Motif finding using distributed systems

Literature review

CSE 527
15 December 2004

Marty Lyons
Overview

• Finding strings in sets of otherwise unaligned sequences is considered an NP-hard problem

• Gibbs sampling does not improve its result after each iteration as does EM

• Greedy search algorithms can get trapped in regions of local maxima

• Weight matrix models such as Maximum Likelihood and Expectation Maximization (MEME) can be used in a distributed environment
Parallel approach

• Motif finding is complex
  – distributing work in parallel a useful approach

• Limited number of research papers cover this activity

• Review of work performed at San Diego, UT/Oak Ridge National Lab, Ohio State, Indiana University
Systems issues

• Parallel and distributed systems have unique issues
• Interprocess communication (typically handled with Remote Procedure Call or Message Passing Interface)
• process/thread synchronization
• shared access to common resources (memory/disk)
• queuing, load balancing, and latency
ParaMEME: Grundy et. al.

- Web-accessible service
- Method: MEME (Multiple EM for Motif Elicitation)
- Does not assume any prior knowledge of motif instances within a dataset
- Considers a motif as a "block" containing patterns that repeat in a sequence
- Can locate motifs unevenly distributed throughout the dataset; automatically calculate their length
Method

• Searches by building statistical models of found motifs
• Then a matrix of (discrete) probabilities
• Optimization to locate the model with maximal posterior probability
Algorithm

(1) procedure MEME (X : data set of sequences)
(2) for pass = 1 to num motifs
(3)     for width = w min to w max
(4)         for starting point = 1 to num starting points
(5)           Estimate score of model with this initial model
(6)     end
(7)     Choose initial model with maximum score
(8)     for lambda = lambda min to lambda max
(9)         Run expectation-maximization to convergence
(10)    end
(11)    Choose model with maximal $G(.)$
(12)    end
(13)    Choose and report model with maximal $G(.)$
(14)    Erase the model from the prior probabilities
(15)    end
MEME steps

• First: scan for possible initial models
  – build score which is estimated log-likelihood after one EM iteration

• Then: EM is performed until convergence
  – beginning with the initial model containing the greatest score (prevents search from getting stuck in local max)

• Parameter G(.) specifies best model (Lambda) and to select motif width

Grundy
Three possible models

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOPS</td>
<td>One occurrence per sequence -- assume each motif appears exactly once in each sequence</td>
</tr>
<tr>
<td>ZOOPS</td>
<td>Zero or one occurrence per sequence -- motif may be absent from some sequences</td>
</tr>
<tr>
<td>TCM</td>
<td>Two-component mixture -- motif may occur any number of times in each sequence</td>
</tr>
</tbody>
</table>
Parallelism

• Inherent parallelism of two of the four loops
  – moving to successive motifs incurs delay while removed from further consideration

• Single Program Multiple Data (SPMD): the whole dataset is available to every processor working on the problem.

• At end of main loop - maximization reduction
  – propagate the best scoring model to all processors
Environment

- Run on San Diego Supercomputer Center's Intel Paragon XP/S 400 node cluster
- 8000 lines of ANSI C; MP++ message-passing library
- Load balancing: processors select initial models using an interleave approach
  - System ensures equal number of characters are sent to each processor, even if the number of sequences may vary
Efficiency v. processors

- Balanced
- Not balanced

Efficiency % vs. # of processors
Experimental results

Grundy
Feedback

- Most complete treatment of the topic of using parallel systems for motif finding
- Additional information on parallelization of MEME would have been useful
  - granular timing data, particularly inter-nodal communication
- Clearly a paper written by computer systems people, with detail oriented towards that group
- Very little information in this paper on the actual motif finding results
### High performance computational tools for motif discovery (Baldwin et. al.)

