The Maximum Clique Enumeration Problem: Algorithms, Applications and Implementations

John D. Eblen, Charles A. Phillips, Gary L. Rogers, and Michael A. Langston

Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville TN 37996-3450, USA

Abstract. Algorithms are designed, analyzed and implemented for the maximum clique enumeration (MCE) problem, which asks that we identify all maximum cliques in a finite, simple graph. MCE is closely related to two other well-known and widely-studied problems: the maximum clique optimization problem, which asks us to determine the size of a largest clique, and the maximal clique enumeration problem, which asks that we compile a listing of all maximal cliques. Naturally, these three problems are $\mathcal{NP}$-hard, given that they subsume the classic version of the $\mathcal{NP}$-complete clique decision problem.

MCE can be solved in principle with standard enumeration methods due to Bron, Kerbosch, Kose and others. Unfortunately, these techniques are ill-suited to graphs encountered in our applications. We must solve MCE on instances deeply seeded in data mining and computational biology, where high-throughput data capture often creates graphs of extreme size and density. MCE can also be solved in principle using more modern algorithms based in part on vertex cover and the theory of fixed-parameter tractability (FPT). While FPT is an improvement, these algorithms too can fail to scale sufficiently well as the sizes and densities of our datasets grow.

An extensive testbed of benchmark MCE instances is devised, based on applications in transcriptomic data analysis. Empirical testing reveals crucial but latent features of such high-throughput biological data. In turn, it is shown that these features distinguish real data from random data intended to reproduce salient topological features. In particular, with real data there tends to be an unusually high degree of maximum clique overlap. Armed with this knowledge, novel decomposition strategies are tuned to the data and coupled with the best FPT MCE implementations. It is demonstrated that the resultant run times are frequently reduced by several orders of magnitude, and that instances once prohibitively time-consuming to solve are now often brought into the domain of realistic feasibility.

Keywords: maximum clique enumeration, maximal clique, gene expression analysis, software tools and applications.

1 Introduction

Clique is one of the best known and most widely studied combinatorial problems. Although classically formulated as an $\mathcal{NP}$-complete decision problem [17], the

J. Chen, J. Wang, and A. Zelikovsky (Eds.): ISBRA 2011, LNBI 6674, pp. 306–319, 2011.

search and optimization formulations are probably most often encountered in practice. In computational biology, one needs to look no farther than PubMed to gauge clique's utility in a variety of applications. A notable example is the search for putative molecular response networks in high-throughput biological data. Popular clique-centric tools include clique community algorithms for clustering [24] and paraclique-based methods for QTL analysis and noise abatement [9,10].

A clique is *maximal* if it cannot be augmented by adding additional vertices. A clique is *maximum* if it is of largest size. A maximum clique is particularly useful in our work on graphs derived from biological datasets. It provides a dense core that can be extended to produce plausible biological networks [13]. Other biological applications include the thresholding of normalized microarray data [6,25], searching for common *cis*-regulatory elements [3], and solving the compatibility problem in phylogeny [16]. See [5] for a survey of additional applications of maximum clique.

Any algorithm that relies on maximum clique, however, has the potential for inconsistency. This is because graphs often have more than just one clique of largest size. Thus it is that algorithmic idiosyncrasies, not scientific reason, are apt to lead to an arbitrary choice of cliques. This motivates us to find an efficient mechanism to enumerate all maximum cliques in a graph. These can then be examined using a variety of relevant criteria, such as the average weight of correlations driven by strain or stimulus [2].

We therefore seek to solve the *Maximum Clique Enumeration (MCE)* problem. Unlike maximal clique enumeration, for which a substantial body of literature exists, very little seems to be known about MCE. The only exception we have found is a game-theoretic approach for locating a predetermined number of largest cliques [8]. We begin by creating a testbed of graphs derived from gene expression data on which to test MCE performance. We concentrate on transcriptomic data, given its abundance, and eschew synthetic data, having learned long ago that effective algorithms for one have little bearing on the other. (The pathological matchings noted in [15] for vertex cover can be extended to clique, but likewise they too are of course hugely irrelevant to real data.) We then implement and compare both standard enumeration algorithms and more advanced codes based on the theory of fixed-parameter tractability (FPT) [1,11]. In an effort to improve performance, we scrutinize the structure of transcriptomic graphs and explore the notion of maximum clique covers and essential vertex sets. Indeed, we find that with the right preprocessing we are able to tailor algorithms to the sorts of data we routinely encounter, and that we can now solve instances previously considered unassailable.

