Ammolite: Aligned Molecule Matching

David Danko, *Massachusetts Institute of Technology*

This semester I have continued a project I began last semester with a post-doctoral researcher, Noah Daniels, and the Berger group at CSAIL. The goal of the project is to develop a faster multifold search for similarities between small molecules.

Background

Small organic molecules are an important part of modern society. Every modern drug and many modern materials are based off of small organic molecules. Until recently every new drug has included the production of one or more novel molecules which require a their own process for synthesis and FDA approval.

Recently researchers have started to use the large existing database of molecules to search for compounds similar to drugs they want to synthesize. This allows for shorter drug development and approval times. Unfortunately current search methods for molecules are too slow to be practical on very large databases.

Current Approaches to Small Molecule Search

Current techniques for small molecule search treat molecules as graphs. Vertices represent atoms and edges represent the bonds between them. To represent different types of atoms vertices are colored; different types of bonds are represented by different edges. This representation is simple but works fairly well for small organic molecules which rarely have complicated bond types or exotic isotopes. The similarity between a pair of molecules is strongly related to the size of their maximum common subgraph, the portions of each molecule which are identical. Representing molecules as labeled, weighted graphs is a common if slightly flawed way of representing molecules. It cannot convey the differences between cis and trans molecules or between resonance structures but it still provides a large amount of information.Two molecules with similar structure will have a large MCS and may have similar function.

The size of the MCS relative to its parent molecules is generally computed as one of two coefficients: the tanimoto coefficient and the overlap coefficient. These provide a numerical way to compare large sets of molecules.

$$Tanimoto = \frac{MCS}{G_1 + G_2 - MCS}$$

$$Overlap = \frac{MCS}{min\{G_1, G_2\}}$$

Current approaches to SMS use these coefficients to affect a simple linear search. A coefficient (of either type) is calculated between a query molecule and each molecule in a target set. The molecules with the coefficients above some predetermined threshold are then returned as results.

Calculating Maximum Common Subgraphs

Simple SMS techniques rely on a fast way to calculate MCS. Unfortunatley finding a maximum common subgraph is NP-Hard in general but it is usually possible to come up with a good aproximation. Most SMS techniques have focused on making these approximations fast.

Since simple SMS techniques run in linear time with respect to the size of the database faster ways to calculate the MCS result in appreciably faster search. Even fast approximations of MCS are relatively slow taking about 50-250ms per operation. The pubchem small molecules database (the largest publicly available small molecule database) contains about 65 million molecules. A single query performed with simple SMS using MCS would take about

$$\frac{38days}{1query} = \frac{50ms}{1mcs} * \frac{65 * 10^6 mcs}{1query} * \frac{1hr}{3.6 * 10^6 ms} * \frac{1day}{24hr}$$

Which is quite impractical.

Techniques For Calculating the MCS

The project so far has employed three different techniques for calculating the MCS between two molecules.

The first approach tried was to reimplementat fMCSr (Girke, et al.) in Java. This technique allowed us to find MCS with slight differences (potentially resulting in larger, more accurate MCS) but the current reimplementation is too slow.

The second approach tried was the SMSD library (Rahman, et al). This library was written natively in Java. The current version of Ammolite uses SMSD for all MCS calculation but the version of SMSD used may be too slow.

The third approach we plan to use as a corallary to SMSD is the IsoRank algorithm (Singh, et al.) We believe this algorithm might produce a good approximation of SMSD's results on unlabeled graphs but would be signifigantly faster.

Our Approach

Rather than attempt to create a faster way to calculate MCS our project has focused on using existing MCS techniques as tools to create a faster search technique. Currently our approach to search combines two distinct techniques both of which we developed. Either could be used seperately to accelerate search but they complement each other well.

Compressed Representation of Molecules

Compression is a specialty of the Berger group so our initial approach to search acceleration focused on compression. Our goal was to create a compression schema that would produce databases which could be searched by simple SMS techniques. A first pass would search the compressed database returning a few promising targets. These targets would then be decompressed to the molecules they represented. A second pass would search the set of decompressed target molecules returning any molecules above a given threshold. This would improve the runtime of search to $O(compressed - database + target - decompression)$. In the best case this could be $O(\sqrt{database})$.

