A Survey of Planted Motif Search Algorithms

Literature Review

Adrian Kwok | CMPT 711
adriank@sfu.ca | Professor Wiese
SFU ID #200136389 | February 23rd, 2011

1. INTRODUCTION

In Bioinformatics, discovering patterns in a set of DNA sequences is an important problem with many implications in research. Certain genes are responsible for the expression of other genes; more specifically, these 'short' A,T,C,G strings are examples of *regulatory motifs* of which proteins known as *transcription factors* bind onto which ultimately influence the transcription of downstream genes[19]. Thus, given a sequence of DNA sequences for which the expression of other genes are present, we wish to discover the motifs in each sequence that are responsible for the expression of these genes. We can define a simplified version of this problem as:

"Given m DNA sequences of length n over the alphabet$\{A, C, G, T\}$, discover an l-character long motif that occurs in each sequence at least once."

This problem by itself is relatively simple to solve, and can be done so "quickly"[19]. However, most regulatory motifs are not exactly the same in every sequence – that is, certain positions in the motif may differ in each instance in a sequence, also known as a mutation. Therefore, the simplified problem becomes:

"Given m DNA sequences of length n over the alphabet $\{A, C, G, T\}$, discover an l-character long motif that is present with at most d differing mutations in each sequence at least once."

This is known as the *planted (l, d)-motif search* (PMS) problem, which is much more difficult to solve. In fact, this problem is closely related to a well-known difficult problem in computing science – the *COMMON-APPROXIMATE-SUBSTRING* (CAS) problem – which is formally defined as follows:

Instance: A set $\mathcal{F} = \{S_1, \ldots, S_m\}$ of strings over an alphabet Σ such that $|S_i| \leq n$, $1 \leq i \leq m$, and positive integers l and d such that $1 \leq l \leq n$, and $1 \leq d \leq l$.
Question: Is there a string $\mathcal{L} \epsilon \Sigma^l$ such that each string $S \epsilon \mathcal{F}$, $\mathcal{L}$ is Hamming Distance $\leq d$ from some length-l substring of S?

Clearly, the PMS problem is a specific case of the CAS problem, where $\Sigma = \{A, C, G, T\}$. Unfortunately, both problems are shown to be NP-Hard[20]. However, certain optimizations can be done with the PMS problem since d is relatively small in practical applications; there are many algorithms which give an approximate solution to this problem (that is, it returns a motif which may or may not be correct) as well as algorithms which solve specific (l, d) instances exactly *relatively efficiently* such that these algorithms can be used in practice. This literature survey intends to provide an overview of the different types of algorithms which provide solutions to the planted (l, d)-motif discovery problem.

2. BROAD OVERVIEW OF LITERATURE

As previously mentioned, the planted (l, d)-motif search problem (PMS) is a difficult problem. However, clearly the difficulty of the problem depends on different values of l and d – at a glance, an (10,1) instance of the problem seems to be much easier to solve than an (15,4) instance. In fact, the (15,4) instance was the first challenging instance proposed by Pevzner and Sze[1], and several other instances – e.g. (14,4), (16,5), and (18,6) [2] – have been shown to be also challenging.

The basis for the definition of these challenging instances relies on the expected number of random patterns which would occur in any particular sequence; these challenging instances are cases where this number is greater than 1[3]. Intuitively, this corresponds to a loss in accuracy where false motifs are frequently considered due to noise. Buhler and Tompa[2] describe that in actuality, there is a defined threshold for d for which it is unlikely that any algorithm can distinguish a planted motif from random noise past this threshold[3]. While algorithms in the literature successfully solve most general instances of the PMS problem, some earlier algorithms such as WINNOWER(2000)[4] fail at challenging instances while others such as PROJECTION(2002) succeed[2].

The majority of algorithms which attempt to solve the PMS problem (and its particularly challenging instances) are approximation algorithms, while a significantly smaller set of algorithms such as PMS1, PMS2, PMSPrune[5], and MITRA attempt to solve the problem *exactly*. A brief description of these two categories and an overview of different algorithms belonging to these categories will now be described.

2.1 Approximation Algorithms

Approximation algorithms return motifs with a likelihood of certainty that it is the 'correct' motif (e.g. correct in the artificial PMS problem implies the planted motif). One of the earliest papers uses expectation maximization (EM)[9], where the expected starting position of an l-long motif is found using an iterative approach which re-estimates and re-evaluates the motif position probabilistically until a threshold has been reached[10][11]. EM laid the foundations for which many other algorithms such as MEME[10], ProfileBranching[7], and PROJECTION[2] are based on.

