High Performance Computational Tools for Motif Discovery[*]

Nicole E. Baldwin[†], Rebecca L. Collins[†], Michael A. Langston[†], Michael R. Leuze[‡§], Chris T. Symons[†] and Brynn H. Voy[‡]

Abstract

The research described in this paper highlights a fruitful interplay between biology and computation. The sequencing of complete genomes from multiple organisms has revealed that most differences in organism complexity are due to elements of gene regulation that reside in the non protein coding portions of genes. Both within and between species, transcription factor binding sites and the proteins that recognize them govern the activity of cellular pathways that mediate adaptive responses and survival. Experimental identification of these regulatory elements is by nature a slow process. The availability of complete genomic sequences, however, opens the door for computational methods to predict binding sites and expedite our understanding of gene regulation at a genomic level. Just as with traditional experimental approaches, the computational identification of the molecular factors that control a gene's expression level has been problematic. As a case in point, the identification of putative motifs, which is the subject of this paper, is a challenging combinatorial task. For it, powerful new motif finding algorithms and high performance implementations are described. Heavy use is made of graph algorithms, some of which are exceedingly computationally intensive and involve the use of emergent mathematical methods. An approach to fully dynamic load balancing is developed in order to make effective use of highly parallel platforms.

Key Words and Phrases

Computational genomics, elucidation of *cis*-regulatory elements, parallel algorithms for molecular sequence analysis, protein binding and mRNA transcription

[*] This research has been supported in part by the National Science Foundation under grants EIA-9972889 and CCR-0075792, by the Office of Naval Research under grant N00014-01-1-0608, by the Department of Energy under contract DE-AC05-00OR22725, and by the Tennessee Center for Information Technology Research under award E01-0178-081.

[†] Department of Computer Science, University of Tennessee, Knoxville, TN 37996-3450, USA

[‡] Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37831-6124, USA

[§] Communicating author: leuzemr@ornl.gov

1 Introduction

1.1 Computational identification of *cis*-regulatory elements and modules

Coordinate regulation of gene expression provides the basis for activation and repression of specific cellular pathways and the adaptive response to changing cellular conditions. Although the coding regions of each gene involved in any given pathway are now/will soon be known as mouse and human genome sequence are annotated, the molecular mechanisms that mediate shared regulation and determine which subsets of genes interact in specific pathways are poorly understood. These molecular mechanisms, consisting of transcription factor (TF) proteins and their DNA binding sites (TFBSs), form the architecture of gene regulatory networks, which in turn underlie basic physiology and adaptive responses within an organism. While traditional approaches to identifying factors that control regulation of a single gene have been largely experimental, the availability of complete genomic sequence for human and mouse opens the door for computational approaches to predict regulatory sequence with a throughput that surpasses the scope of experimental techniques.

Eukaryotic gene transcriptional regulation is now appreciated to be organized in 2 basic levels. The first level consists of individual TFBSs, short motifs (8-15 bp) in non protein-coding DNA to which TFs bind in *cis*. A single TF protein binds not to an exact DNA motif but rather to a consensus motif in which the precise identity of some bases is critical while other bases within the string can vary. Specific collections of TFBSs are organized into modules that mediate a gene's reponse to a specific factor, representing the second level of eukaryotic regulatory organization (reviewed in Halfon and Michelson [15] and Bolouri and Davidson [8]). For example, transcriptional control of the Endo16 gene during sea urchin development is mediated through 56 individual TFBSs that cluster into 7 regulatory modules within 2.2 kb of DNA sequence upstream of the gene's transcriptional start site [36]. Genes that have shared expression patterns under a specific condition—for example, genes up-or down-regulated in pancreatic beta cells in response to glucose exposure—are widely assumed to share at least some TFBSs or modules that, coupled with the corresponding TFs, mediate a common gene expression response. This "guilt-by-association" concept is increasingly used as a starting point from which to identify a subset of genes that can be mined for the presence of shared regulatory mechanisms [28].

1.2 Review of Computational Approaches

Motif-finding. The body of literature on motif discovery is extensive. We are aware of over 300 papers directly related to computational methods designed to identify transcription factor binding sites in DNA sequences. The difficulty of the problem has led to the development of numerous approaches for analyzing non-coding regions of eukaryotic genomes with the goal of finding those elements that regulate gene expression.

For TFs for which the consensus binding site is known, additional sites in collections of genes can be discovered by searching for hits against a position weight matrix (PWM) constructed from the statistical likelihood of base representation at each position of the binding site. However, robust matrix models require extensive experimental data from which to construct them, and at present only a small subset of TFs in multicellular eukaryotes are represented in databases such as ConSite [34]. Therefore ab initio approaches to TFBS discovery hold the most promise for rapid discovery of motifs, particularly given the ongoing annotation and availability of eukaryotic genomes. Most approaches to motif discovery search for patterns that are overrepresented in the regulatory regions of orthologous or coregulated genes. Search strategies fall into two major categories, those that use local heuristics, such as Gibbs sampling [21] or expectation maximization [5, 22], and those that use globally optimal methods, such as enumeration of all motifs in the search space [29, 32, 33]. Globally optimal methods are exhaustive and deterministic, but, due to computational complexity, tend to be limited to detection of short patterns. Heuristic approaches may find longer patterns, but may fail to converge to a globally optimal solution. For a recent review of motif discovery methods, particularly heuristic approaches, see Wasserman and Krivan [34].