2 Implementation Environment

To test the performance of different algorithmic approaches, we created a testbed of 75 graphs, 25 from each of three different transcriptomic datasets. The datasets were selected for diversity within the domain of mRNA microarray expression experiments. Since experimental conditions drive the correlations used to create the graphs, we selected datasets with three different types of conditions: strain,

time, and stimulus. Two of the datasets were from experiments on *M. musculus* (mouse); the third was from an experiment on *S. cerevisiae* (yeast). In all three, mRNA microarrays were used to measure the intensity of mRNA expression.

Our first dataset used microarrays containing 45127 probes and consisted of expression data collected for adult mouse specimens from 41 different BXD strains. The data was segregated by sex, so we constructed 13 graphs from female data and 12 from male data. Our second dataset used microarrays containing 46632 probes and measured expression on mice on successive days during prenatal and postnatal development on two different strains, C57BL/6 and DBA/2J, at 12 and 13 time points respectively. Our third dataset used microarrays containing 6214 probes and measured expression on yeast at 16 different oxygen levels and 15 glucose concentrations.

To analyze expression data, we first constructed weighted graphs in which vertices represented probes and edge weights were Pearson correlation coefficients computed across experimental conditions. We then converted the weighted graphs into unweighted graphs by retaining only those edges whose weights were at or above some chosen threshold, t. By employing incremental values for t between 0.7 and 0.94, a range typical of correlation values used to analyze microarray data, we obtained a testbed of graphs of various sizes and densities. All size/density values were within the spectrum typically seen in our work with biological datasets. The smallest graph had 5,300 vertices and 292,829 edges; the largest had 30,033 vertices and 1,818,945 edges.

The number of maximum cliques for the graphs in our testbed ranged from 5 to 47496, with no discernible pattern based on graph size or density. One might ask why there is such wide, unpredictable variability. It turns out that the number of maximum cliques can be extremely sensitive to small changes in the graph. Even the modification of a single edge can have a huge effect. Consider, for example, a graph with a unique maximum clique of size k, along with a host of disjoint cliques of size $k-1$. The removal of just one edge from what was the largest clique may now result in many maximum cliques of size $k-1$. Edge addition can of course have similar effects. See Figure 1 for an illustrative example.

3 Fundamental Approaches to MCE

While very little prior work seems to have been done on MCE, the problem of maximal clique enumeration has been studied extensively. Since any algorithm that enumerates all maximal cliques also enumerates all maximum cliques, it is reasonable to approach MCE by attempting first to adapt existing maximal clique enumeration algorithms. An implementation of an existing maximal clique enumeration algorithm also provides a useful runtime benchmark that should be improved upon by any new approach. For completeness, we also consider other enumeration algorithms. One possibility is to compute the maximum clique size and then test all possible combinations of vertices of that size for complete connectivity. While this approach may be reasonable for very small clique sizes, as the maximum clique size increases the runtime of this approach quickly becomes prohibitive.

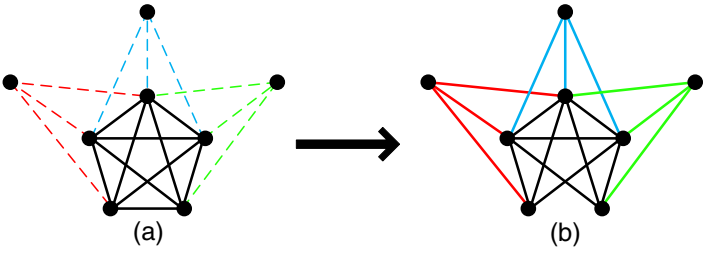

Fig. 1. The number of maximum cliques in a graph can be highly subject to perturbations due, for example, to noise. For example, a graph may contain a single maximum clique C representing a putative network of size k, along with any number of vertices connected to $k-2$ vertices in C. In (a), there is a single maximum clique of size $k=5$, with "many" other vertices (only three are shown) connected to $k-2=3$ of its nodes. In (b), noise results in the removal of a single edge, creating many maximum cliques now of size $k-1=4$.