An early greedy algorithm by Stormo and Hartzell from 1989 is described in [12] which generates an initial score matrix containing each l-length candidate motif in the first sequence, and then greedily goes through subsequent sequences one at a time and updates the score matrix based on the 'best choice' that preserves information content between an entry in the matrix and a specific candidate. Stormo builds upon this idea to form CONSENSUS[13] with Hertz[7] ten years later, which is shown to be efficient compared to other algorithms[7], but with less accuracy.

PatternBranching, by Price, Ramabhadran, and Pevzner, is a "shamefully simple" yet effective and efficient algorithm which branches from sample strings to search the motif space[5]. It is formulated specifically to tackle the challenging instances proposed by Pevzner and Sze[5], and in that regard it is extremely effective, taking 3 seconds with a 99.7% success rate to solve the initial challenging problem from [1]. Although the original algorithm is not able to solve some challenging instances such as (15,5), a small improvement to the algorithm, coined ePatternBranching by Davila and Rajasekaran[16], allows the algorithm to handle much more difficult algorithms at the expense of generality. Similarly, an improved voting algorithm, with projection, proposed by Leung and Chin[3] is able to solve the relatively long (40,15)-motif problem with over a 95% success rate.

2.2 Exact Algorithms

Also known as combinatorial algorithms, the amount of literature on exact algorithms with an emphasis on challenge problems is relatively little compared to that on heuristic-based algorithms most likely due to its impracticality as the problem is NP-Hard; the predominant algorithms are PMS1, PMS2, and its variants, by Rajasekaran et. al. [5][6], and MITRA, by Pevzner and Eskin[8]. PMS1 and PMS2[6] are relatively simple algorithms, which exhaustively generate all possible candidate motifs from each sequence and prunes them to return a consensus motif – these algorithms are also known as pattern-based algorithms[7]. PMS2 is known to solve challenging instances such as (15,4) but requires nearly 1GB of memory[5]. PMSi, PMSP, and PMSprune[5] build upon these algorithms to achieve better time complexity and space usage using techniques such as branch and bound. MITRA[8] is another pattern-based algorithm and uses mismatch trees to prune the search space as an improvement to the WINNOWER algorithm; it solves challenging instances such as (15,4) quicker than PMS2 and requiring only one tenth of memory[6][8].

3. KEY PAPERS

Finding Composite Regulatory Patterns in DNA Sequences – Eskin and Pevzner (2002)

An efficient exact algorithm for its time is proposed in Eskin and Pevzner's 2002 paper[8], named MITRA (**MI**smatch **TR**ee **A**lgorithm). MITRA follows a pattern-based approach to solving the problem, sharing similarities with Rajasekaran's PMS1/2 paper[6] in analyzing the (l, d)-neighbourhood of all 4^l possible l-mers, deemed the "Sample Driven Approach" (SDA) initially described by Waterman et. al in 1984[5]. However, MITRA does one notable optimization: it realizes that going through the (l, d)-neighbourhood of **all** l-mers is not optimal, since there are many l-mers that are just not frequent enough and should not be considered. It discards these uninteresting l-mers through the use of a Mismatch Tree, a data structure which is similar to a suffix tree except that the children of a node all contain suffixes of its parent with up to k mismatches. It uses this Mismatch Tree in a branch and bound approach to split the space of all l-mers into subspaces based on the number of mismatches, only going through paths in the tree that it deems to be "not weak", thus saving a lot of memory as it does not require the storage of all possible candidate l-mers.

According to the paper's results, MITRA is able to solve all of Pevzner and Sze's[4] challenge problems with ease, a task that many other prior algorithms has failed. Furthermore, MITRA is able to solve other instances that the authors deem to be more difficult than the challenge problems. Unfortunately, the authors do not rigorously define an upper bound for MITRA and its graph-based variant, so it is uncertain how well it may perform for other, non-preconceived instances, especially since solving the exact PMS problem is at best exponential.

