Projeto e Análise de Algoritmos
Maximum Common Substructure

Raquel Yuri da Silveira Aoki (201566918)

[1]Universidade Federal de Minas Gerais (UFMG)
Departamento de Ciência da Computação

raquel.aoki@dcc.ufmg.br

1. Introdução

Na química, um problema de interesse é detectar qual a maior sub-estrutura comum a duas estruturas químicas. Uma estrutura química refere-se ao arranjo espacial das massas em uma molécula e as ligações eletrônicas que mantém os átomos separados. Essas estruturas químicas podem variar das muito simples, como a molécula de água, às muito complexas, tais como molécula de câncer e DNA.

Uma forma de resolver esse problema é modelando as estruturas como se fossem grafos, dessa forma o problema se torna o *Maximum Common Subgraph* (MCS). O MCS é conhecidamente um problema NP-completo[4]. Existem bons algoritmos para obter o MCS exato, entretanto no pior caso eles demandam um tempo não polinomial.

Para o trabalho final de Projeto e Análise de Algoritmos de 2015/2, será proposta uma heurística para resolver o problema do MSC e seus resultados serão comparados com um $baseline$.

Na Seção 2 é mostrada a modelagem que será utilizada, o $baseline$ e a heurística proposta, na Seção 3 será mostrada a análise de complexidade dos algoritmos utilizados. A Seção 4 tem a análise dos experimentos, na Seção 5 mostra o uso e a compilação dos algoritmos e por fim na Seção 6 são feitas as considerações finais.

2. Modelagem

Como citado anteriormente, uma estrutura química refere-se ao arranjo espacial das massas em uma molécula e as ligações eletrônicas que mantém os átomos separados. A Figura 1 mostra um exemplo de estrutura química em 2D e 3D. A partir da sua forma 3D, é fácil deduzir um grafo, em que os átomos(massas) são os vértices e as ligações as arestas. Dessa forma, duas estruturas químicas E_1 e E_2 em que se deseja obter a maior sub-estrutura comum, se tornam os grafos não direcionados G_1 e G_2 respectivamente.

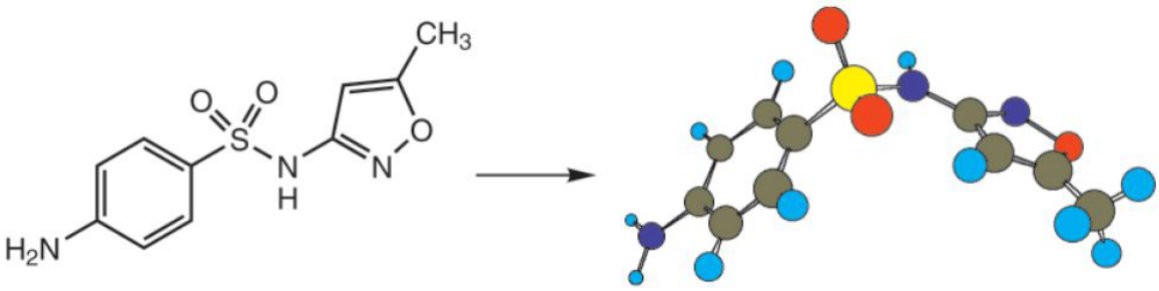

Figura 1. Ilustração de uma estrutura química na forma 2D e 3D

Considerando as estruturas químicas, nota-se que existe diferença entre um átomo e outro, portanto, os vértices do grafo deveriam refletir essa diferença através da utilização de $labels$, por exemplo. Entretanto, para esse trabalho decidiu-se não fazer distinções entre os átomos. Note que mesmo assim esse ainda é um problema real, pois existem estruturas químicas que são compostas somente por um único elemento, como o diamante e o grafite, que são compostos somente de Carbonos.

Depois de modelar as estruturas químicas em grafos, o problema de encontrar a maior sub-estrutura comum a duas estruturas químicas se resume ao *Maximum Common Subgraph*. Como mostrado em [6], o MSC é usado para se referir a problemas MCIS (*Maximum Common Induced Subgraph*) e MCES (*Maximum Common Edge Subgraph*). Considerando a propriedade de que S é um subgrafo induzido G se, para qualquer par de vértices u,v de S, uv é uma aresta de H se e somente se uv é uma aresta de G:

- MCIS: G_{12} é um subgrafo comum induzido de G_1 e G_2 se G_{12} é um isomorfismo induzido de G_1 e G_2. O maior G_{12} possível é o MCIS.
- MCES: Um subgrafo G_{12} com o maior número de arestas comuns a G_1 e G_2 é MCES.

