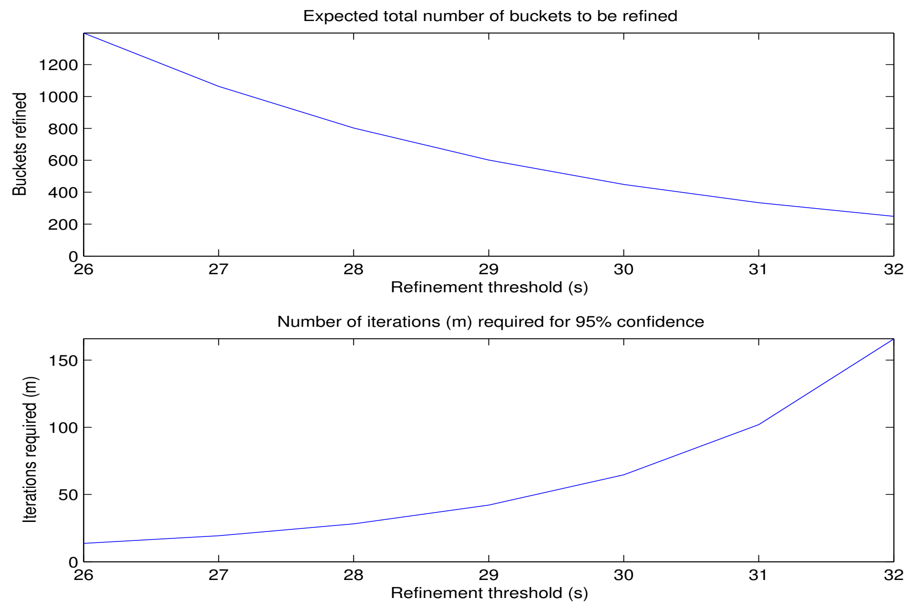
Mathematical analysis of Aggregation (and Projection, for that matter) is impractical unless we make the assumption that background l-mers are independently distributed. This, of course, is not true because some l-mers in the original data overlap each other. However, the overlap is sufficiently short relative to the background sequence that analysis under our assumption is applicable to the real case.

For any given set sequences, Projection's probability of discovering the consensus motif in an iteration is fixed. That is, with some fixed probability the projection positions will be chosen in such a way that the consensus motif is found in the refinement. Therefore, for a given set of input sequences, we can consider one iteration of Projection to be a Poisson trial. The expected number of times the consensus is recovered in m iterations follows a simple binomial distribution. Also, the probability that we *do not* recover the consensus decays exponentially as m increases (assuming that Projection is capable of recovering said consensus at all). Naturally, the same things can be said for Aggregation.

The probability of refining the planted bucket in a single iteration will vary depending on the input data. In terms of performing meaningful analysis we can approximate the expectation of this probability, which is particularly relevant to the challenge problem. It should be noted that this probability is not exactly the same as the probability of recovering the consensus motif since refinement of the planted bucket will not always yield the consensus and the consensus can be refined from buckets other than the planted one.

By determining the expected probability of refining the planted bucket and the expected probability of refining any other bucket we can make some theoretical comparisons between Projection and Aggregation, as well as choose good threshold values for Aggregation. Choosing a good threshold is a tradeoff — a threshold that is too low will send too many spurious buckets to refinement whereas a threshold that is too high will cause the algorithm to require too many iterations (see Figure 1). To compare Projection and Aggregation we use our analysis to determine how many iterations we need in order to refine the planted bucket with 95% probability and how

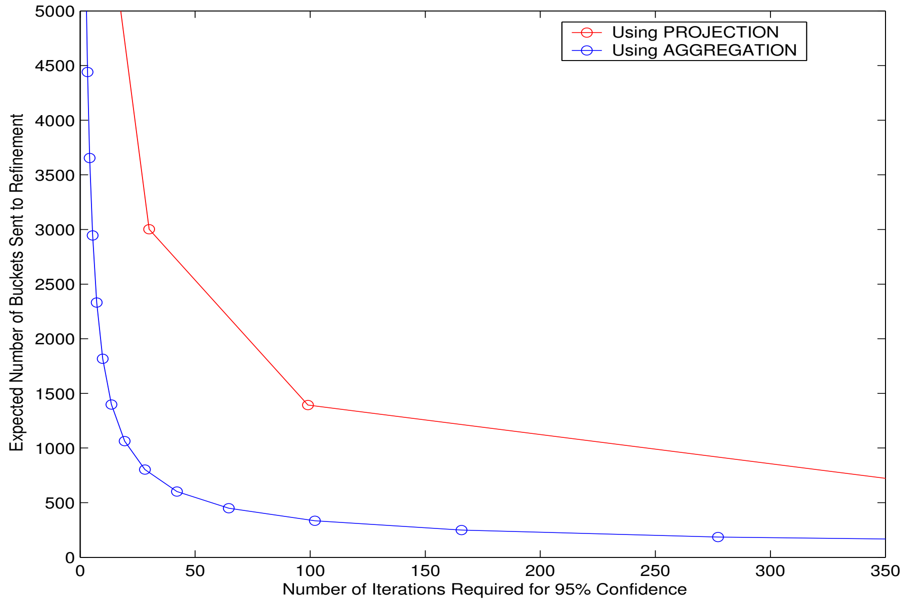
Figure 1: Expected number of buckets refined and iterations required for AGGREGATION with varying threshold. A good threshold keeps both of these values low; in this example 28, 29, and 30 would make good thresholds.