Once we had an approach to searching a compressed database we began to think about what requirements this would place on our compression schema. Compressed version of molecules would need to overlap with one another, hopefully signfigantly. It would not be helpful if no two molecules had the same compressed representation. Additionally the compressed representation of molecules would need to be searchable with MCS. A good match between a query and compressed representations of target would have to correlate with a good match between a query and the targets themselves.

Using MCS to match molecules in coarse and fine search was a key requirement so we began our search for a good compression schema by modifying the graph representation of molecules. Organic molecules have common

structural motifs, like rings, tails, and functional groups. Often, however, molecules vary in the exact types of atoms that compose these groups. It made sense that molecules with similar structural motfis should be grouped together regardless of their exact elemental composition. This implied that we could start a compression schema by making a simple change to the standard representation of molecules as weighted, labeled graphs: we would simply represent molecules as unlabeled, unweighted graphs.

We developed several different compression techniques based off of representing molecules as unlabeled graphs; each yielding progressively better performance. In every case we removed hydrogens from molecules before producing graph representations. Hydrogens tend to be chemically uninteresting, vastly increase computational complexity, and reveal a great deal of the information we were trying to abstract by compression.

Our earliest and most generic technique to group molecules was to convert their representation as labeled, weighted graphs to unweighted, unlabeled graphs. With this technique molecules like cyclopentane, cyclopentene, and THF would be clustered together. Molecules with tails like, like methoxycyclohexane, would be put into a separate group, and so on. On its own this technique produced somewhat lackluster results. Compression ratios typically improved when large databases were compressed because the database contained more redundancy. However even on large tests this technique was only able to find about 1 representatives to 2 actual molecules. Large complex molecules generally were grouped by themselves while small simple molecules were grouped into large groups together.

The second clustering technique we tried was to remove tails from molecules long chains of acyclic carbons. This allowed us to group molecules like cyclohexane and methoxycyclohexane together.

After the initial graph representation was produced from a molecule every vertex of degree one was removed from the graph. This process was repeated iteratively until it converged. This prodcued a representation of the original molecule which only contained vertices that were part of cycles (simple and complex cycles). Molecules that were acyclic (many alkanes, alkenes, and alkynes) were represented as a sparse graph with all original vertices but no edges.

This technique produced significantly larger clusters. There was approximately 1 representatives for 10 actual molecules. This technique also produced groups with signifigantly more even size. There were some large molecules which were not grouped with any others but most groups consisted of less than 20 molecules.

Building off this success we developed a third technique. Any vertex or edge that was not part of a simple cycle was removed. With this technique Bicyclohexyl and Dicyclohexylamine would be assigned to the same group. Edges that were not part of a simple cycle were identified using a modified version of depth first search. This was relatively slow compared to our other compression schema (this isn't necessarily a problem as compression time doesn't greatly impact search time. Only one molecule needs to be compressed for each query the query itself.) but produced initially promising results. This technique resulted in initially promising results with about 1 representative to 20 molecules.

Once we had compression schema which met our first criteria, reducing the size of the original database, we needed to determine if a match on the compressed representation of molecules would correlate with matches on the molecules themselves. Our first compression technique performed well. There was a strong correlation between the size of the MCS on unweighted unlabeled graphs and the orignal graphs. The second compression technique also performed relatively well. Once their tails were removed most molecules still retained most of the critical portions of their orignal structure. A few molecules had almost no correlation with their compressed structure and many molecules which did not have particularly similar actual

structures had fairly similar tailless structures but there was enough of an overall correlation.

Our third technique fell flat here. There was no correlation whatsoever between the size of the MCS on the ring representation of molecules and the molecules themselves. This was surprising but did not sstop our project. Since it had reasonable compression ratios and correlation we decided to use our second compression technique.

Compressing a Database

For each clustering technique the same approach was used to compress the database. As molecules were read from file they would be converted to their compressed representation (which varied according to technique). The structures would then be sorted by a preliminary hash into groups based on their size and order.

This hash was used to improve compression performance. If two graphs were isomorphic they would definitely hash to the same spot but if the graphs were not isomorphic they would likely hash to different locations. This was implemented by finding the ten vertices with highest degree, sorting them by degree, and building a numeral out of the sorted degrees. The use of a preliminary hash decreased compression times from several days to just a few minutes.