Some methods use a process known as "phylogenetic footprinting," in which orthologous sequences are compared to first identify highly conserved regions of noncoding DNA, under the premise that selection pressure for conservation is due to the presence of functional elements within. Searching for motifs only within conserved regions is an effective way to limit the search area to the most meaningful regions, thus significantly reducing the numbers of false positive motifs that are identified. In fact, Krivan and Wasserman [20] recently reported that regulatory elements were ~ 320 times more likely to occur in conversed vs. nonconserved regions, based on computational and experimental findings from a set of liver-specific genes. These methods may compare two or more orthologs, may take phylogenetic relationships into consideration [7], and may be based on global or local alignment. Clustal W [31] and DIALIGN [26] are methods for multiple sequence alignment that are used to locate local genomic regions of similarity.

Widely used motif discovery algorithms based on Gibbs sampling include AlignAce [18], ANN-Spec [35], BioProspector [24], Co-Bind [14], and MDscan [25]. Among other heuristic approaches are MEME [4], which uses expectation maximization, and CONSENSUS [16, 17], which uses a greedy algorithm. Hybrid algorithms combining concepts from sample-driven heuristics and pattern-driven enumeration are beginning to emerge; among these hybrid algorithms is MULTIPROFILER [19].

Module-finding. Despite the increasing recognition that modules, rather than individual TFBSs, mediate transcriptional regulation in eukaryotes, computational approaches to predict modules lag far behind motif discovery efforts. The simplest approach has been to search for pairs of TFBSs that either overlap or lie in close proximity to each other, for which the spacing between sites is relatively invariable. This method has proven successful for specific combinations of TFBSs that are known to work in concert, such as Mef2 and MyoD [11], and it may also be useful for recognizing pairs of sites that bind TFs that function as heterodimers, as do many nuclear hormone receptors. However, simply searching for pairs of sites oversimplifies the typical biological situation and is unlikely to define more than a very small fraction of regulatory modules. Approaches that are not restricted to either only pairs of sites or to strict rules about spacing between TFBSs are much more likely to accurately model the molecular reality of transcriptional regulation. Recent success has been reported for techniques that search for modules by cluster analysis of defined TFBS [6, 13, 15], and methods such as this benefit from the lack of restrictions on numbers and positions of TFBSs within a module.

1.3 Lipogenic genes

Due to their agricultural importance in several species, the genes involved in lipogenesis represent an extremely well-characterized set of genes in terms of their expression levels and the TFBSs that mediate their individual regulation. Synthesis and storage of lipids from carbohydrate precursors allows an organism to store excess energy for later periods in which calories may be scarce. This lipogenic response is innately sensitive to both the presence of excess carbohydrates, such as glucose, and to hormones such as insulin that signal an organism that energy is abundant. Lipogenesis is activated by coordinate regulation of genes encoding enzymes involved in the pathway of lipid synthesis from glucose precursors. An abundance of prior studies demonstrate that the majority of this activation is due to changes in the genes' transcriptional activities, and many important TFBSs have been identified for genes in the lipogenic pathway (reviewed in Foufelle and Ferre [12]). Consistent with the guilt-by-association concept, several TFBSs are found in multiple lipogenic genes, suggesting that common regulatory mechanisms underlie at least a subset of shared regulation. Therefore, due to the wealth of experimentally-confirmed binding site data that exists for these genes and the detailed knowledge about their regulation during changing nutritional conditions, we selected this class of genes to which to apply our system for first identifying potential TFBSs and then collating these sites into regulatory modules.

2 The Motif Discovery Toolkit

The suite of novel computational tools that we have developed for the Motif Discovery Toolkit [23] constitute a new approach for identifying functional DNA in non-coding regions by sequence comparison. Our approach differs from other motif-finding methods in a number of important ways, which collectively make it unique:

- it is applicable to any collection of related genomic sequences,
- it does not depend on the global alignment of these sequences,
- it does not require a motif to be present in each sequence of the collection, and it is able to discover multiple copies of a motif in a single sequence,
- it can find multiple motifs in a single run, and
- it is a general computational method; it does not require specific biological information about the sequences being examined.

Our motif discovery toolkit contains tools for a large number of tasks, including filtering sequences, finding short inexact matches, combining short matches, constructing graphs to represent relationships between motifs, identifying interesting graph structures, producing position weight matrix and profile hidden Markov models of motifs, scoring sequences relative to a motif model, and searching for inexact inverted and direct palindromes.