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sequence selection: Input an arbitrary set of nucleotide sequences</td>
</tr>
<tr>
<td>1</td>
<td>Filtering: Filtering of input sequences with tools that more aggressively remove low-complexity DNA; or identification of regions conserved between pairs of orthologous sequences.</td>
</tr>
<tr>
<td>2</td>
<td>Extracting ((l,m)) subsequences. ((l,m)) are pairs of subsequences of length (l) for which (m) or more of the nucleotide bases are identical</td>
</tr>
<tr>
<td>3</td>
<td>Combining ((l,m)) subsequences. Overlapping / adjacent ((l,m)) subsequences are merged into the longest possible pairs of subsequences.</td>
</tr>
<tr>
<td>4</td>
<td>Graph construction for a specified motif length or pattern</td>
</tr>
<tr>
<td>5</td>
<td>Clustering. Clusters of motifs are identified using information contained in the graph; selection and refinement possible</td>
</tr>
<tr>
<td>6</td>
<td>Develop motif models; including WMM and HM models</td>
</tr>
</tbody>
</table>
Branching

- Computation is able to be balanced - search space now broken into disjoint subtrees
- Developing dynamic parallel branch procedure to overcome the overhead from lack of queuing
- Socket mechanism - processors signal scheduler when idle, Vertex Cover Driver can also issue signals to clients
Load balancing
Experimental results

- Eight motif clusters in trial - models searched all known mouse sequences excluding exons
- Of clusters defined, some correlated to known transcription factor binding sites (per TransFac database)
- Some motifs grouped into modules within 150 bp upstream of start of transcription of lipogenic genes
Feedback

- Explanation of clique and branching in section four falls short
- Results not clearly defined
- No operational measurements or descriptions available
Parallel algorithms for mining frequent structural motifs in scientific data (Wang and Parthasarathy)

- Algorithm first broken into two separate components
  - Intra-Structure model: goal to locate all motifs which occur frequently in a single molecule
  - Inter-Structure model: goal to identify motifs which are frequent across all molecules
Level-wise refinement

• Intra-structure p-MotifMiner is a level-wise algorithm.

• At iteration $k$, generate $C_k$, the set of candidate $k$-atomsets from the frequent atomsets in the previous iteration ($L_{k-1}$) and then prune infrequent atomsets from $C_k$ until we get $L_k$, the set of frequent $k$-atomsets.
Parallelism

1. parallel candidate generation algorithm
   - allows the partitioning of the iteration space
2. parallel recursive fuzzy hashing (RFH) pruning
   - how two different atomsets may belong to the same motif

• Reduction of search space
  - Dynamic Duplicate Screening (DDS) removes identical neighboring trees with equivalent atomsets
Experimental results

- MPI, 16 node cluster, 1Ghz Pentium III, 256 KB secondary cache, 512 MB RAM, Linux kernel 2.4.18

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>Intra-molecule frequent substructures analysis query on protein lysozyme (PDB ID: 193L)</td>
</tr>
<tr>
<td>Query 2</td>
<td>Intra-molecule frequent substructures analysis query on protein hemoglobin (PDB ID: 1BZ0)</td>
</tr>
<tr>
<td>Query 3</td>
<td>Intra-molecule frequent substructures analysis query for the phenylalanine tRNA of yeast (PDB ID: 1EVV) that finds a novel substructure in the codon regions</td>
</tr>
</tbody>
</table>
Experimental results

- Speedup during atomset / pruning up to 16 processors
- Efficiency falls as processors increase:
  - 8 proc: 0.81
  - 16 proc: 0.66
  - attributed to potential workload imbalance
- Biotonic partitioning yields improved results in pruning correlating to the number of processors in use
- Self-adaptive expansion affects pruning time
Experimental result

- Fuzziness has a large bearing on overall execution time
- $=0$: exact match between atomsets
  - time mainly spent in candidate selection/generation
- $=2$: pruning becomes primary concern

- Scalability test at Ohio Supercomputer Center
  - moderate scalability as cluster size increases
  - overall efficiency decreases
Efficiency

Speedup for Query 1 and 2 on IA32 cluster
Feedback

• Most "general" paper of those surveyed
  – Designed a system for multiple domains
  – Biological motif finding is just one use

• Comparison implementing MotifMiner in a non-parallel environment would have been useful as a control
Guiding motif discovery by iterative pattern refinement (Z. Wang et. al.)

