International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Virtual Screening through Substructure Matching

Kunal Sonalkar[1], Akshay Jain[2]

[1, 2]Computer Science and Engineering, National Institute of Technology, Calicut
NIT Calicut Campus PO Chattamangalam, Kozhikode, Kerela, India

Abstract: ***Virtual Screening involves use of computational techniques for selection of active compounds from a pool of unscreened compounds. It is the elementary phase of drug-development process .These selected active compounds are then further subjected to screening against specific biological target There are two types of virtual screening techniques : Ligand-based and Structure-based. Our project deals with ligand-based screening approach. This approach is used in the absence of biological targets 3D structure synopsis and it banks on the knowledge of active and inactive compounds for the specific target. The number of computing tools and software using target and ligand based drug discovery is increasing at an exponential rate. When we use computational techniques to find the maximum common sub graph between two compounds it is very economical as compared to wet lab experiments.***

Keywords: virtual screening, MCS problem, parallel computing, substructure matching

1. Introduction

Graphs are data structures which can depict structured objects. It represents positions with arbitrary connections between them. Graphs can be used to model complex biological structures, chemical compounds, networks, etc. A graph structure contains a finite set of ordered pairs, called edges or arcs of vertices. A sub graph of a graph G is another graph with the vertex set being a subset of that of G. The edges of a sub graph is also a subset of that of G, but restricted to the vertex set of the sub graph. Many complex compounds are represented as graphs and they have attributes similar to other compounds which resemble with their structures. They are termed as Structurally Similar molecules. They exhibit similar chemical properties and biological activity. So it is convenient to find common structure between compounds when represented as graphs.

MCS, known as Maximum Common Substructure Problem denotes the largest common substructure between the graphs under observation. The maximum common sub graph (MCS) problem has become increasingly important in those aspects of chemo-informatics that involve the matching of 2D or 3D chemical structures. The maximum common sub graph of two graphs can be obtained by constructing a search tree. As the size of the graph increases the solution tree also burgeons combinatorial. There are a few peculiar methods which can be employed to resolve the MCS problem. A general solution to maximum common sub graph comprises of the generation of the search tree using a back track method and an alternative method is by reduction of the MCS problem into maximum clique enumeration.

2. Literature Survey

Virtual Screening is an early stage of drug discovery which helps in ranking the compounds based on the extent of similarity. This similarity ranking can be achieved with structural similarity measures. There are many parameters which can be used to judge the extent of similarity like Jaccard's Coefficient, Tanimoto Coefficient and Cosine Rule, etc. For many applications it is important to find maximal common sub graphs in two graphs. As the problem is NP-complete it cannot be solved for arbitrarily large graphs. Searching for connected maximal common sub graphs can reduce the complexity of the problem drastically, although it remains NP-complete.

A non empty set S of vertices of G forms a complete graph if each vertex of S is joined to every other vertex of S. A complete sub graph of G is called a clique if it is maximal i.e., if it is not contained in any other complete sub graph of G. A clique is a complete maximal sub graph. Maximal Clique problem comes into picture only when we are interested in finding out the clique of maximum cardinality. Because this problem is important for many practical applications, a lot of research has been done to generate optimum algorithms, for example the branch and bound algorithm, the recursive backtracking method of Tarjan and Trojanowski, etc. These methods work for arbitrary graphs and have exponential runtime. So as these algorithms are brute force algorithms, they are not very feasible

CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model that boosts computing performance. It does the above mentioned process by using the enormous amount of GPU. The programming model is created by NVIDIA. Each CUDA compliant device is a set of multiprocessor cores, capable of executing large number of threads concurrently. Here there are various threads which perform a same function but on different data and hence help in doing tasks at a faster pace. Every processor operates on a unique thread id and particular allocated space. The multiprocessor cores are called device and the main CPU is called host. Both host and device have their own separate memory, referred as host memory and device memory.

Producer: In CUDA programming, the producer object is regarded as the one which makes pair of an unknown compound and a target compound and places the pair in a particular thread.
Consumer: The consumer object is the one which loads the pair made by the producer object into a GPU to perform parallel computing.
Producer Consumer Synchronization: This method ensures that as soon as a pair (unknown and target compound) is processed, the new pair can be loaded into the

CPU thread. This helps in continuous processing of the pool of unknown compounds with the target compound.

3. Maximum Clique Enumeration and Bron-Kerbosch Algorithm

3.1 Maximum Clique Enumeration

The clique problem is one of the six fundamental problems of known NP-complete problems. The problem of maximum clique enumeration (MCE) is to find the clique with largest number of vertices. A facile way is to enumerate all maximal cliques in the graph and select the largest among them. The running time of the algorithm depends upon the number of maximal cliques in the graph. As the clique size increases the time to process the edge product graph also increases significantly.