A significant feature of our Motif Discovery Toolkit is its extensive use of graph algorithms. Few previous approaches to motif discovery have made explicit use of graph algorithms. An exception is WINNOWER [27], which transforms the search for motifs to the task of finding large cliques in multipartite graphs but is limited to problems of relatively small size. The integration into the Motif Discovery Toolkit of a number of innovative graph algorithms, some of which execute three orders of magnitude faster than competitors, will allow consideration of problems much larger than any previously addressed. Additionally, although the Motif Discovery Toolkit uses global search methods, it examines only those patterns actually found in the input sequences, rather than all 4^n possible patterns for an n-base sequence, thus greatly improving efficiency. We are also developing novel statistical approaches to quantify the relationship between two sequences, incorporating biological information into a distance metric. As a result, the graphs we examine are generally smaller than would be the case if an edge between two patterns were weighted only by the number of matches. Our metric tends to connect only those patterns for which there is a biologically significant relationship. Collectively, these new ideas make the Toolkit well suited to the task of motif discovery.

3 Use of the Toolkit

A typical search for shared motifs using the Motif Discovery Toolkit proceeds through the following steps:

Step 0: ***Sequence selection.*** Input consists of an arbitrary set of nucleotide sequences that may contain common motifs, such as the upstream regions from orthologs, co-expressed, or co-regulated genes.

Step 1: ***Filtering.*** Filtering of input sequences with RepeatMasker [30] or additional tools that more aggressively remove low-complexity DNA. Filtering may also be done through identification of regions conserved between pairs of orthologous sequences.

Step 2: ***Extracting (ℓ,m) subsequences.*** (ℓ,m) subsequences are pairs of subsequences of length ℓ for which m or more of the nucleotide bases are identical. All (ℓ,m) subsequences between every pair of input sequences and within an individual input sequence are found.

Step 3: ***Combining (ℓ,m) subsequences.*** Overlapping and adjacent (ℓ,m) subsequences are merged into the longest possible pairs of subsequences. Non-adjacent (ℓ,m) subsequences that are separated by a small gap of fixed or variable size may also be combined. Step 3 results in the set of maximal subsequences, an important intermediate data structure.

Step 4: ***Graph construction.*** For a specified motif length or pattern, a graph is constructed. Edges that meet specified criteria are extracted from the set of maximal subsequences. Nodes of the graph correspond to subsequences from the input sequences.

Step 5: ***Clustering.*** Clusters of motifs are identified using information contained in the graph constructed in step 4. Clusters may be chosen and refined in a number of ways. A cluster may correspond to a clique, a connected component, or the nodes located in a highly-connected region of the graph. A variety of graph algorithms may be applied to identify highly-connected regions. Options include *k*-connected components, dense *k*-subgraph, cut sets, separators, neighborhood search, treewidth and many others. Central to solving most of these are optimization variants of the *NP*-hard clique problem. Maximum clique, maximal cliques, near-clique, bi-clique and clique intersection graphs are just a few examples. Clusters may also be formed by choosing all nodes within a specified distance (a *k*-neighborhood) of a starting node, which may correspond to a known transcription factor binding site or to a significant palindromic sequence.

Step 6: ***Develop motif models.*** Models of the clusters chosen in step 5 are developed. Models include position weight matrix models and profile hidden Markov models. The model may be used to refine the cluster, by trimming low scoring motifs, or to quickly screen a larger collection of genomic sequences, including an organism's entire genome.

4 High Performance Computations

4.1 Clique, Vertex Cover and Fixed-Parameter Tractability

A clique in an undirected graph is a subset of vertices each pair of which is connected by an edge. Formally, the decision problem usually asked is whether an arbitrary graph *G* of order *n* contains a clique of size $k<n$ or more. Clique is extraordinarily difficult. Not only is it *NP*-complete, but it cannot even be approximated to within a constant factor (assuming $P \neq NP$). One might hope therefore that fixing the value of *k* would help, via the theory of fixed-parameter tractability[**] (FPT). But clique is not FPT (unless the *W* hierarchy collapses). Fortunately, the *NP*-complete vertex cover is a complementary dual to clique. That is, *G* has a clique of size at least $n-k$ if and only if the complement of *G* has a vertex cover of size at most *k*. Vertex cover *is* FPT. We shall give only a brief sketch of the practical significance of this in the sequel. We refer the reader to [10] for background, rationale and structural foundations on FPT as an effective algorithm design paradigm.

4.2 Kernelization and Branching

FPT algorithms generally proceed in two main stages. The first stage is termed "kernelization." Here the goal is to reduce a problem of size *n* to its computational kernel, whose size depends only on *k*. We have reported extensively on our work on kernelization elsewhere. See, for example, [1].

[**] A problem is fixed-parameter tractable if it has an algorithm that runs in $O(f(k)n^c)$ time, where *n* is the problem size, *k* is the input parameter, and *c* is a constant.

The second stage is called “branching.” Here the goal is to explore the kernel as efficiently as possible. Given the complexity of the underlying problem, this step is exhaustive in nature. Some form of high performance computing is generally required even on problem instances of only modest size. Although subproblems can be solved independently, their relative computational needs are difficult to estimate in advance. This frequently causes a very uneven processor loading, thereby eliminating the opportunity for attractive speedup. This has been shown to be especially troublesome in the case of “no” instances (those without small enough covers). For details, timings and a primitive form of load balancing, see [2].