Current maximal clique enumeration algorithms can be classified into two general types: iterative enumeration (breadth-first traversal of a search tree) and backtracking (depth-first traversal of a search tree). Iterative enumeration algorithms, such as the method suggested by Kose *et al* [19], enumerate all cliques of size k at each stage, test each one for maximality, then use the remaining cliques of size k to build cliques of size $k+1$. The process is typically initialized for $k=3$ by enumerating all vertex subsets of size 3 and testing for connectivity. In practice, such an approach can have staggering memory requirements, because all cliques of a given size must be retained at each step. In [29], this approach is improved by using efficient bitwise operations to prune the number of cliques that must be saved. Nevertheless, storage needs can be excessive, since all maximal cliques of one size must still be made available before moving on to the next larger size. Figure 2 shows the number of maximal cliques of each size in a graph near the mean size in our testbed. This graphic illustrates the enormous lower bounds on memory that can be encountered with iterative enumeration algorithms.

Many variations of backtracking algorithms for maximal clique enumeration have been published in the literature. To the best of our knowledge, all can be traced back to the algorithms of Bron and Kerbosch first presented in [7]. Some subsequent modifications tweak the data structures used. Others change the order in which vertices are traversed. See [18] for a performance comparison between several variations of backtracking algorithms. As a basis for improvement, however, we implemented the original, highly efficient algorithm of [7]. We made this choice for three reasons. First, an enormous proportion of the time consumed by enumeration algorithms is spent in outputting the maximal cliques that are generated. This output time is a practical limitation on any such approach. Second, a graph can theoretically contain as many as $3^{(n/3)}$ maximal cliques [23]. It was shown in [28] that the algorithm in [7] achieves this bound in the worst case. No algorithm with a theoretically lower asymptotic runtime can thus exist. Third, and most importantly, the improvements we introduce do not

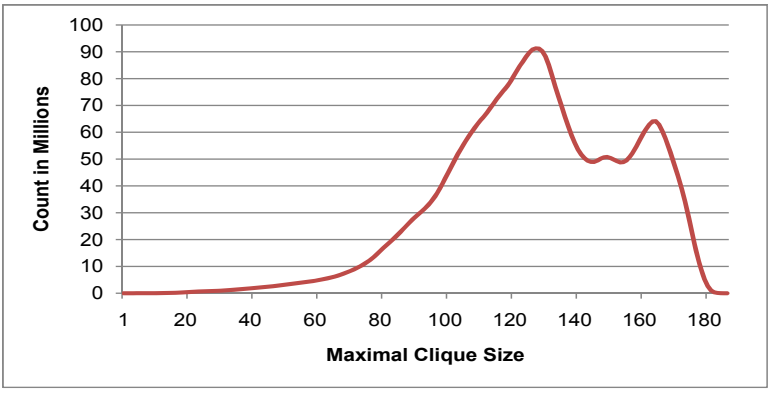

Fig. 2. The maximal clique profile of graph 70 in our testbed. MCE algorithms that are based on a breadth-first traversal of the search tree will retain at each step all maximal cliques of a given size. This can lead to titanic memory requirements. This graph, for example, contains more than 90 million maximal cliques of size 131. These sorts of memory demands tend to render non-backtracking methods impractical.

depend on the particulars of any one backtracking algorithm; they can be used in conjunction with any and all of them.

In the following sections, we test multiple approaches to solving MCE. We first test *Basic Backtracking*, the algorithm of [7]. We next modify this approach to take advantage of the fact that we only want to find maximum cliques. We call this approach *Intelligent Backtracking*. We then modify our existing tool for finding a single maximum clique to enumerate all maximum cliques. We call this approach *Parameterized Maximum Clique*, or *Parameterized MC*. This can be seen as another backtracking approach that goes even further to exploit the fact that we only want to find maximum cliques. Finally, based on observations about the properties of biological graphs, we introduce the concepts *maximum clique covers* and *essential vertex sets*, and apply them to improve the runtime of backtracking algorithms.

3.1 Basic Backtracking

The seminal maximal clique publication [7] describes two algorithms. A detailed presentation of the second, which is an improved version of the first, is provided. It is this second, more efficient, method that we implement and test. We shall refer to it here as Basic Backtracking. All maximal cliques are enumerated with a depth-first search tree traversal. The primary data structures employed are three global sets of vertices: COMPSUB, CANDIDATES and NOT. COMPSUB contains the vertices in the current clique, and is initially empty. CANDIDATES contains unexplored vertices that can extend the current clique, and initially contains all vertices in the graph. NOT contains explored vertices that cannot extend the current clique, and is initially empty. Each recursive call performs the following steps:

- Select a vertex v in CANDIDATES and move it to COMPSUB.
- Save CANDIDATES and NOT lists.
- Remove all vertices not adjacent to v from both CANDIDATES and NOT. At this point, if both CANDIDATES and NOT are empty, then COMPSUB is a maximal clique. If so, output COMPSUB as a maximal clique and continue the next step. If not a maximal clique, then make recursive call.
- Restore CANDIDATE and NOT lists.
- Move v from COMPSUB to NOT. Make recursive call.