Finally the entry in the hashtable where the query structure was hashed would be iteratively searched for any structure that was isomorphic to the query structure. If no isomorphic structure was found the query structure would be added to the hashtable entry in its own right. If any structure isomorphic to the query structure was found it was given a pointer to the molecule from which the query structure had been generated.

Once the entire set of molecules in the database had been appropriatley matched the hash table was serialized for easy retrieval later.

Searching the Compressed Set of Molecules

Searching the compressed database is more complex than simple SMS techniques but still fairly straightforward. The query molecule is converted to the same sort of representation that was used to compress the database. The MCS is calculated between the query-representative and the representatives in the database. Representatives which match above a threshold are converted into sets of molecules. The MCS is then calculated between the original query and these molecules. Molecules above a certain threshold are returned.

Simple SMS techniques only require users to input a threshold. Any molecules over the threshold are returned as results. Searching on our compressed databases requires a threshold and a probability. The threshold works the same way as in simple SMS; any molecule found to match over the threshold is returned as a result. However because our technique does not calculate a MCS between a query and every target molecule it is possible we miss certain molecules which were good matches to the query. This is where the probability becomes useful. To search the set of representatives we need to convert the original threshold to a different, usually lower value, representatives which match above the converted threshold are likely to correspond to some molecules which match above the actual threshold. The probability paramter allows us to tune how we set the converted threshold. A lower probability will only select representatives which match the query representative very well. A higher probability one will be more permissive; this will generate more results but will also be slower. The use of two parameters allows users to tune the number of results and the speed of the search. If a user needs all possible results they can set the probability near one. If the user only needs a few good results they can set the probability lower.

In all cases a higher threshold will result in fewer results but this paramter has a smaller affect on search speed than probability. This

is because a MCS needs to be calculated between the query-representative and every target representative. The number of target representatives is asymptotically larger than the number of molecules each representative corresponds to. A higher threshold also reduces the number of results relatively linearly whereas a higher probability can generate many more representative matches.

Informally the size of the representative set is linear with respect to the original database dpending on the compression ratio of the technique used. Unlabeled graphs provide a somewhat pathological case for most methods that calculate the MCS taking roughly twice as long to calculate on average. Our preliminary results suggest that we will be able to achieve a 4-5 times speedup searching a compressed database compared to simple SMS techniques on an uncompressed database. This can be done without a signifigant decrease in the quality of results.

Clustering for Potentially Greater Speed Gains

My work this semester has largely been focused on developing a clustering technique to search databases of molecules.

The clustering technique I've developed could be used independently of database compression techniques but I've developed them together. This was partly for expediency; there was a large body of code I had written for database compression which provided a useful framework for clustering. More importantly database compression also provided potential speedups for clustering. The overall goal of the project was to create a fast tool for SMS. Not to adhere to a specific technique.

Hierarchical Clustering of Molecules

Viewed differently our original technique for database compression could be considered a type of clustering. Real molecules were clustered based off of similarities in their structure. This technique only allowed for a single layer of clustering; because of the way they were generated representatives in the compressed database could not be further compressed.

Standard clustering techniques, like K-Means, were not helpful because they require elements to be represented as points in Euclidean Space. To the best of our knowledge no standard way to express graphs in Euclidean Space exists.

Certain clustering techniques only require a pairwise distance function between elements. The simplest of these techniques is aggregation clustering. In aggregation clustering distances are calculated between every pair of elements. The closest pair of elements is then selected and made into a cluster. The distances between the selected pair of elements and every other element are averaged and a new closest pair of elements is selected. This is essentially a greedy approach to clustering.

Like many greedy approaches aggregation clustering can produce locally optimal but globally suboptimal results. A suboptimal clustering wouldn't concern us on its own but aggregation clustering has a tendency to drift. It starts by selecting elements which are very similar but can quickly end up selecting groups of elements which are similar on average but contain very dissimilar elements.

We ammeliorated this issue by creating a checked version of aggregation clustering. Every time a cluster is generated a centroid for the cluster is also generated. When a new element is added to the cluster it is compared to the centroid. If the new element is too different from the centroid it is rejected from the cluster. Otherwise the new element is added to the cluster and the centroid is recomputed. This results in somewhat smaller but much tighter clusters.