There are many other pragmatic issues, including input preprocessing, parametric tuning and the interleaving of kernelization and branching. As we scale up to problems of larger size, however, it has become clear that parallel load balancing during branching remains a major stumbling block. Much more work is needed on this subject, and we turn to it now.

4.3 Parallel Load Balancing

Branching uses a tree to structure the exploration of the kernel’s search space, thus breaking the vertex cover problem into disjoint subtrees. This makes the problem very well suited for parallel programming. Due to the unpredictable nature of the search, however, these subtrees are often unbalanced with respect to solution density. Simply dividing the problem and distributing the subtrees among processors has the potential to make the computation effectively sequential if all but a handful of processors complete the search of their respective subtrees early. We have already incorporated a very simple form of load balancing designed to handle the extreme case in which all but one processor finish while the vast majority of the computation remains to be done on the single remaining active processor. This strategy suffices in a surprising number of cases. When it works, computation times can be reduced by days when using 32 or 64 processors to search for large cliques in protein domain data [2]. Nevertheless, the shortcomings of this approach are manifest. It is of no help when there are more than one, but still only a few, difficult subtrees. Even when there is only one, there is a troubling amount of overhead because there is no job queue. Instead, the subtree must be sent back to be re-split, and all work that has already been done on it is lost.

We are now developing a dynamic parallel branching procedure to balance the workload more consistently and without wasting cycles. A central scheduler maintains a job queue and assigns new jobs to processors as they become available. When a processor is handling an excessively large subtree and room appears in the job queue, that processor prunes off a branch of its subtree and sends it to the scheduler to be reassigned as soon as another processor becomes available. The notion of subtree size is relative, of course, and can be tailored to a given problem, network configuration and processor architecture. In this manner, subtrees need be searched at most once, and all processors can be kept busy with useful tasks.

The major differences between the former and the current dynamic branching implementations are based around communications and queuing.

Communication. In our initial approach, the vertex cover driver consists mainly of a process splitter, which executes secure shell (ssh) commands to initialize a branching process on each processor. Processors then operate in complete isolation from one another and from the driver. When a processor finishes searching its subtree, relevant information is stored in an output file. The driver continuously monitors the files to see what has been written, and will perform one of two operations. If all but one of the processors have finished their respective jobs, but no satisfying cover has been found, then the driver halts the task on the only processor still working and uses the splitter to divide and redistribute subtrees across all processors. On the other hand, if a cover has been found, then the driver halts all tasks and returns the cover to the user. See Figure 1.

In our new approach, communication is handled in a client/server-style environment. Here the driver contains both a splitter and a scheduler to open a socket before executing ssh to initialize branching at the processors. Each parallel process then requests a connection with the scheduler, and uses this connection to communicate job status, indicate availability and so forth. This client/server design eliminates the need for polling the file system, because a processor can simply notify the scheduler when it is free. Likewise, the driver can signal all of the clients directly if a solution is found so that they can terminate quickly and cleanly. The scheduler follows the threaded server design with one thread per client. Note that this design can also be used when only one processor is available, in which case it will likely outperform sequential codes in current use in single-processor environments due to uneven subtree sizes. There are additional benefits to direct communication. The overhead of file I/O is eliminated. Moreover, with this design we should be able to incorporate software management tools such as Netsolve [3], thereby making our codes more easily available to others. See Figure 2.