Note that NOT is used to keep from generating duplicate maximal cliques. The search tree can be pruned by terminating a branch early if some vertex of NOT is connected to all vertices of CANDIDATES. Vertices are selected in a way that causes this pruning to occur as soon as possible, We omit the details since they are not pertinent to our modifications of the algorithm.

The storage requirements of Basic Backtracking are relatively modest. No information about previous maximal cliques needs to be retained. In the improvements we will test, we focus on speed but also improve memory usage. Thus, such limitations are in no case prohibitive for any of our tested methods. Nevertheless, in some environments, memory utilization can be extreme. We refer the interested reader to [29].

Our implementation of Basic Backtracking solved 29 of the 75 graphs in our testbed within 24 hours. See Figure 4. We therefore have now an initial benchmark upon which we can now try to improve.

3.2 Backtracking with Knowledge of Maximum Clique Size

Given the relative effectiveness with which we can find a single maximum clique, it seems logical to consider whether knowledge of that clique's size can be helpful in enumerating all maximum cliques. We first describe how our software implementation does its job. We then discuss subtree prunings that work to improve the backtracking approach. Finally we implement code changes and test the resultant algorithm on our testbed.

We use the term Maximum Clique Finder (MCF) to denote the software we have implemented and refined for finding a single clique of largest size [12]. MCF employs a suite of preprocessing rules along with a branching strategy that mirrors the well-known FPT approach to vertex cover [1,26]. It first invokes a simple greedy heuristic to find a reasonably large clique rapidly. This clique is then used for preprocessing, since it puts a lower bound on the maximum clique size. The heuristic works by choosing the highest degree vertex, v, then choosing the highest degree neighbor of v. These two vertices form an initial clique C, which is then iteratively extended by choosing the highest degree vertex adjacent to all of C. On each iteration, any vertex not adjacent to all of C is removed. The process continues until no more vertices exist outside C. Since $|C|$ is a lower bound on the maximum clique size, all vertices with degree less than $|C-1|$ can be permanently removed from the original graph. Next, all vertices with degree $n-1$ are temporarily removed from the graph, but retained in a list

since they must be part of any maximum clique. MCF exploits a novel form of color preprocessing [12], used previously in [27] to guide branching. This form of preprocessing attempts to reduce the graph as follows. Given a known lower bound k on the size of the maximum clique, for each vertex v we apply fast greedy coloring to v and its neighbors. If these vertices can be colored with fewer than k colors, then v cannot be part of a maximum clique and is removed from the graph. Once the graph is thus reduced, MCF uses standard recursive branching on vertices, where each branch assumes that the vertex either is or is not in the maximum clique.

Intelligent Backtracking. Given that MCF rapidly finds a maximum clique, we now modify Basic Backtracking to make use of this information. First we compute the maximum clique size k using MCF and apply color preprocessing as previously described to reduce the graph. Only a slight modification to the internals of Basic Backtracking is necessary to make use of k to prune the search tree. Specifically, at each node in the search tree we check if there are fewer than k vertices in the union of COMPSUB and CANDIDATES. If so, that branch cannot lead to a clique of size k, and so we return.

While the modification may seem slight, the resultant pruning of the search tree can lead to a substantial reduction in the search space. As seen in Figure 4, Intelligent Backtracking results in a significant runtime improvement. We are now able to solve 58 of the 75 graphs in our testbed, twice the number solvable by Basic Backtracking.

Parameterized Enumeration. We now modify MCF to find not just one, but all maximum cliques. The modification is straightforward. We maintain a global list of all cliques of maximum size found thus far. Whenever a larger maximum clique is found, the list is flushed and refreshed to contain only the new maximum clique. When the search space has been exhausted, the list of maximum cliques is output.

We must take special care, however, to note that certain preprocessing rules used during interleaving are no longer valid. Consider, for example, the removal of a leaf vertex. The clique analogue is to find a vertex with degree $n-2$ and remove its lone non-neighbor. This rule patently assumes that only a single maximum clique is desired, because it ignores any clique depending on the discarded vertex.