Exact Algorithms for Planted Motif Challenge Problems – Rajasekaran, Balla, Huang (2005)

In their paper, Rajasekaran et. al. introduce two exact algorithms to solve the planted motif problem: PMS1 and PMS2[6]. Their first algorithm, PMS1, is relatively simple; it goes through each single input sequence S_i and lists all l-mers from S_i into a corresponding set C_i. Then, for each l-mer fromC_i, it generates the set of all possible l-length patterns with a hamming distance d away from the$l$-mer and stores it into L_i. Each L_i is then sorted in linear time using an integer sorting algorithm and merged together. The motif that occurs in all L_i is the planted motif.

Essentially, the algorithm exploits the idea that *one* l-mer in *each* sequence **must** be d distance away from the planted motif. Thus, it goes through all candidate motifs for each sequence (the (l, d)-neighbourhood of each l-mer) and outputs the same candidate motif that shows up across all sequences, which ultimately corresponds to the planted motif.

The authors claim a bound of $O\left(tn\binom{1}{d}|\Sigma|^d \frac{1}{w}\right)$, where t is the number of input sequences, and w is the word length of the computer. This bound is understandable based on the algorithm presented, but the authors then quickly state that a secondary bound of $O\left(tn + n\binom{1}{d}^2|\Sigma|^d \frac{1}{w}\right)$ is "achievable". It's not immediately clear how this is the case, as this secondary bound is stricter for problems when d is small.

The authors then present a second algorithm, PMS2, which improves on PMS1 based on the following two observations:

1.) Since the target motif occurs in each input sequence, all substrings must also occur in each input sequence. This corresponds to $l - k + 1$ different k-mers for each input sequence which corresponds to the target motif.
2.) These k-mers, in sequence, can be used to obtain the target l-mer.

Like PMS1, two very complex bounds for PMS2 is shown as a theorem. No explanation or proof for the bound is given, and it is very unclear how one can arrive at such a conclusion.

The paper provides new insight as to how to efficiently explore the motif search space to find exact solutions to the PMS problem. Again, however, attempting to solve this problem exactly is likely not that practical for all input as the problem is NP-Hard. The authors show that PMS1 and PMS2 are much faster than MITRA for small d (e.g. ≤ 4); these results correlate with the independent implementation results shown in [14]. However, unlike MITRA which was shown as being able to solve cases even when $d = 9$, albeit slowly, Rajasekaran's algorithms are **unable** to compute instances with higher $d > 4$ due to the algorithms' high memory requirements. This is quite a big deterrent for these algorithms since the set of applicable instances is so restricted. Furthermore, even for applicable instances, computing challenge instances such as (15,4) requires orders of magnitude more time over other heuristic based approaches such as PatternBranching, which gives excellent results in practice.

Finding Motifs Using Random Projections – Buhler, Tompa (2002)

Buhler and Tompa introduce a fresh approach to solving the motif finding problem; their algorithm, PROJECTION, utilizes randomization to find motifs that have a high likelihood of being the planted motif. It was quite effective for its time, besting GibbsDNA, WINNOWER, and SP-STAR for several challenging PMS instances.

In PROJECTION, the user first chooses a value k, where $k < l - d$ ideally, and 4^k hash buckets are created, representing all possible k-mers. m independent trials are then performed; in each trial, the algorithm chooses k random numbers from the set $\{1, \dots, l\}$. The algorithm then goes through each l-mer in every input sequence, and each l-mer is projected (hashed) into one of the 4^k buckets by using the k random numbers (chosen at the beginning of the trial) as positions in the l-mer; an l-mer is hashed into the bucket represented by the concatenation of the k positions in the l-mer – for example, if we have the 5-mer ACGTA and positions {1, 3, 5} are randomly chosen initially, then $\underline{A}C\underline{G}T\underline{A}$ would hash to the AGA bucket.

Thus, some of the 4^k buckets will have more l-mers than others; the algorithm uses a set threshold s to figure out which buckets are 'interesting' – the l-mers in these interesting buckets are then refined to find the planted motif using Expectation Maximization[9].

According to the paper's results, PROJECTION performs similarly, if not better, than GibbsDNA, WINNOWER, and SP-STAR for cases where l is between 10 and 19 and d is between 2 and 6 – more importantly, it is able to solve the (14,4), (17,5), (19,6) difficult problems exceptionally better than these algorithms. Unfortunately, while the algorithm's effectiveness is shown in the results, its efficiency is not, as computing times are omitted altogether. Furthermore, as mentioned by its authors, PROJECTION has problems with instances outside of these (l, d) ranges; later improvements by other authors[3][25] attempt to fix these problems.