many buckets we should expect to refine during that process. We choose thresholds for AGGREGATION that keep both of these values low. In terms of comparison our analysis suggests that, in general, AGGREGATION can achieve 95% confidence after refining less than half as many buckets as PROJECTION, given that we want to run both methods for the same number of iterations (see Figure 2).

Because of the simplifications made in our analysis and the fact that AGGREGATION will be slower per iteration than PROJECTION, we cannot consider this to be any kind of proof that AGGREGATION is the superior method. Fortunately we see similar results in our experiments.

Figure 2: Theoretical comparison of PROJECTION and AGGREGATION with varying thresholds. This graph suggests that AGGREGATION does not need to refine as many buckets as PROJECTION.



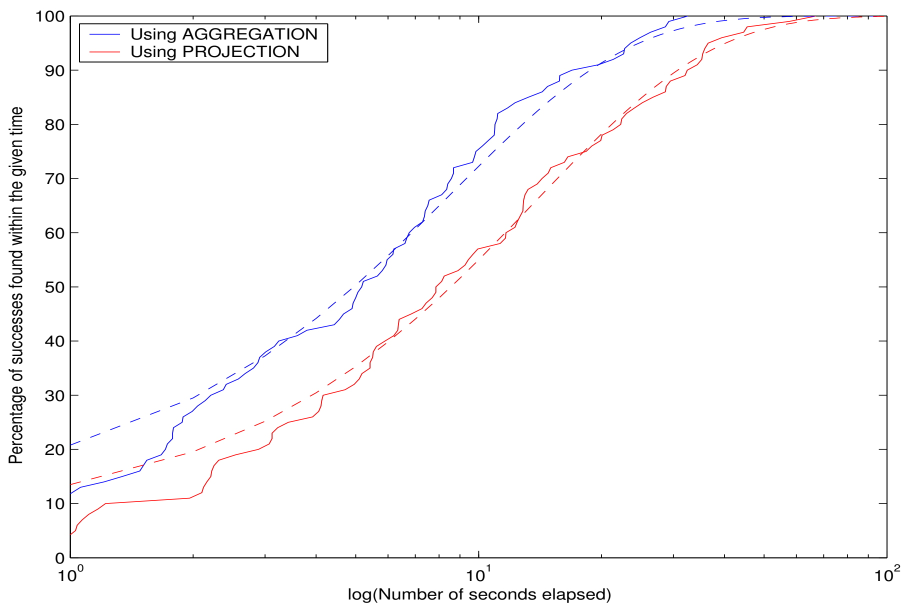
6 Experimentation

As stated before, for any fixed input we can consider one iteration of PROJECTION or AGGREGATION to be a Poisson trial. Therefore, in comparing the performance of PROJECTION to that of AGGREGATION for a fixed input, the only two things that need to be measured are the probabilities of the Poisson trials and the average iteration runtimes. The probabilities can be approximated by running the algorithms for a large value of m and, for each algorithm, dividing the number of successful iterations by m. The average iteration times, of course, can be determined by dividing the total runtimes of the two methods by m. Combining these two values we can obtain the expected number of successes per second.

Doing this in our experimentation gives us a fair, linear measurement of quality in terms of successes per second. That is, for each method we can determine the expected number of times the consensus motif will be recovered in one second of CPU time. Of course, this is all for a fixed set of input sequences. To measure the quality over a number of different inputs we must do this for each input and take the average of the measurements. For synthetic data this can be done quite easily since we can sample our inputs uniformly at random from whatever problem space we want.

We should note that, since a single iteration is a Poisson trial, the length of time for which the algorithm runs until the first success follows a geometric distribution. We simply want the Poisson trials of AGGREGATION to have a greater success probability than those of PROJECTION. For a given input we can analyse these running times and fit a geometric curve to our experimental data to approximate the success probabilities of our Poisson trials. Figure 3 gives a typical example of such data with the fitted geometric curves.