After implementing these modifications, we are now able to solve 63 of the 75 graphs in our testbed. See Figure 4. While this is encouraging, our algorithms are still inadequate to handle readily all our test graphs. We will therefore explore two new approaches that are tailored to exploit properties of the sort of graphs we need to solve.

4 Maximum Clique Covers

We view MCF as a subroutine that can be called repeatedly. This provides us with a simple greedy algorithm for computing a maximal set of disjoint maximum

cliques. We merely compute a maximum clique, remove it from the graph, and iterate until the size of a maximum clique decreases. To explore the advantages of computing such a set, we introduce the following notion:

Definition 1. *A* maximum clique cover *of* $G = (V, E)$ *is a set* $V' \subseteq V$ *with the property that each maximum clique of* G *contains some vertex in the cover.*

The union of all vertices contained in a maximal set of disjoint maximum cliques is of course a maximum clique cover (henceforth MCC), because all maximum cliques must overlap with such a set. This leads to a useful reduction algorithm. Any vertex not adjacent to at least one member of an MCC cannot be in a maximum clique, and can thus be removed.

In practice, we find that applying MCC before the earlier algorithms yields only marginal improvement. When coupled with Intelligent Backtracking, we can solve 59 of our 75 graphs. Even when coupled with Parameterized MC, we are still only able to solve 65 of our 75 graphs. Nevertheless, the concept of MCC leads us to a more useful approach based on individual vertices.

5 Essential Vertex Sets

Our investigation of the MCC algorithm showed us that it typically does not reduce the size of the graph more than the preprocessing rules already incorporated into our clique codes. For example, our clique codes already quickly find a lower bound on the maximum clique size and removes any vertex with degree lower than this bound. Upon closer examination, however, we found that for 74 of our 75 graphs, only one clique was needed in an MCC. Moreover, the lone outlier was sparse and easy to solve. In fact this coincides closely with our experience, in which we typically see high overlap among large cliques in the transcriptomic graphs we encounter on a regular basis. Thus, based on this observation, we shall now refine the concept of MCC. Rather than covering maximum cliques with cliques, we will cover maximum cliques with individual vertices.

We define an *essential vertex* as one that is contained in every maximum clique. Of course no such vertex may exist. On the contrary, however, based on computations we suspect that there are often numerous essential vertices. But even just one may suffice. An essential vertex has the potential to be extremely helpful, because it allows us to remove all its non-neighbors. We employ the following observation. For any graph G, $\omega(G) > \omega(G/v)$ if and only if v covers all maximum cliques, where $\omega(G)$ is the maximum clique size of G.

We define an *essential set* to be the set of all essential vertices, and propose the Essential Set (ES) Algorithm as described in Figure 3. The goal of this procedure is to find all essential vertices in hopes that we can compress the graph as much as possible.

When we run the ES algorithm before Intelligent Backtracking, we are able to solve 74 of 75 graphs. When we run it before Parameterized MC, we are able to solve the entire testbed (see Figure 4). Furthermore, the runtime improvement is great enough to suggest that the algorithm will scale to considerably larger graphs.

```
The Essential Set (ES) Algorithm for Graph Reduction
input: a simple graph G
output: a reduced graph G'

begin
    M = MCF(G) (M is a single maximum clique)
    For each vertex v in M
        G' = G/v
        M' =MCF(G')
        if (|M'| < |M|) then G = G/vertices not adjacent to v (v
covers all maximum cliques)
    end
    output G'
end
```

Fig. 3. Pseudocode for the Essential Set Algorithm

6 Analysis and Discussion

We timed the performance of Basic Backtracking, Intelligent Backtracking, and Parameterized MC on graphs built from biological data. Basic Backtracking was found to be non-competitive. We then reduced the graphs using reduction by MCC and ES and retested with Intelligent Backtracking and Parameterized MC. Run times include both the reduction step and Intelligent Backtracking or Parameterized MC. ES is shown to perform much faster on such instances. The effect is more pronounced at lower correlations, which correspond to larger graphs, though we do not observe strict monotonicity; such a non-monotonic runtime progression is not unusual for $\mathcal{NP}$-complete problems.

ES serves as a practical example of an innovative algorithm tailored to handle a difficult combinatorial problem by exploiting knowledge of the input space. It succeeds by exploiting properties of the graphs of interest, in this case the overlapping nature of maximum cliques. More broadly, these experiments underscore the importance of considering graph types when testing algorithms.