<u>Finding subtle motifs by branching from sample strings – Price, Ramabhadran, Pevzner (2003)</u>

Pevzner, Price, and Ramabhadran introduce a simple pattern-based branching algorithm, PatternBranching alongside an equally simple profile-based branching algorithm, ProfileBranching in their seminal 2003 paper[7]. Although said to be "shamefully simple" by its authors, it's one of the fastest and most effective solutions to the PMS problem – Davila and Rajesekaran[16] describe PatternBranching as "one of the fastest non-exact algorithms and […] runs in a couple of seconds with an experimental success rate of near 100%".

Instead of exhaustively analyzing all l-mers in each sequence, the PatternBranching algorithm looks at the whole sample sequence as one concatenated input, and focuses only on d l-mers that are deemed interesting. Initially, it creates an arbitrary candidate motif and chooses the first l-mer in the concatenated sequence as the 'start' of the branch. Then, for d iterations, it chooses the next l-mer to look at based on a BestNeighbour() function – this function chooses an l-mer that differs from the current l-mer by exactly 1 position and minimizes the total distances between the chosen l-mer and the overall sequence. If at each iteration the current l-mer is a better choice than the current candidate motif – that is, it minimizes the total distance as discussed previously – the current candidate motif is updated as the current l-mer.

The ProfileBranching algorithm uses the same approach as the PatternBranching algorithm, but with four notable changes listed by the authors:

1.) Each sample string is converted instead to a profile for the string using a method similar to that used in MEME's [17].
2.) Change the scoring method to score profiles based on an entropy score.
3.) Modify the branching method to account for profiles by modifying BestNeighbor's criteria
4.) Use the top-scoring profile as a seed in the EM algorithm, running until convergence.

Both PatternBranching and ProfileBranching, as discussed before, are extremely effective algorithms, even for the (15,4) challenge problem. PatternBranching is shown to be significantly faster than other algorithms such as PROJECTION, MITRA, and MULTIPROFILER while maintaining close to 100% accuracy. However, the authors' comparison method seems haphazard, as the other algorithms – aside from MULTIPROFILER – are run on significantly slower machines. On more challenging problems, defined as instances with 'dim' planted motifs such as the (15,4) instance with sequences of length 2000 (instead of 600 originally), PatternBranching has over a 99% success rate while PROJECTION falls to 80%, taking only a "few minutes" compared to MULTIPROFILER which takes over an hour. Again, however, the tests for PatternBranching are run on a 1.0GHz machine versus 500Mhz for MULTIPROFILER, which skews the results somewhat.

ProfileBranching is shown to be roughly twice slower than CONSENSUS and GibbsDNA on the original challenge problem, but with a much greater degree of 'success' based on a performance coefficient of the resultant profile; PatternBranching, is still roughly 27 times faster than ProfileBranching. However, on instances deemed unsolvable for pattern-based algorithms – that is, instances where motifs differ in always the same two positions – ProfileBranching has a performance coefficient of 0.99 versus 0.1 for PatternBranching and 0.63 for MEME.

PatternBranching has been shown to fail to solve more difficult problems[16], but a simple improvement to the algorithm to account for these problems is provided by Davila and Rajasekaran.

4. Future Trends

Since the planted (l, d)-motif search problem is NP-Hard, and reflected through the results in most algorithms that try to solve the PMS problem exactly, it seems that this approach is less than ideal for most applications; however, research for specific hardware designed to find exact solutions to this problem has recently been proposed[18], which may be a worthwhile alternative method to consider. On the other hand, heuristic-based approaches are able to solve the problem significantly faster(even more so as the size of the input increases), and with an extremely high degree of accuracy – 99.7% on average as in the case with pattern-based heuristics[7] – which for most practical applications should be sufficient. Of course, these heuristics are likely to be quickly outdated by even more efficient and effective algorithms – the majority of the algorithms covered in this review were published in the past ten years, with major leaps in efficiency occurring every few years; it is only expected that more research with emphasis on heuristics would be around the corner.