3.2 Bron-Kerbosch Algorithm

1. Let A be the set [u1,u2.....um]
2. if (A = Null) and (B = Null) then
3. Report-Clique.
4. else
5. for i=1 to k do
6. A = A - [u]
7. A' = A
8. B' = B
9. n = v ∈ |u, v ∈ E
10. Enumerate – Cliques (V0 ∪ Ui, A' ∩ N, B ∩ N)
11. B0 = B ∪ Ui
12. end for
13.end if

The above stated algorithm finds all the possible cliques exactly once. Three sets V', A, and B are the main operational sets. Set V' contains vertices of the current active set. Set A has all the vertices which can be utilized for the finishing of V' because they are adjacent to the vertex added last to V'. In B all vertices are collected, which can no longer be used for the completion of C, because all cliques containing these vertices are always generated. The algorithm starts with the empty sets V' and B. Initially, A includes all vertices of the graph G. If A and B are empty, a clique was found and will be reported (line 03). Besides each vertex of A is considered in a loop (lines 04 to 11), where the arbitrarily chosen vertex ui is eliminated from A. A and B are copied into A' and B', respectively (line 06 and 07) for the recursion. The neighbors of vertex ui are generated and stored in N (line 08). Vertex ui is added to V' and the recursion call with A' and B' takes place. All vertices in V' are adjacent in a pair of two; i.e. V is a vertex set of a complete sub graph. A and B will form the complete vertex set of all those vertices which are adjacent to V. Each vertex u belonging to set A is adjacent to all vertices in V'. The vertices from A are used for the extension of V'. Once a vertex from A is used for the extension of V' it will be moved to B. Each vertex u belonging to set B is adjacent to all vertices in V'. The vertex sets of all cliques containing V' U u are already enumerated once. Once the call of the function ENUMERATE CLIQUES () is over; the set of vertices with all cliques containing V' are generated exactly once.

4. Algorithm Design and Implementation

4.1 Building an Edge Table from SDF File

An edge table describing number of elements, nature of bonds present between two atoms, number assigned to an element in the algorithm was built from extracting this information from a given sdf file of the considered compound. Entire algorithm was implemented in python and the ligand interaction data can be obtained as sdf file from NCBI Pubchem repository..

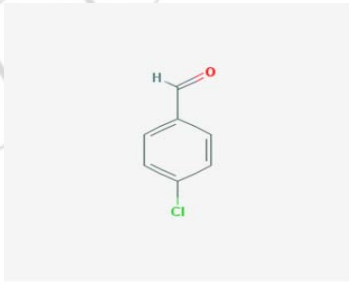
4.2 Conceiving edge product graph

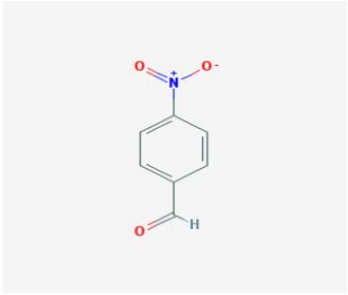
During this juncture, the maximum common sub graph problem is transformed into maximum clique enumeration problem. When given two graphs it is possible to map parts of both the graphs onto one another. These compatibility details of the bonds present between elements is stored in Edge Product Graph. The maximum clique present in this product graph gives us the maximum common sub graph between the two examined compounds

4.3 Bron Kerbosch (BK) algorithm for substructure matching

Bron-Kerbosch algorithm, also known as BK algorithm furnishes the maximum clique present in the edge product graph of the two compounds. Once we get the maximum clique present, we can easily select the maximum common substructure between the two compounds by utilizing details from our earlier created matrix.

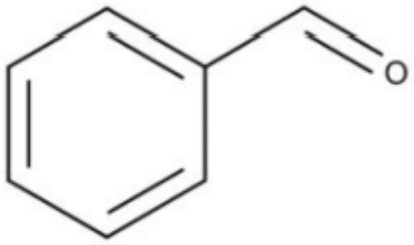
4.4 BK Algorithm application





If we consider the two given figures: **4-chlorobenzaldehyde** and **4-nitro-benzaldehyde**, by using Bron-Kerbosch algorithm on the edge product graph of these two compounds we get **1-benzaldehyde** as the maximum common

substructure between the two graphs. The result is depicted below as follows.



5. GPU based parallel algorithm

5.1 Parallel Algorithm Design and Description

The algorithm starts with a search initiated from a special root node, which represents the lower bound for the optimal solution. In the branch and bound algorithm, the branching is done according to the search strategy adopted by the algorithm. The proposed algorithm uses Breadth first search method for branching. A set of nodes called the **currentActiveSet** contains the list of nodes to be evaluated, which are not yet branched. The bounding procedures are used for the evaluation.

The algorithm starts serially, filling entries in the currentActiveSet until there are enough number of nodes to generate considerable number of threads. GPUs can execute large number of threads in parallel, each thread doing simple operations. In this algorithm, each thread will take a node from the currentActiveSet and then evaluate a small portion of the total search space.