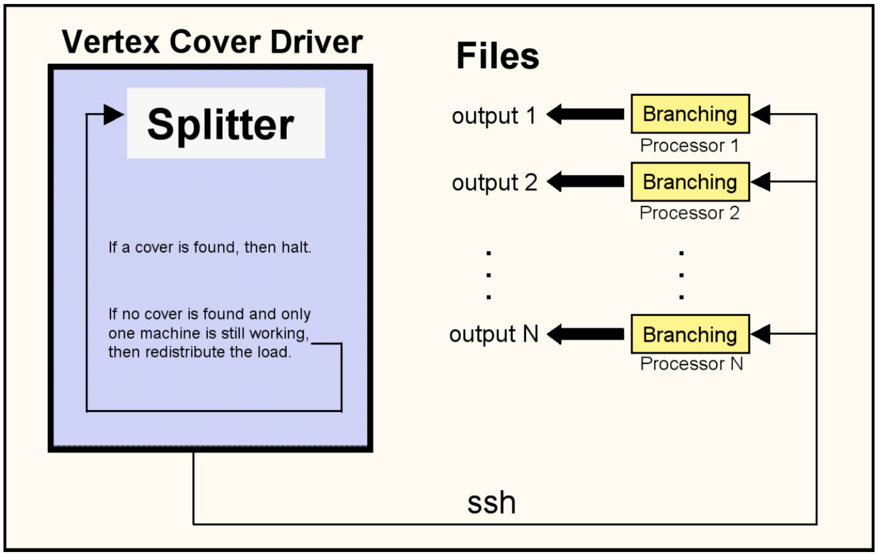

Figure 1: The Original Load Balancing Architecture

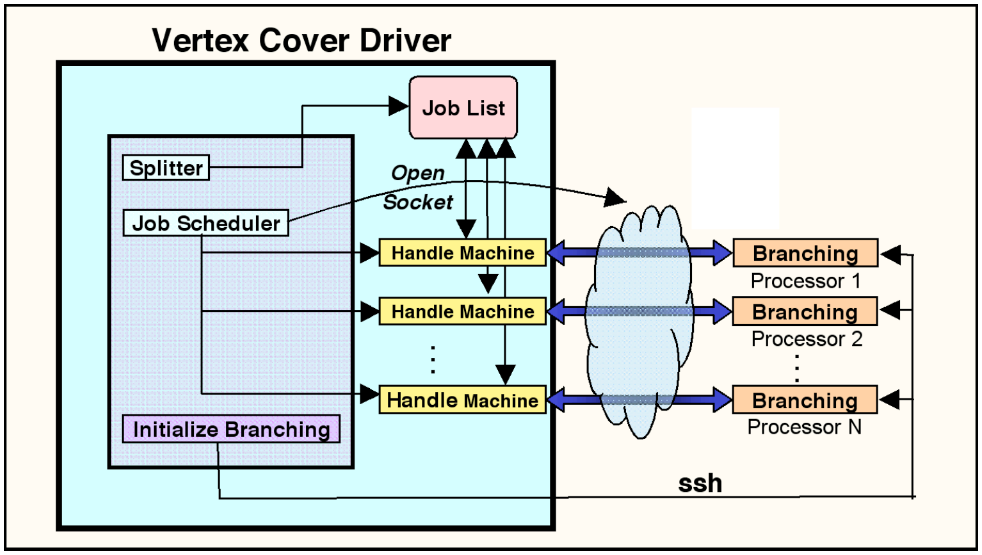

Figure 2: The New Load Balancing Architecture

Queuing. Because it is not known in advance which subtrees will take the longest to run to completion, load balancing mechanisms must be designed with dynamic pruning and reallocation in mind. The chief advantage of our original approach was its simplicity. But without a job queue, even when it works to reduce the overall runtime its disadvantages in terms overhead can be significant. This drawback is illustrated graphically in Figure 3.

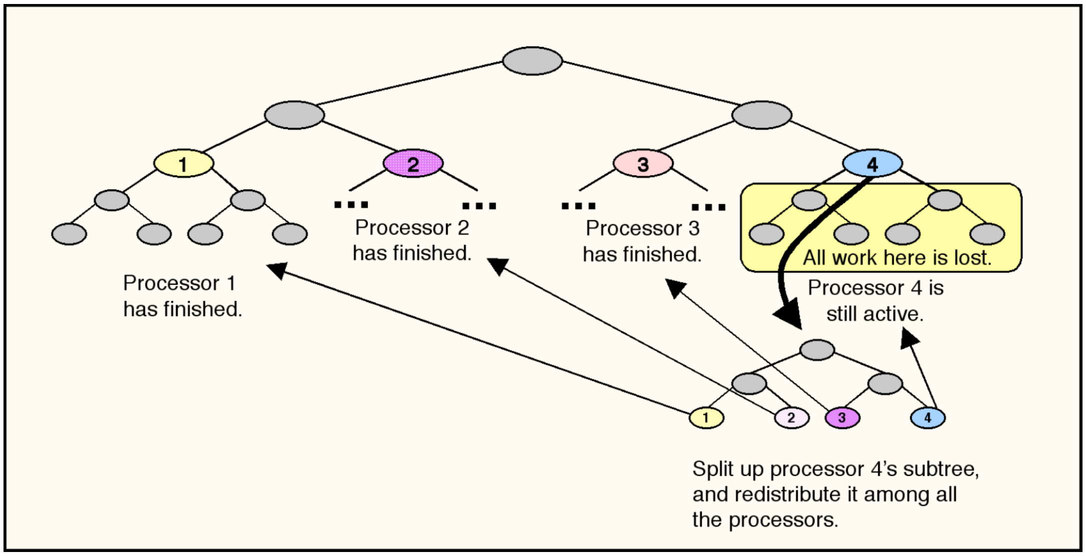

Figure 3: The Original Subtree Splitting Mechanism

Our new approach avoids redundant computation by pruning off a new subtree at its parent's current computational location. Only the new subtree is sent back to the scheduler for placement in the job queue. No previously-searched part of the original subtree is examined again. The decision to prune must balance the short-term pruning overhead with its long-term computational benefit. Subtree size and difficulty can only be approximated, and overly frequent pruning must be avoided. We are currently experimenting with a number of pruning factors, including subtree size estimates, elapsed time since the last pruning operation was performed, search tree depth, processor availability and other variables. We believe it will be a challenging but rewarding task to determine a near-optimal pruning strategy as we ramp up to bigger and bigger instances of the clique problem and many of its variants.