It may be useful to examine graph size after applying MCC and ES, and compare to both the size of the original graph and the amount of reduction achieved by color preprocessing alone. Figure 5 depicts original and reduced graph sizes for a selected subgroup of graphs.

While MCC seems as if it should produce better results, in practice we find it not to be the case for two reasons. First, the vertices in an MCC may collectively be connected to a large portion of the rest of the graph, and so very little reduction in graph size takes place. And second, any reduction in graph size may be redundant with FPT-style preprocessing rules already in place.

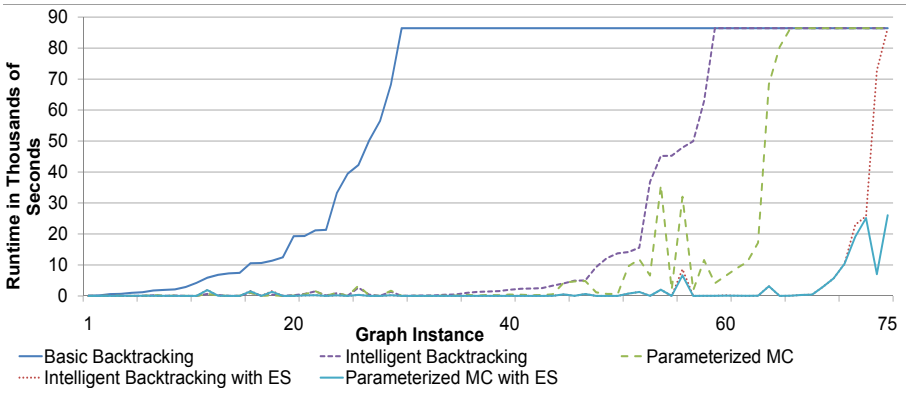

Fig. 4. Timings on various approaches to MCE on the testbed of 75 biological graphs. The tests were conducted under the Debian 3.1 Linux operating system on dedicated machines with Intel Xeon processors running at 3.20 GHz and 4 GB of main memory. Timings include all preprocessing, as well as the time to find the maximum clique size, where applicable. Runs were halted after 24 hours and deemed to have not been solved, as represented by those shown to take 86400 seconds. The graph instances are sorted first in order of runtimes for Basic Backtracking, then in order of runtimes for Intelligent Backtracking. This is a reasonable way to visualize the timings, though not perfect, since graphs that are difficult for one method may not be as difficult for another, hence the subsequent timings are not monotonic.

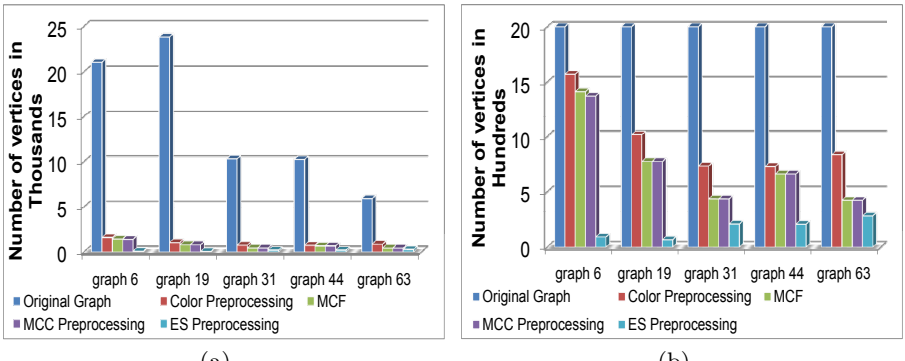

Fig. 5. Reduction in graph size thanks to preprocessing. (a) Five representative graphs are chosen from our testbed. The resulting number of vertices from post-processed graphs are plotted. (b) A closer view of graph sizes.

7 Contrast to Random Graphs

It would have probably been fruitless to test and design our algorithms around random graphs. (Yet practitioners do just that with some regularity.) In fact it has long been observed that the topology of graphs derived from real

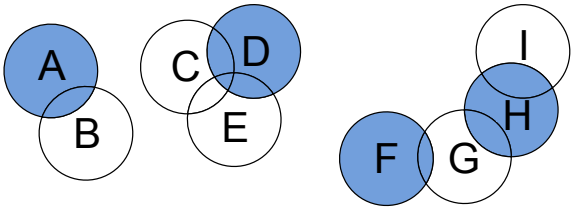

Fig. 6. The Subset Cover Problem. The decision version asks if there are k or fewer subsets that cover all other subsets. A satisfying solution for $k = 4$ is highlighted.