Keeping the given node as root, each thread performs a Breadth First Search operation on the graph. Here each thread must know the branching and evaluation rules. The new nodes found by each of the threads will be added into the set of partial solutions.

The newly generated nodes are then evaluated for simpler procedures. The number of times the kernel is called, depends upon the size of the problem. Each time the kernel is called, more threads will be launched and the sub problems analysed will be more restrictive than the sub problem used in the previous searches.

Parallel BK algorithm Design

```
Producer()
for each graph G3 in G2 do in CPU(in parallel)
   G=Edge Product of G1 and G3
   Add G to buffer of EP.graphs
Consumer()
Count=1
while(count ≤ n)
   Select a graph G from buffer
   V=Number of vertices in G
   A=Initiate Current Active Set()
   t=Calculate the number of threads()
   b=Number of blocks
   Allocate device memory
   Copy Current Active Set to device memory
   While(No Change in A)
     for all nodes p in A do in GPU (in parallel)
        for all neighbours nbr in p
        if nbr is connected to all parents of p in A
        atomically add nbr to A
        Synchronize GPU data
   Backtrack to get the Maximum Clique
   Rank the compound and add it to Rank list
Sort the Rank list
Display the Rank list
```

5.2 Steps involved in computation

1) Consider the known compound and the pool of unknown compounds. Start pairing the target compound with an unknown compound sequentially.
2) Allocate five threads in CPU to store these first five pairs.
3) After calculating the edge product graph in CPU, the EP graph is sent to the GPU by consumer function for calculation of maximum common substructure.
4) As soon as the computation in first thread is done, it takes the next unknown(query) compound from the pool. This is done by Producer Consumer Synchronization method.
5) The above stated process goes on till all the query compounds have been classified on the basis of extent of similarity.
6) The extent of similarity is calculated by finding out the cosine co-efficient for the given unknown compounds and sorting them accordingly.

6. Comparison between Serial and Parallel Algorithm

The first level implementation of parallelized BK algorithm considers two compounds and engenders the maximum common substructure between them. We applied the same underlying principle by comparing a particular active compound with a pool of unknown query compounds to generate a common substructure with maximum similarity between the considered compounds. The unknown query compounds were ranked according to the extent of similarity with the target compound.

(A,B)	(A,B)	EP GraphSize	Maximal CliqueSize	CPUTime	GPUTime
(18,19)	(17,17)	53	10	93.40	3.08
(18,19)	(20,21)	85	13	210.89	3.09
(31,31)	(17,17)	87	10	346.08	3.30
(18,19)	(23,24)	89	15	356.70	5.33
(18,19)	(22,23)	103	15	924.26	4.56
(18,19)	(31,32)	109	7	1028.35	3.27
(20,21)	(22,23)	125	10	N/A	5.63
(20,21)	(31,32)	133	15	N/A	5.68
(18,19)	(35,37)	136	12	N/A	3.69
(25,26)	(20,21)	144	14	N/A	5.49
(18,19)	(38,40)	148	10	N/A	4.56
(25,26)	(23,24)	155	16	N/A	17.4

References

[1] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. Caves, Multinational Enterprise and Economic Analysis, Cambridge University Press, Cambridge, 1982. (book style)
[2] Andreas Klockner. PyCUDA: GPU programming with python.New York University, NY 10012, United States.
[3] R. Chudobaa, V. Sadilekb, R. Rypla. Using Python for scientific computing: Efficient and flexible evaluation of the statistical characteristics. 2012
[4] Andreas Klockner, Nicolas Pinto, Bryan Catanzaro,Yunsup Lee, Paul Ivanov and Ahmed Fasih. GPU Scripting and Code Generation with PyCUDA.23rd April, 2013
[5] Andreas Klckner. Courant Institute of Mathematical Sciences. PyCUDA and PyOpenCL. July 2012.
[6] Ina Koch. Enumerating all connected maximal common sub graphs in two graphs. Theoretical Computer Science, 250(1-2):1-30,2001. H.H. Crokell, "Specialization and International Competitiveness," in Managing the Multinational Subsidiary, H. Etemad and L. S, Sulude (eds.), Croom-Helm, London, 1986. (book chapter style)
[7] Min Ragan-Kelley. Particle Simulation on a GPU with PyCUDA. University of California, Berkeley
[8] James J. McGregor. Backtrack Search Algorithms and MCS problem,12:23-34,ss1982. J. Geralds, "Sega Ends Production of Dreamcast," vnunet.com, para. 2,:Jan.31,2001.

Author Profile

Kunal Sonalkar received the degree Bachelor of Technology in computer science and engineering from National Institute of Technology, Calicut, Kerela, India. He is currently pursuing his masters at University of Florida, Gainesville, FL, USA. His research interests include data mining, machine learning and data science.

Akshay Jain received the degree Bachelor of Technology in computer science and engineering from National Institute of Technology, Calicut, India. He is currently working at Oracle Server Technologies at Banglore, India. His interests include analysis of algorithms and computer networks and security.