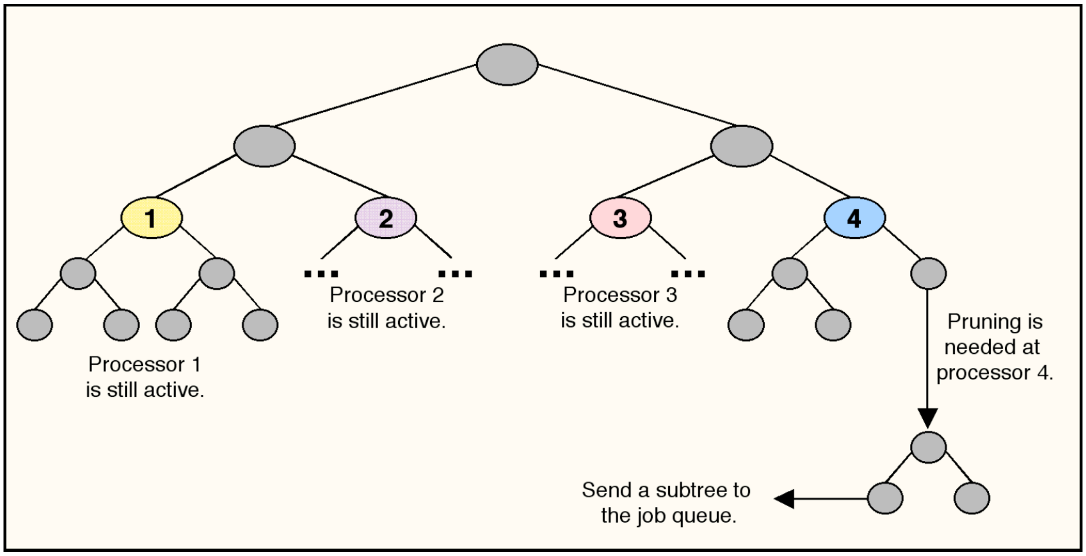

Figure 4: The New Subtree Splitting Mechanism

5. Application to the Lipogenic Gene System

We applied the Motif Discovery Toolkit to a collection of 12 lipogenic mouse genes and their human orthologs. For each of the orthologous pairs, we extracted the genomic sequence consisting of the entire upstream region,[††] the transcribed region,[‡‡] and 5K bp downstream. An initial filtering, designed to retain only conserved regions, was performed by comparison between orthologous mouse and human upstream regions. We first found all (*50*,*25*) matches between the corresponding mouse and human sequences, and then produced the set of maximal subsequences by merging matches that were adjacent, overlapping, or separated by a fixed gap of size 30 or smaller. These parameters were chosen with the goal of discovering relatively large regions with at least 50% similarity.

The highest scoring subsequence within each maximal subsequence was identified using a scoring function based on subsequence length, number of matches, and nucleotide base frequencies. Subsequences with a score of 20.0 or greater were considered to be conserved and were retained.[§§]

Two observations are relevant at this point: 1) It is not necessary to identify all conserved regions of the sequences. If we begin with sufficiently many sequences and identify a sufficiently large percentage of the conserved regions, it is probable that the motif corresponding to the binding sites of a TF will be present often enough to be discovered. Additional putative binding sites for this TF may then be found using other tools. 2) High scoring subsequences may be useful in aligning two sequences. We have explored the use of a greedy algorithm that examines the subsequences in order from highest to lowest score, accepting new subsequences into the alignment only if they are consistent with the set of subsequences already accepted. The resulting alignments compare very favorably with those produced by Avid [9]. However, in identifying conserved regions, we do not depend on global alignment of sequences, in order to bypass the possibility that small genomic arrangements (insertions or deletions) over time might have disrupted the sequential organization of these regions.

The collection of mouse sequences, filtered to retain only conserved regions, was then analyzed. (*15*,*7*) sequences between and within strings were found and merged into maximal subsequences using a fixed-length gap of 5 or smaller. Edges were extracted from the maximal subsequences, and the resulting graphs were searched for interesting structures. Motif length and minimum edge weight parameters were adjusted to produce graphs in which the largest connected components were not so large as to preclude examination by hand. The application of state-of-the-art graph algorithms to the problem of identifying biologically relevant structures in the motif graphs will enable us to efficiently search the parameter space, resulting in significant improvements to the detection of subtle motifs.

6. Results

Eight motif clusters were used to create position specific log-odds weight matrix models, and each model was used to search the entire collection of mouse sequences filtered to remove only exons. Some of these clusters correspond to binding sites for transcription factors known to the TransFac database, such as AP-1 (activator protein 1), Gfi-1 (growth factor independence1), Oct-1 (octamer factor 1), and C/EBPbeta (CCAAT / enhancer binding protein beta). Other clusters appear to correspond to novel TF binding sites.

[††] The "upstream region" of a gene is that region from the beginning of transcription up to the next known gene or pseudo gene. In those instances where this distance is significantly that 100K bp, the upstream region is limited to 100K bp.