relationships differs drastically from the Erdös-Rényi random graph model introduced in [14]. Attempts to characterize the properties of real data graphs have been made, such as the notion of scale-free graphs, in which the degrees of the vertices follow a power-law distribution [4]. While work to develop the scale-free model into a formal mathematical framework continues [21], there remains no generally accepted formal definition. More importantly, the scale-free model is an inadequate description of real data graphs. We have observed that constructing a graph so the vertices follow a power law (scale-free) degree distribution, but where edges are placed randomly otherwise using the vertex degrees as relative probabilities for edge placement, still results in graphs with numerous small disjoint maximum cliques. For instance, constructing graphs with the same degree distribution as each of the 75 biological graphs in our testbed resulted in maximum clique sizes no greater than 5 for even the highest density graphs. Compare this to maximum clique sizes that ranged into hundreds of vertices in the corresponding biological graphs. Other metrics have been introduced to attempt to define important properties, such as cluster coefficient and diameter. Collectively, however, such metrics remain inadequate to model fully the types of graphs derived from actual biological data. The notions of maximum clique cover and essential vertices stem from the observation that transcriptomic data graphs tend to have one very large highly-connected region, and most (very often all) of the maximum cliques lie in that space. Furthermore, there tends to be a great amount of overlap between maximum cliques, perhaps as a natural result of gene pleiotropism. Such overlap is key to the runtime improvement achieved by the ES algorithm.

8 Future Research Directions

Our efforts with MCE suggest a number of areas with potential for further investigation. A formal definition of the class of graphs for which ES achieves runtime improvements may lead to new theoretical complexity results, perhaps based upon parameterizing by the amount of maximum clique overlap. Furthermore, such a formal definition may form the basis of a new model for real data graphs. We have noted that the number of disjoint maximum cliques that can be extracted provides an upper bound on the size of an MCC. If we parameterize by the maximum clique size and the number of maximum cliques, does an FPT

algorithm exist? In addition, formal mathematical results may be achieved on the sensitivity of the number of maximum cliques to small changes in the graph.

Note that any MCC forms a hitting set over the set of maximum cliques, though not necessarily a minimum one. Also, a set D of disjoint maximum cliques, to which no additional disjoint maximum clique can be added, forms a *subset cover* over the set of all maximum cliques. That is, any maximum clique $C \notin D$ contains at least one vertex $v \in$ some maximum clique $D' \in D$. See Figure 6. To the best of our knowledge, this problem has not previously been studied. All we have found in the literature is one citation that erroneously reported it to be one of Karp's original $\mathcal{NP}$-complete problems [22].

For the subset cover problem, we have noted that it is $\mathcal{NP}$-hard by a simple reduction from hitting set. But in the context of MCE we have subsets all of the same size. It may be that this alters the complexity of the problem, or that one can achieve tighter complexity bounds when parameterizing by the subset size. Alternately, consider the problem of finding the minimum subset cover given a known minimum hitting set. The complexity of this tangential problem is not at all clear, although we conjecture it to be $\mathcal{NP}$-complete in and of itself. Lastly, as a practical matter, exploring whether an algorithm that addresses the memory issues of the subset enumeration algorithm presented in [19] and improved in [29] may also prove fruitful. As we have found here, it may well depend at least in part on the data.

Acknowledgments

This research was supported in part by the National Institutes of Health under grants R01-MH-074460, U01-AA-016662 and R01-AA-018776, and by the U.S. Department of Energy under the EPSCoR Laboratory Partnership Program. Mouse data were provided by the Goldowitz Lab at the Centre for Molecular Medicine and Therapeutics, University of British Columbia, Canada. Yeast data were obtained from the experimental work described in [20].

References

1. Abu-Khzam, F.N., Langston, M.A., Shanbhag, P., Symons, C.T.: Scalable parallel algorithms for FPT problems. Algorithmica 45, 269–284 (2006)
2. Baldwin, N.E., Chesler, E.J., Kirov, S., Langston, M.A., Snoddy, J.R., Williams, R.W., Zhang, B.: Computational, integrative, and comparative methods for the elucidation of genetic coexpression networks. J. Biomed. Biotechnol. 2(2), 172–180 (2005)
3. Baldwin, N.E., Collins, R.L., Langston, M.A., Leuze, M.R., Symons, C.T., Voy, B.H.: High performance computational tools for motif discovery. In: Proceedings of 18th International Parallel and Distributed Processing Symposium (2004)
4. Barabási, A.-L., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
5. Bomze, I., Budinich, M., Pardalos, P., Pelillo, M.: The maximum clique problem. Handbook of Combinatorial Optimization 4 (1999)