- Use subsequence selection to aid in motif discovery
  - Extract subsequences likely to contain motifs
  - Iterate on resampling until there is convergence

- Framework tested using PROSITE patterns
  - overall performance compared with MEME
# Method

**STEP** | **Process**
---|---
1 | Extract a set $S$ of subsequences around a set of motifs $M$
2 | Input $S$ to a motif discovery algorithm, producing a new set of motifs $M'$
3 | Search entire sequences for more occurrences of $M'$, producing $M''$. Set $M''$ to $M$ and go to step 1
Seed generation

- Initial seed set built by collecting a set of statistically significant patterns
- During construction of initial patterns, fixed length $L$ at 3, such that they are also present in a number of sequences
- Seed motifs constructed using some of the most frequently observed patterns
  - some should be part of the actual motif during protein analysis
Seed motif (protein)

Z. Wang
Refinement / Iteration

- Iterative refinement is carried out in three steps
- Continues as long as more motif instances are found
Experimental results

- Testing: Pentium IV (dual) processor 1.7 GHZ system with 4 GB of RAM, Red Hat Linux 8.0.
  - Parallel version - MEME - IBM SP cluster
- 108 PROSITE patterns used as test data
  - Test framework located 23 additional patterns undiscovered by MEME.
  - MEME located 7 not found by framework
  - Both MEME and framework did not find 15

- Failure rate of framework: 20%
- Failure rate of MEME: 35%
Feedback

- Focus on subsequence use is clear
- Operational results presented poorly
- No comparison offered to other approaches other than that of MEME
MotifMiner: A general toolkit for efficiently identifying common substructures in molecules (Coatney and Parthasarathy)

- MotifMiner is the same research effort as in Wang, this paper was written a year prior
- Written in Java - designed as a flexible and extensible framework
- Has three components:
  - Converters
  - Abstract Data Model
  - Core Toolkit
Converters

- Converters allow molecular structures to be defined using an abstract data model
  - allows atoms and Euclidean coordinates to be specified along with "mining bonds"
  - specifying atomic type and distance between them
Core services

• A set of core services for substructure discovery and comparison
  – range pruning service; discretization and fuzzy noise services

• Motif services are defined
  – frequency pruning
  – duplication removal of motifs
  – fingerprinting
Experimental results

- Verification of a Si lattice with two defects detected successfully
- Pharmacology test: classified compounds with 95% success compared to a C4.5 classifier.
Feedback

• An interesting explanation of the architectural ideals behind the MotifMiner concept
  – but little else
• Mainly an "idea" paper
• Lack of useful experimental findings
• Later work [21] builds on this foundation
## Summary

<table>
<thead>
<tr>
<th>Author</th>
<th>Title</th>
<th>Technique</th>
<th>Environment</th>
<th>Scalability</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grundy [10]</td>
<td>ParaMEME</td>
<td>MEME</td>
<td>Intel Paragon 400 node</td>
<td>72% efficient at 64 processors</td>
<td>90% (balanced)</td>
</tr>
<tr>
<td>Wang [21]</td>
<td>Parallel algo. for mining/Motif Miner (2004)</td>
<td>Range pruning, candidate pruning, recursive fuzzy hash, distance binning + resolution</td>
<td>16 nodes with MPI, 1 GHz P3, 256 KB sec. cache, 512 MB RAM, Linux 2.4.18</td>
<td>Good to 16 processors</td>
<td>8 processors: 0.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16 processors: 0.66</td>
</tr>
</tbody>
</table>
Summary

<table>
<thead>
<tr>
<th>Author</th>
<th>Title</th>
<th>Technique</th>
<th>Environment</th>
<th>Scalability</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16 proc: 0.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32 proc: 0.69</td>
</tr>
<tr>
<td>Z. Wang [24]</td>
<td>Guiding motif / iterative</td>
<td>MEME (motif discovery) + MAST (motif searching) -- iterate</td>
<td>Pentium IV dual proc 1.7 GHz 4 GB RAM, Red Hat Linux 8.0; MEME run on IBM SP cluster (details unspecified)</td>
<td>Unspecified</td>
<td>Found same 64 PROSITE patterns as MEME, found addtl 23 patterns MEME did not, missed 7 patterns MEME found. Method and MEME both missed 15 pat. Framework failure: 20%, MEME failure: 35%</td>
</tr>
<tr>
<td>Author</td>
<td>Title</td>
<td>Technique</td>
<td>Environment</td>
<td>Scalability</td>
<td>Efficiency</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>Li [18]</td>
<td>Free parallel mining</td>
<td>Bounded optimal split resolution trees</td>
<td>Sun Sparc 5 workstations, 5-45 nodes in increments of 5</td>
<td>Near linear to 45 nodes</td>
<td>Unspecified</td>
</tr>
</tbody>
</table>
References


References


References


References


References


References


References