The pairwise distance function we chose for aggreagtion clustering was inevitably the size of their MCS. This suggested a natural way to calculate the centroid. In order to calculate the size of the MCS we had to calculate the MCS itself. When attempting to add a new element

to a cluster we could calculate the MCS between the new element and the MCS which was the centroid of the cluster. If the resulting MCS was large enough (based on a threshold) we had a molecule which matched the cluster well. Otherwise the element was rejected.

Using MCS as centroids also suggested a natural way to perform multiple layers of clustering. A first pass at checked aggregation clustering would find some set of clusters, each of which had a MCS as a centroid. Since the MCS was itself a graph we could repeat the process of checked aggregation clustering on the set of centroids. This process can be continued arbitrarily.

In preliminary tests we have observed about 5 layers of clustering before we are unable to produce more clusters.

Problems with Checked Aggregation Clustering and Refinements

In principle checked aggregation clustering seemed to work quite well. However it was too slow to work in practice. Since aggregation clustering calculates the distance between every pair of elements it requires $O(n^2)$ distance operations on the number of elements. Since databases of small molecules can be huge (up to 65 million entries) and calculating the distance between two molecules using MCS is slow aggregation clustering is infeasible on large datasets. Even small datasets require a large amount of time to compress which made intial testing difficult.

Rather than calculate all pairwise distances up front we refined checked aggregation clustering to reduce the number of required MCS operations. Each element to be clustered was compared to existing clusters. If the element was found to match any cluster well enough (that is the resulting MCS between the element and the centroid MCS wasn't too small) the element was immediatley added to the cluster and no further calculations were performed. If the element was not found to match any cluster well enough after having been compared to all of them (the worst case scenario) it became its own cluster with a single element.

Though this approach is still worst case $O(n^2)$ it is much faster than the original version of checked aggregation clustering in practice. What's more surprising is that the clusterings it produces are very similar to the clusterings produced by the original version of checked aggregation clustering.

Because the process of creating clusters is quite slow using our compression method for databases can be quite beneficial. Large databases can be compressed to a fraction of their original size before being clustered.

Producing clusters requires a threshold for how similar each member needs to be to the centroid of its cluster. A higher threshold results in more smaller clusters. Different clusterings will impact the speed and quality of search but it is not computationally feasible to produce many different clusterings on the same database. Once the program is complete a good clustering threshold will be emprically determined and built into the program.

Searching a Hiearchical Clustering

Multiple layers of clustering allow us to build a tree which can be searched layer by layer. If a match between a query molecule and the representative cenroid of a cluster is found we will compare the query to the members of the clusters (which can themselves be clusters) repeating the process until we have reached the bottom of the tree. At this point we can perform either a simple SMS or a two-fold SMS on our compressed database.

Every time the query molecule does not match a cluster we reject a large number of molecules that we would otherwise have compared our query molecule to directly. This could potentially reduce the quality of results though this may be worthwhile depending on the speed gains.

Searching clusters requires a user to input a probability and a threshold; analagous to searching a compressed database. Higher

probability means that clusters will be rejected less often; lower, more. This could potentially have a large impact on speed.

In the best case searching a Heirarchical Clustering is $O(\log(n))$ in the size of the database.

Ongoing Work

The programs for database compression and clustering are completely developed. We are currently working on refining our current values for certain parameters: notably the threshold conversion for database compression and the threshold value for clustering.

We are also working to integrate a faster way of calculating MCS. The methods we have tried so far have been too slow for anything more than very small tests. This is the only thing preventing us from running large scale timed runs of the system; I expect to have this rectified soon.

Once I have large scale timed runs the only portion of the project left is too clean up the codebase for release and to write a paper to be published. This should be done within about a month.

References

[Wang Y, Backman T, Horan K and Girke T]
fmcsR: Mismatch Tolerant Maximum Common Substructure Searching.

[S. A. Rahman, M. Bashton, G. L. Holliday, R. Schrader and J. M. Thornton]
Small Molecule Subgraph Detector (SMSD) toolkit Journal of Cheminformatics 2009, 1:12. DOI:10.1186/1758-2946-1-12

[Rohit Singh, Jinbo Xu, and Bonnie Berger]
Global alignment of multiple protein interaction networks with application to functional orthology detection Proc. Natl. Acad. Sci. USA, 105:12763-12768.