6. Borate, B.R., Chesler, E.J., Langston, M.A., Saxton, A.M., Voy, B.H.: Comparison of thresholding approaches for microarray gene co-expression matrices. BMC Research Notes 2 (2009)
7. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Commun. ACM 16(9), 575–577 (1973)
8. Bul, S.R., Torsello, A., Pelillo, M.: A game-theoretic approach to partial clique enumeration. Image and Vision Computing 27(7), 911–922 (2009); 7th IAPR-TC15 Workshop on Graph-based Representations (GbR 2007)
9. Chesler, E.J., Langston, M.A.: Combinatorial genetic regulatory network analysis tools for high throughput transcriptomic data. In: RECOMB Satellite Workshop on Systems Biology and Regulatory Genomics (2005)
10. Chesler, E.J., Lu, L., Shou, S., Qu, Y., Gu, J., Wang, J., Hsu, H.C., Mountz, J.D., Baldwin, N.E., Langston, M.A., Hogenesch, J.B., Threadgill, D.W., Manly, K.F., Williams, R.W.: Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function. Nature Genetics 37, 233–242 (2005)
11. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, New York (1999)
12. Eblen, J.D.: The Maximum Clique Problem: Algorithms, Applications, and Implementations. PhD thesis, University of Tennessee (2010), http://trace.tennessee.edu/utk_graddiss/793/
13. Eblen, J.D., Gerling, I.C., Saxton, A.M., Wu, J., Snoddy, J.R., Langston, M.A.: Graph algorithms for integrated biological analysis, with applications to type 1 diabetes data. In: Clustering Challenges in Biological Networks, pp. 207–222. World Scientific, Singapore (2008)
14. Erdős, P., Rényi, A.: Random graphs, pp. 17–61. Publication of the Mathematical Institute of the Hungarian Academy of Science (1960)
15. Fernau, H.: On parameterized enumeration. In: Proceedings of the 8th Annual International Conference on Computing and Combinatorics (2002)
16. Fernndez-Baca, D.: The perfect phylogeny problem. In: Cheng, X., Du, D.-Z. (eds.) Steiner Trees in Industry (2002)
17. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. WH Freeman & Co., New York (1979)
18. Harley, E.R.: Comparison of clique-listing algorithms. In: Proceedings of the International Conference on Modeling, Simulation and Visualization Methods, pp. 433–438 (2004)
19. Kose, F., Weckwerth, W., Linke, T., Fiehn, O.: Visualizing plant metabolomic correlation networks using clique-metabolite matrices. Bioinformatics 17, 1198–1208 (2001)
20. Lai, L.C., Kosorukoff, A.L., Burke, P.V., Kwast, K.E.: Metabolic-state-dependent remodeling of the transcriptome in response to anoxia and subsequent reoxygenation in saccharomyces cerevisiae. Eukaryotic Cell 5(9), 1468–1489 (2006)
21. Li, L., Alderson, D., Doyle, J.C., Willinger, W.: Towards a theory of scale-free graphs: Definition, properties, and implications (extended version). Internet Mathematics (2005)
22. Malouf, R.: Maximal consistent subsets. Computational Linguistics 33, 153–160 (2007)
23. Moon, J.W., Moser, L.: On cliques in graphs. Israel Journal of Mathematics 3, 23–28 (1965)
24. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814–818 (2005)

25. Perkins, A.D., Langston, M.A.: Threshold selection in gene co-expression networks using spectral graph theory techniques. BMC Bioinformatics 10 (2009)
26. Rogers, G.L., Perkins, A.D., Phillips, C.A., Eblen, J.D., Abu-Khzam, F.N., Langston, M.A.: Using out-of-core techniques to produce exact solutions to the maximum clique problem on extremely large graphs. In: ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2009), IEEE Computer Society, Los Alamitos (2009)
27. Tomita, E., Kameda, T.: An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. Journal of Global Optimization 37, 95–111 (2007)
28. Tomitaa, E., Tanakaa, A., Takahashia, H.: The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical Computer Science 363(1), 28–42 (2006)
29. Zhang, Y., Abu-Khzam, F.N., Baldwin, N.E., Chesler, E.J., Langston, M.A., Samatova, N.F.: Genome-scale computational approaches to memory-intensive applications in systems biology. In: Supercomputing (2005)