

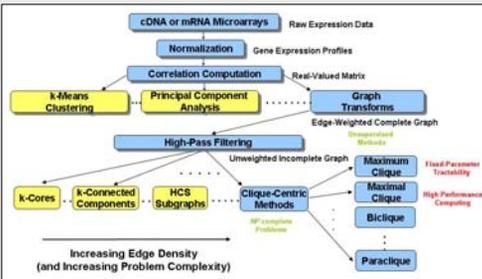
Demands and Solutions for Genome-Scale Combinatorial Analysis

Mike Langston¹, Elissa Chesler², John Eblen¹, Loren Hauser², Bob Hettich², Phil LoCascio², Andy Perkins¹, Arnold Saxton¹, Dave Tabb³, Dorothea Thompson⁴, Nathan VerBerkmoes², Brynn Voy², Roumyana Yordanova¹, Bing Zhang³, Yun Zhang¹

¹University of Tennessee, Knoxville, TN 37996-3450 ²Oak Ridge National Laboratory, Oak Ridge, TN 37831-6445
³Vanderbilt University, Nashville, TN 37232-8340 ⁴Purdue University, West Lafayette, IN 47907-2108

Background

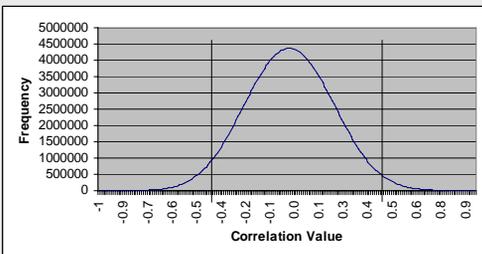
For simplicity, we focus only on microarray technology. In a single slice, microarrays can identify genes that are up- or down-regulated in a condition of interest. It is this analysis of differential expression that today drives the majority of array experiments. Yet it is the next level of analysis – the potential to extract meaningful relationships between multiple genes – that sets the power of arrays apart from traditional measures of gene expression. “Guilt by association,” the assumption that genes with similar expression patterns participate in common cellular functions, drives the emerging attempt to extract pathways from microarray data. We employ a graph theoretical approach to identify sets of co-regulated genes. Many innovative graph algorithms are based on decades of basic research, and constitute a class of tools that can help elucidate relationships hidden in matrices of correlations across thousands of genes. From such a matrix we create an unweighted graph. The challenge is not to study the graph in its entirety, but rather to extract its embedded subgraphs, or small, tightly connected regions of the graph that represent subsets of genes with strong correlations between every pair of its members and are thus likely to represent biologically significant interactions. In the most extreme case, in which a subgraph contains all possible edges between its vertices, this structure is called a *clique*. The importance of clique lies in the fact that each and every pair of vertices is joined by an edge, from which we can infer some form of co-regulation among the corresponding genes.



A overview of clique-based clustering algorithms for microarray analysis. Our approach is shown in blue.

Correlation Structures

To build a graph from a correlation matrix, a threshold is selected above which an edge is retained and below which the edge is removed. We use a variety of thresholding schemes, including the use of functional similarity, statistical relevance and graph structure theory.



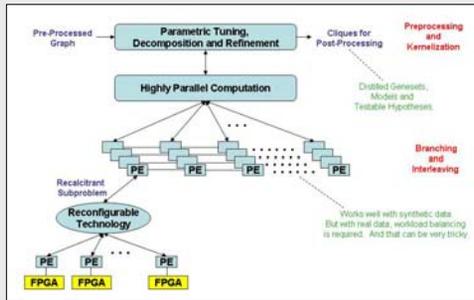
Sample gene-gene correlation coefficient distribution with possible thresholds marked.

Computational Complexity

The inputs to the standard decision version of clique are an undirected graph G with n vertices, and a parameter $k \leq n$. The question asked is whether G contains a clique of size k , that is, a subgraph isomorphic to K_k . Subgraph isomorphism, clique in particular, is NP-complete. From this it follows that there is no known algorithm for deciding clique that runs in time polynomial in the size of the input.