[‡‡] When there are multiple transcripts of a gene, the "transcribed region" is that region from the 5' end of the exon farthest upstream in any transcript to the 3' end of the exon farthest downstream in any transcript.

[§§] A score of 20.0 is comparable in probability to 34 consecutive matches between two sequences or 70% similarity between two sequences of length 100, assuming, in both cases, equal distribution of all four bases.

Some motifs from the search can be grouped into modules, such as those upstream from Lep and Thrsp, as illustrated in Figure 5. We expect to improve greatly our ability to detect motif modules through the use of co-occurrence matrices, bipartite graphs, and fast graph algorithms.

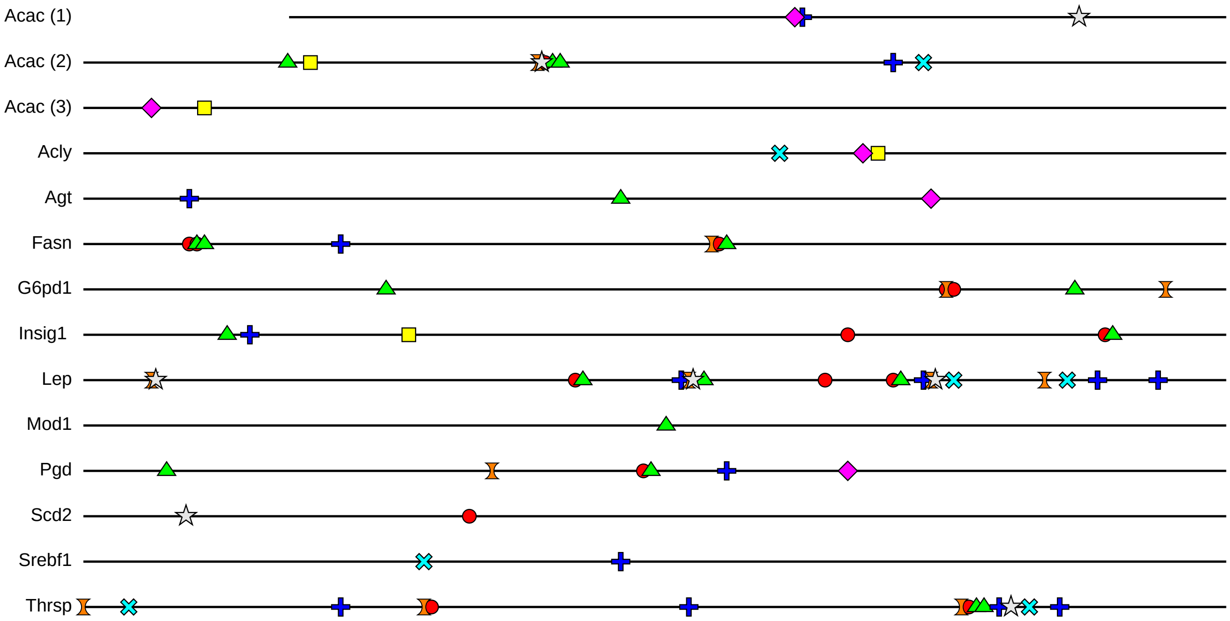

Figure 5: Putative regulatory modules. The lines correspond to 150 bp segments immediately upstream from the start of transcription in the lipogenic genes. Note that three promoters for Acac are known. The symbols represent putative motifs found using weight matrix models. Observe the similar motif groupings upstream from Lep and Thrsp.

References

[1] F. N. Abu-Khzam, R. L. Collins, M. A. Langston, W. H. Suters and C. T. Symons, *Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments*, *Proceedings, Workshop on Algorithm Engineering and Experiments (ALENEX)*, New Orleans, LA, 2004.

[2] F. N. Abu-Khzam, M. A. Langston and P. Shanbhag, *Scalable Parallel Algorithms for Difficult Combinatorial Problems: A Case Study in Optimization*, *Proceedings, International Conference on Parallel and Distributed Computing and Systems (PDCS)*, Los Angeles, CA, 2003.

[3] D. Arnold, W. Lee, J. Dongarra and M. Wheeler, *Providing Infrastructure and Interface to High Performance Applications in a Distributed Setting*, *Proceedings, ASTC High Performance Computing Symposium*, Washington, DC, 2001.

[4] T. L. Bailey and C. Elkan, *Fitting a mixture model by expectation maximization to discover motifs in biopolymers*, Proc Int Conf Intell Syst Mol Biol, 2 (1994), pp. 28-36.

[5] T. L. Bailey and C. Elkan, *Unsupervised learning of multiple motifs in biopolymers using expectation maximization*, Machine Learning, 21 (1995), pp. 51-80.

[6] B. P. Berman, Y. Nibu, B. D. Pfeiffer, P. Tomancak, S. E. Celniker, M. Levine, G. M. Rubin and M. B. Eisen, *Exploiting transcription factor binding site clustering to identify cis-regulatory modules involved in pattern formation in the Drosophila genome*, Proc Natl Acad Sci U S A, 99 (2002), pp. 757-62.

