Mining Frequent Motif Combinations in Regulatory Regions of the DNA

Project report

Markko Merzin
2009

Introduction.
Many proteins bind on the regulatory regions of the DNA only if certain binding sites (motifs) are physically close. As the DNA molecule is in packed form in cells, the physical proximity does not always imply proximity on the DNA sequence. However, in practice, we can still seek motif combinations that are in certain window in order to discover motif combinations that seem to attach certain transcription factors or their combinations.

Assignment.
Project assignment was to find frequent motif combinations in yeasts regulatory regions given yeast genome in FASTA format, list of motifs and list of regulatory areas on yeast genome. Additional constraint was given: motifs in discovered combinations must be close to each other. Window width 50 was chosen to this project.

Methodology and experiments
Project work is divided to few sequential step:

Finding motif matches.
Motif matches were found using storm program from cread-0.84\(^1\) software package. As yeast genome sequences were given in separate FASTA files by chromosome, storm was runned once per chromosome. Match threshold score was 0.8. Storm output was piped to ad-hoc converter and filter script, motifs2timeline.pl, which converted storm output to format “[motif match starting position] : [motif length] : [motif name]”. As provided regulatory regions are all in positive strand of dna sequence, this script additionally filtered out motif matches that were on negative strand. Output from converter script were piped to sort program and saved to file for next step. Command line for described actions is following:

```
cread-0.84/bin/storm -f -t 0.8 -s $cr transfac-pfms-all.tab | ./motifs2timeline.pl | sort -n > $outname
```

Filtering motifs in regulatory areas.
In this step output from last step is piped trough filter scripts filtermotifs.pl and filtermotifs2.pl.

\(^1\) CREAD: Comprehensive Regulatory Element Analysis and Discovery. http://rulai.cshl.edu/cread
Those scripts read in regulatory region positions from file “oreganno-regulatory-regions-systematic.tab” and output motifs matches in regulatory regions (filtermotifs.pl) or motif matches out of regulatory regions (filtermotifs2.pl). Command line for filtering is:

```
filtermotifs.pl $chrom oreganno-regulatory-regions-systematic.tab < $ms > ${ms}_in_regulatory_areas
filtermotifs2.pl $chrom oreganno-regulatory-regions-systematic.tab < $ms > ${ms}_outof_regulatory_areas
```

filtermotif.pl and filtermotifs2.pl scripts output are in following format:

```
#start location:length:motif name
45034:10:TF-M00031
```

**Episode mining.**

Episode mining is performed by perl script `minepi.pl`. It implements sequential mining algorithm MINEPI\(^2\). Although MINEPI does not assume any fixed windows width, kind of windowing was implemented by limiting look-ahead length because semantic constraints of project. Mining script were run on regulatory areas and out of regulatory areas separately. Limit for episode length was 50, minimal support was 5 and minimal confidence was 0.7. Mining script assumed that all episodes were sequential and did not considered parallel episodes. Mining script outputs found episode rules as:

```
# rule:support:confidence
M1,M2,M3>M4:16:0.8
```

which reads as “after seeing motifs M1, M2 and M3 within given episode length (50 in our case), we see motif M4 within same length with 80% probability. Sequence M1,M2,M3,M4 is observed 15 times and sequence M1,M2,M3 is observed 16/0.8=20 times”.

**Comparing, unifying and aggregating.**

Results from episode mining from regulatory areas and non-regulatory areas were aggregated together by `compare-and-unify.pl` script. This script outputs episode rules as:

```
#rule:support over count of episodes in regulatory areas:confidence in regulatory areas:support over episode count out of regulatory areas:confidence out of regulatory areas
TF-M00204,TF-M00031,TF-M00713>TF-M00664:0.0052:0.39:0.011:0.51
```

As results are still chromosome by chromosome, last step is to aggregate all result together and output only those episode rules witch are significantly more expressed in regulatory areas. That step is done by script `aggregate-and-report.pl`. Given rule is printed out when its relative support in regulatory areas is at least 2 times bigger then relative support in out of regulatory areas.

**Command and queue.**

All described scripts and programs are launched from shell script named `command-and-queue.sh`. It

tries to parallelize workloads and has metainfo for executing in HPC. Turned out that used parallelism approach is inefficient in HPC context and needs to be revised prior to possible re-usage in different projects.

**Results**

All created program artifacts, raw data and computing results are available at address [http://www.ut.ee/~markkom/fc-project](http://www.ut.ee/~markkom/fc-project).

97 unique episode rules were identified. 10 most expressed sequences in regulatory areas are:

- TF-M00303>TF-M00031:0.00958083832335329:0.8:0:0
- TF-M00713,TF-M00303>TF-M00031:0.00755429650613787:1:0:0
- TF-M00732>TF-M00713:0.00718562874251497:0.857142857142857:0:0
- TF-M00303,TF-M00031>TF-M00713:0.00718562874251497:0.75:0:0
- TF-M00061>TF-M00048:0.00718562874251497:0.857142857142857:0:0
- TF-M00337>T-M00713:0.00646868011273402:0.840509270402105:0:0
- TF-M00664,TF-M00728>TF-M00713:0.0056657223796034:0.75:0:0
- TF-M00168>TF-M00713:0.00523560209424084:0.785714285714286:0:0
- TF-M00713,TF-M00308>TF-M00048:0.0049926884036246:0.75:0:0
- TF-M00015,TF-M00197>TF-M00713:0.00473641448929408:0.857142857142857:0:0

Rest of rules are in file: [http://www.ut.ee/~markkom/fc-project/output/final_results](http://www.ut.ee/~markkom/fc-project/output/final_results)

We may assume that those motif combinations may be someway related to gene expression regulatory mechanisms.

**Summary and discussion**

This project work was first glance at working with genome data to the author. MINEPI algorithm is known from its large memory footprint and working with as large dataset as genome demanded very careful programming and optimization to fit into existing hardware. For example, first implementations of minepi quickly exhausted 32GB memory but later ones used 1,3GB per whole chromosome.

MINEPI implementation which was written for this project turned out to be useful also in other areas. It was with some success used for mining frequent sequences from universities internal telephone system.

Also it was first time to the author to use High Performance Cluster to large set of calculations. Turned out that efficient HPC usage implies somewhat different approaches to workload parallelism than working with single, although multi-CPU computer.