Fixed Parameter Tractability

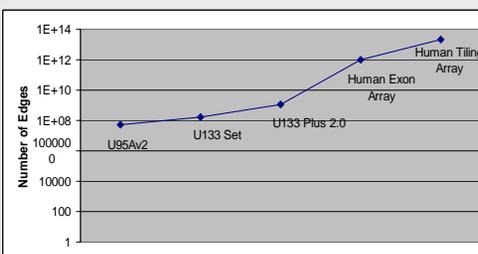
Novel approaches are thus required if clique is to be applied to microarray data sets. In this context we employ *fixed parameter tractability* (FPT): A problem is FPT if it has an algorithm that runs in $O(f(k)n^c)$ time, where n is the problem size, k is the input parameter, and c is a constant independent of both n and k . Clique itself is not FPT (unless the W hierarchy collapses). We therefore focus instead on clique's complementary dual, the *vertex cover* problem. Consider G' , the complement of G . (G' has the same vertex set as G , but edges present in G are absent in G' and vice versa.) The question now asked is whether G' contains a set C of k vertices that covers every edge in G' , where an edge is said to be covered if either or both of its endpoints are in C . Like clique, vertex cover is NP-complete. Unlike clique, however, vertex cover is FPT. The crucial observation here is this: a vertex cover of size k in G' turns out to be exactly the complement of a clique of size $n - k$ in G . Thus, we search for a minimum vertex cover in G' , thereby finding the desired maximum clique in G . Key algorithmic factors in the success of an FPT-based approach are *kernelization*, in which a problem is reduced to its compute kernel, and *branching*, in which a search tree is employed to explore the solution space efficiently. Under some conditions these operations can be iterated, a process termed *interleaving*.



The clique compute engine, highlighting the use of parametric tuning, fixed-parameter tractability, massive parallelism and hardware acceleration.

Demands for Petascale Computing

As more complex and integrated forms of data come on line, we encounter severe computational bottlenecks as we ramp up to genome-size problems. Integrating varying types of data, including that from gene expression, proteomics and SNPs, is a key priority. The scale of correlation across these diverse data types tends to lower correlation values and thus increase edge densities. In addition, data from exon and tiling arrays can produce graphs with millions of vertices and trillions of edges. Enumerating, interpreting and prioritizing maximal cliques incurs extreme memory overhead: a graph of order n may have as many as $3^{n/3}$ maximal cliques. A typical, modest-sized problem required 607 GB of core memory to hold newly generated cliques, and 404 GB to hold other needed cliques, when it was terminated after 12 hours on 256 CPUs of the ORNL SGI Altix, “Ram.”



Typical graph sizes produced by data from microarray and related high throughput biological experiments.

High Performance Implementations

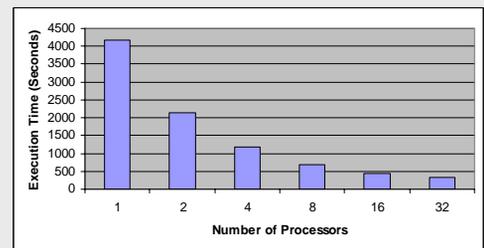
Computing the maximum clique size is a first and foundational computational step. In it we employ a recursive branching algorithm that is not trivially parallelized. A sophisticated dynamic load balancing strategy is necessary, because dividing up the search space in advance generally produces a very unbalanced search tree on real data (often only a few nodes end up with most of the work). A useful strategy is to monitor the system as the computation proceeds, redistributing workloads as needed to keep all processors busy. Each processor stores several possible jobs that can be delegated to other processors as necessary. Frequent communication is imperative. All of this must be accomplished with as little overhead as possible, so that the cost of parallelization does not exceed its benefits.



Supercomputing resources are required. Our codes were executed on an SGI Altix at ORNL, with 256 dual-CPU processors, each clocked at 1.5 GHz, with two terabytes of shared memory.

Illustrative Timings

We ran our codes on *Mus Musculus* spleen gene expression data obtained with Affymetrix M430A arrays. After thresholding by removing correlations not significant at the $p=0.01$ level, a graph with 22750 vertices and 11417976 edges was obtained. Kernelization further reduced the problem to a graph of just 3103 vertices and 1673174 edges. Runs were conducted using 1, 2, 4, 8, 16, and 32 processors on the SGI Altix at ORNL. The maximum clique size was found to be 291. This is of course not a genome-size problem, and is used only to demonstrate the scalability of our methods.



Branching times for a co-regulation graph of order 3101 with 35% edge density.

Remarks

Identifying co-expressed genes, interacting proteins, putative regulatory networks and other important biological features are formidable tasks. The data sets required are enormous and often hard to integrate. The combinatorial problems to be solved are intractable with traditional methods. Novel algorithmic tools, including FPT and others, are helpful for central problems such as clique, which represents the most trusted potential for finding sets of inter-correlated genes and gene products. Nevertheless, computational demands rapidly exhaust current computing capabilities and require petascale resources.