[7] M. Blanchette, B. Schwikowski and M. Tompa, *An exact algorithm to identify motifs in orthologous sequences from multiple species*, Proc Int Conf Intell Syst Mol Biol, 8 (2000), pp. 37-45.

[8] H. Bolouri and E. H. Davidson, *Modeling transcriptional regulatory networks*, Bioessays, 24 (2002), pp. 1118-29.

[9] N. Bray, I. Dubchak and L. Pachter, *AVID: A global alignment program*, Genome Res, 13 (2003), pp. 97-102.

[10] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, 1999.

[11] J. W. Fickett, *Coordinate positioning of MEF2 and myogenin binding sites*, Gene, 172 (1996), pp. GC19-32.

[12] F. Foufelle and P. Ferre, *New perspectives in the regulation of hepatic glycolytic and lipogenic genes by insulin and glucose: a role for the transcription factor sterol regulatory element binding protein-1c*, Biochem J, 366 (2002), pp. 377-91.

[13] M. C. Frith, J. L. Spouge, U. Hansen and Z. Weng, *Statistical significance of clusters of motifs represented by position specific scoring matrices in nucleotide sequences*, Nucleic Acids Res, 30 (2002), pp. 3214-24.

[14] D. GuhaThakurta and G. D. Stormo, *Identifying target sites for cooperatively binding factors*, Bioinformatics, 17 (2001), pp. 608-21.

[15] M. S. Halfon and A. M. Michelson, *Exploring genetic regulatory networks in metazoan development: methods and models*, Physiol Genomics, 10 (2002), pp. 131-43.

[16] G. Z. Hertz, G. W. Hartzell, 3rd and G. D. Stormo, *Identification of consensus patterns in unaligned DNA sequences known to be functionally related*, Comput Appl Biosci, 6 (1990), pp. 81-92.

[17] G. Z. Hertz and G. D. Stormo, *Identifying DNA and protein patterns with statistically significant alignments of multiple sequences*, Bioinformatics, 15 (1999), pp. 563-77.

[18] J. D. Hughes, P. W. Estep, S. Tavazoie and G. M. Church, *Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae*, J Mol Biol, 296 (2000), pp. 1205-14.

[19] U. Keich and P. A. Pevzner, *Finding motifs in the twilight zone*, Bioinformatics, 18 (2002), pp. 1374-81.

[20] W. Krivan and W. W. Wasserman, *A predictive model for regulatory sequences directing liver-specific transcription*, Genome Res, 11 (2001), pp. 1559-66.

[21] 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 (1993), pp. 208-14.

[22] 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 (1990), pp. 41-51.

[23] M. R. Leuze and B. H. Jones, *A constructive approach to discovery of motif clusters in genomic sequences*, Technical Report, Oak Ridge National Laboratory, 2003.

[24] X. Liu, D. L. Brutlag and J. S. Liu, *BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes*, Pac Symp Biocomput (2001), pp. 127-38.

[25] X. S. Liu, D. L. Brutlag and J. S. Liu, *An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments*, Nat Biotechnol, 20 (2002), pp. 835-9.

[26] B. Morgenstern, K. Frech, A. Dress and T. Werner, *DIALIGN: finding local similarities by multiple sequence alignment*, Bioinformatics, 14 (1998), pp. 290-4.

[27] P. A. Pevzner and S. H. Sze, *Combinatorial approaches to finding subtle signals in DNA sequences*, Proc Int Conf Intell Syst Mol Biol, 8 (2000), pp. 269-78.

[28] J. Quackenbush, *Genomics. Microarrays--guilt by association*, Science, 302 (2003), pp. 240-1.

[29] S. Sinha and M. Tompa, *A statistical method for finding transcription factor binding sites*, Proc Int Conf Intell Syst Mol Biol, 8 (2000), pp. 344-54.

[30] A. F. A. Smit and P. Green, *RepeatMasker*, see http://ftp.genome.washington.edu/RM/RepeatMasker.html.

[31] J. D. Thompson, D. G. Higgins and T. J. Gibson, *CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice*, Nucleic Acids Res, 22 (1994), pp. 4673-80.

[32] M. Tompa, *An exact method for finding short motifs in sequences, with application to the ribosome binding site problem*, Proc Int Conf Intell Syst Mol Biol (1999), pp. 262-71.

[33] J. van Helden, B. Andre and J. Collado-Vides, *Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies*, J Mol Biol, 281 (1998), pp. 827-42.

[34] W. W. Wasserman and W. Krivan, *In silico identification of metazoan transcriptional regulatory regions*, Naturwissenschaften, 90 (2003), pp. 156-66.

[35] C. T. Workman and G. D. Stormo, *ANN-Spec: a method for discovering transcription factor binding sites with improved specificity*, Pac Symp Biocomput (2000), pp. 467-78.

[36] C. H. Yuh and E. H. Davidson, *Modular cis-regulatory organization of Endo16, a gut-specific gene of the sea urchin embryo*, Development, 122 (1996), pp. 1069-82.