5. References

1. Pevzner, P.A. and Sze, S., “Combinatorial approaches to finding subtle signals in DNA sequences”. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pp. 269-278.
2. Buhler, J. and Tompa, M., “Finding motifs using random projections”. *Journal of Computational Biology (2002),* Volume 9, Issue 2, pp. 225-242.
3. Leung, H. and Chin, F., “Algorithms for challenging motif problems*”. Journal of Bioinformatics and Computational Biology (2006),* Volume 4, Issue 1, pp. 43-58.
4. Keich, U. and Pevzner, P.A., “Subtle motifs: defining the limits of motif finding algorithms”. *Bioinformatics (2002),* Volume 18, Issue 10, pp. 1382-1390.
5. Davila, J., Balla, S., and Rajasekaran, S., "Fast and Practical Algorithms for Planted (l, d) Motif Search”. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (2007)*, October-December, pp. 544-552.
6. Rajasekaran, S., Balla, S., and Huang, C.-H., “Exact Algorithms for Planted Motif Problems*”. Journal of Computational Biology (2005)*, Volume 12, Issue 8, pp. 1117-1128.
7. Price, A., Ramabhadran, S., and Pevzner, P.A., “Finding subtle motifs by branching from sample strings”. *Bioinformatics (2003),* Volume 19, Issue2, pp. 149-155.
8. Eskin, E. and Pezner, P.A., “Finding composite regulatory patterns in DNA sequences”. *Bioinformatics (2002)*, Volume 18, Issue 1, pp. 354-363.
9. Lawrence, C.E. and Reilly, A.A., “An expectation maximization(EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences”. *Proteins: Structure, Function, and Bioinformatics (1990)*, Volume 7, Issue 1, pp. 41-51.
10. MEME - Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization
11. Do, C.B. and Batzoglou, S., “What is the expectation maximization algorithm?”. *Nature Biotechnology (2008)*. Volume 26, No. 8, pp. 897-899.
12. Stormo, G.D. and Hartzell, G.W., “Identifying protein-binding sites from unaligned DNA fragments”. In *Proceedings of the National Academy of Sciences of the United States of America (1989)*, Volume 86, Issue 4, pp. 1183-1187.
13. Hertz, G.Z. and Stormo, G.D., “Identifying DNA and protein patterns with statistically significant alignments of multiple sequences”. *Bioinformatics (1999)*, Volume 15, Issue 7, pp. 563-577.

14. Locke, C., “Implementation of Planted Motif Search Algorithms PMS1 and PMS2”, *BioGrid Research Experience for Undergraduates*, Summer 2008. University of Connecticut, CT.
15. Waterman, M.S., Arratia, R., and Galas, D.J., “Pattern recognition in several sequences: Consensus and alignment”. *Bulletin of Mathematical Biology (1984)*, Volume 46, Number 4, pp. 515-527.
16. Davila, J., “Extending Pattern Branching to Handle Challenging Instances”, In *Proceedings of the Sixth IEEE Symposium on Bioinformatics and BioEngineering(BIBE ’06)*. IEEE Computer Society, Washington, DC, USA, pp. 65-69.
17. Bailey, T.L. and Elkan, C., “The value of prior knowledge in discovering motifs with MEME”. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology (1995)*, pp. 21-29.
18. Kent, K.B., Schaick, S.V., Rice, J.E., and Evans, P.A., “Hardware-Based Implementation of the Common Approximate Substring Algorithm”, In *8th Euromicro Conference on Digital System Design (2005),* pp. 314-321.
19. Jones, N. and Pevzner, P., “An Introduction to Bioinformatics Algorithms,” MIT Press, Cambridge, MA, 2004.
20. Evans, P.A.,Smith, A.D., Wareham, H.T., “On the complexity of finding common approximate substrings”. *Theoretical Computer Science (2003)*, Volume 306, Issues 1-3, pp. 407-430.
21. Leung, H.C.M and Chin, F.Y., “Algorithms for challenging motif problems”. *Journal of Bioinformatics and Computational Biology (2006)*, Volume 4, Issue 1, pp. 43-58.
22. Leung, H.C.M, Chin, F.Y.L., Yiu, S.M., Rosenfield, R., Tsang, W.W., “Finding Motifs with Insufficient Number of Strong Binding Sites”. *Journal of Computational Biology (2005)*, Volume 12, Issue 6, pp. 686-701.
23. Pisanti, N., Carvalho, A.M., Marsan, L., Sagot, M.-F., “RISOTTO: Fast extraction of motifs with mismatches”. In *Proceedings of the 7th Latin American Theoretical Informatics Symposium*, Volume 3887, pp. 757-768.