Figure 3: The geometric distributions exhibited by the runtimes of AGGREGATION and PROJECTION on the same fixed input (when we stop at the first success). Dashed lines are the fitted curves of true geometric distributions.



7 Benchmarking

To evaluate the efficacy of the AGGREGATION enhancement to the PROJECTION algorithm, both algorithms were tested on randomly generated synthetic data. AGGREGATION was modified from the original PROJECTION algorithm only to implement the hash-based random access data structure to allow for looking at all the neighbours of an arbitrary bucket. The parameters $(k = 7, s = 4, m)$ chosen for PROJECTION were the same as in [BT01], and the parameters for AGGREGATION were set to $(k = 7, s = 28)$ (see Figure 1 to see why this threshold is reasonable). The number of iterations m for AGGREGATION was set to twice the value of m selected for PROJECTION to ensure that the two methods were given comparable amounts of CPU time without AGGREGATION being given an advantage (actual runtimes are shown in Table 1). For each of the (l, d) conditions, 100 randomly generated sets of input sequences $(t = 20, n = 600)$ were generated, and PROJECTION and AGGREGATION were both run once on each of these sequences. All intervals reported are 95% confidence intervals.

Table 1: Experimental setup

l	d	PROJECTION m	AGGREGATION m	PROJECTION Mean Runtime	AGGREGATION Mean Runtime
9	2	1483	2966	1896 ± 15	1730 ± 50
11	3	2443	4886	3223 ± 18	2800 ± 80
13	4	4178	8356	5550 ± 20	4570 ± 120
14	4	647	1294	704 ± 2	584 ± 14
15	4	172	344	179.9 ± 0.7	157 ± 4
16	5	1292	2584	1683 ± 5	1420 ± 30
18	6	2217	4434	2432 ± 6	1940 ± 40

Comparison of the mean performance coefficients obtained by PROJECTION and AGGREGATION is given in Table 2. AGGREGATION slightly beats out PROJECTION in terms of this metric, though a more meaningful comparison can be seen in Table 5. As the actual motif becomes occluded by equally well preserved spurious motifs in the harder problem instances, we compare the number of times AGGREGATION and PROJECTION successfully find a motif, either correct or spurious, that occurs in all 20 sequences, as done by Keich and Pevzner in [KP02a].

Table 2: Mean performance coefficient comparison

l	d	PROJECTION Performance	AGGREGATION Performance
9	2	0.08 ± 0.03	0.16 ± 0.05
11	3	0.04 ± 0.02	0.08 ± 0.03
13	4	0.05 ± 0.03	0.05 ± 0.02
14	4	0.74 ± 0.04	0.78 ± 0.03
15	4	0.934 ± 0.015	0.93 ± 0.02
16	5	0.59 ± 0.08	0.72 ± 0.06
18	6	0.68 ± 0.07	0.70 ± 0.07

Table 3: Proportion of runs in which the recovered consensus had planted instances in all 20 sequences.

l	d	PROJECTION Proportion	AGGREGATION Proportion
9	2	0.47 ± 0.05	0.71 ± 0.04
11	3	0.80 ± 0.04	0.86 ± 0.03
13	4	0.83 ± 0.03	0.89 ± 0.03
14	4	0.90 ± 0.03	0.970 ± 0.017
15	4	1	1
16	5	0.68 ± 0.05	0.85 ± 0.04
18	6	0.74 ± 0.04	0.79 ± 0.04

As can be seen in Table 4, for all problem instances AGGREGATION discovers the consensus at at least twice the rate of PROJECTION. Higher discovery rates imply that AGGREGATION is able to determine a solution in less time than PROJECTION and is therefore more sensitive to the presence of projected motif buckets.

We can also compare the quality of the AGGREGATION thresholding process by comparing the number of refinements which yield the consensus to the total number of refinements. As can be seen from Table 5, the primary performance gain from AGGREGATION is due to the significant reduction in

Table 4: Mean solution speed comparison

l	d	PROJECTION Successes per 10^3 Seconds	AGGREGATION Successes per 10^3 Seconds	Ratio of Rates AGGREGATION : PROJECTION
9	2	43 ± 18	200 ± 60	5 ± 2
11	3	18 ± 5	51 ± 15	2.9 ± 1.2
13	4	3.5 ± 1.4	10 ± 6	3 ± 2
14	4	10 ± 2	26 ± 5	2.7 ± 0.8
15	4	162 ± 13	400 ± 40	2.5 ± 0.3
16	5	2.4 ± 0.8	6 ± 2	2.5 ± 1.2
18	6	0.9 ± 0.2	2.2 ± 0.6	2.4 ± 0.9

unnecessary refinements. The ratios correspond directly to those in Table 4, as the bulk of the computation time is spent performing refinement.

Table 5: Solutions per 10^6 refinements comparison

l	d	PROJECTION Rate	AGGREGATION Rate	Ratio of Rates AGGREGATION : PROJECTION
9	2	500 ± 200	2500 ± 800	5 ± 3
11	3	220 ± 70	640 ± 190	2.9 ± 1.2
13	4	46 ± 19	130 ± 70	2.9 ± 1.9
14	4	100 ± 20	280 ± 60	2.7 ± 0.8
15	4	1640 ± 130	4200 ± 600	2.6 ± 0.4
16	5	31 ± 10	80 ± 30	2.6 ± 1.3
18	6	10 ± 3	25 ± 6	2.5 ± 0.9

Our experiments show with a high level of confidence that our AGGREGATION method significantly outperforms the original version of PROJECTION in the basic form of the challenge problem. We would also like to show that AGGREGATION outperforms PROJECTION in different variations of the challenge problem. These variations include

- Planted motif instances in background sequences that have non-uniform base distributions
- Planted motif instances in background sequences of increased length

- Motif instances that are not planted in all of the background sequences.

The first two variations were dealt with by Buhler and Tompa [BT01] for the original version of PROJECTION. We are currently performing similar experiments that will give a good comparison of AGGREGATION and PROJECTION in these cases. Buhler and Tompa did not explicitly handle the third variation in their paper. Again, we are performing experiments to compare AGGREGATION and PROJECTION in this case. All of these experiments have been started. We will report our results when they have finished.

Unfortunately we have not yet acquired any biological data. Keich and Pevzner [KP02b] tested their MULTIPROFILER algorithm on a good assortment of biological data. We will shortly be in contact with Uri Keich and we hope to run tests on the same set of biological data by the end of the year. It should be noted that AGGREGATION should succeed in the biological tests in which PROJECTION succeeds.

8 Conclusion

We have developed a modification of Buhler and Tompa's PROJECTION algorithm [BT01] for the motif discovery problem. This modification, which we call AGGREGATION, typically runs twice as fast as the original PROJECTION algorithm when solving the challenge problem posed by Pevzner and Sze [PS00]. This increase in speed is obtained without sacrificing accuracy.

The AGGREGATION method may prove even more useful in more difficult variations of the challenge problem, such as when there are more background l-mers or when the background base distribution is non-uniform. We are currently running experiments that will tell us if this is the case. We also intend to apply AGGREGATION to problems arising in real biological data that were not solved by PROJECTION.

References

[BT01] Jeremy Buhler and Martin Tompa. Finding motifs using random projections. In Thomas Lengauer, David Sankoff, Sorin Istrail, Pavel Pevzner, and Michael Waterman, editors, *Proceedings of the Fith International Conference on Computational Biology (RECOMB-01)*, pages 69–76, New York, April 22–25 2001. ACM-Press.

[EP02] E. Eskin and P.A. Pevzner. Finding composite regulatory patterns in dna sequences. *Bioinformatics*, 18 Suppl 1:S354–63, Jul 2002.

[GS01] D. GuhaThakurta and G.D. Stormo. Identifying target sites for cooperatively binding factors. *Bioinformatics*, 17(7):608–21, Jul 2001.

[KP02a] U. Keich and P.A. Pevzner. Finding motifs in the twilight zone. *Bioinformatics*, 18(10):1374–81, Oct 2002.

[KP02b] U. Keich and P.A. Pevzner. Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics*, 18(10):1382–90, Oct 2002.

[LAB$^+$93] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, October 1993.

[LR90] C. E. Lawrence and A. A. Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins*, 7:41–51, 1990.

[PS00] Pavel A. Pevzner and Sing-Hoi Sze. Combinatorial approaches to finding subtle signals in DNA sequences. In Russ Altman, L. Bailey, Timothy, Philip Bourne, Michael Gribskov, Thomas Lengauer, and Ilya N. Shindyalov, editors, *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB-00)*, pages 269–278, Menlo Park, CA, August 16–23 2000. AAAI Press.

[Roc00] E. Rocke. Using suffix trees for gapped motif discovery. In R. Giancarlo and D. Sankoff, editors, *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, number 1848 in Lecture Notes in Computer Science, pages 335–349, Montréal, Canada, 2000. Springer-Verlag, Berlin.

[ST00] Saurabh Sinha and Martin Tompa. A statistical method for finding transcription factor binding sites. In Russ Altman, L. Bailey, Timothy, Philip Bourne, Michael Gribskov, Thomas Lengauer, and Ilya N. Shindyalov, editors, *Proceedings of the 8th International Conference on Intelligent Systems for Molecular (ISMB-00)*, pages 344–354, Menlo Park, CA, August 16–23 2000. AAAI Press.