Uma comparação entre o MCIS e MCES pode ser vista na Figura 2. Ao longo desse trabalho, será considerando somente o MCIS. Portanto, sempre que for citado MCS, o trabalho está se referindo na verdade, ao MCIS.

O MCS pode ser conectado, em que cada vértice está conectado a cada um dos outros vértices por pelo menos um caminho no subgrafo, ou desconectado, em que esse caminho não é exigido. Para esse trabalho, será considerado somente o MCS conectado.

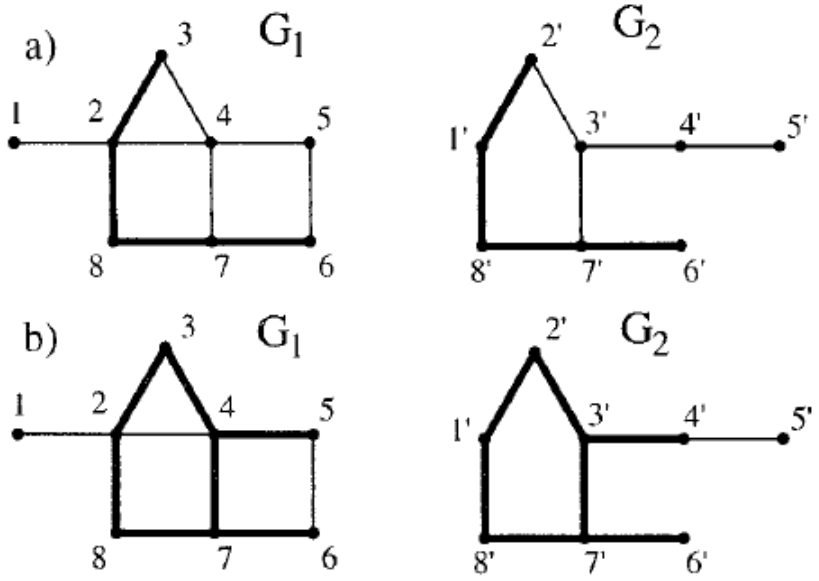

Figura 2. a) MCIS b) MCES. Essa ilustração foi retirada de [6]

2.1. *Baseline*

O *Baseline* utilizado no trabalho é o algoritmo de McGregor, um dos algoritmos mais utilizados para resolver o MCS. Esse algoritmo obtém a resposta exata e utiliza um procedimento de $backtracking$ iterativo.

Funcionamento: Considere que cada estado representa um subgrafo comum de dois grafos. Esse subgrafo comum é uma parte do maior subgrafo comum que eventualmente será encontrado. Em cada estado, um par de vértices que ainda não foi analisado (cada um dos vértices pertencente a um dos dois grafos) é selecionado e faz-se a análise se é possível estender o subgrafo comum corrente adicionando esse par obtendo um subgrafo comum maior. Feito isso, se o estado corrente não é um folha da árvore de busca, então existe pelo menos um nó pertencente ao primeiro grafo que ainda não foi selecionado e o processo iterativo contínua. Depois do novo estado ser analisado, é chamada a função de $backtrack$ para restaurar o subgrafo comum do estado anterior e escolher um novo estado diferente. Usando essa estratégia, qualquer que seja o ramo que a árvore de busca escolha, ele sempre irá seguir o mais fundo possível até uma folha ser encontrada ou uma condição de poda seja verificada. O tamanho do MCS é igual ou menor ao tamanho do menor dos dois grafos iniciais, sendo esta um exemplo de condição de poda.

Mais sobre esse algoritmo pode ser visto em [5].

2.2. Heurística Proposta

A heurística que será proposta gerará um resultado aproximado para o MCS. Para tanto, ela faz o uso da redução do problema MSC para o problema do Máximo Clique, como mostrado na subseção a seguir. As ideias heurísticas serão aplicadas ao problema do clique.

2.2.1. Redução para o Problema do Clique

Uma das abordagens existentes para resolver o problema do MCS é a redução para o problema do clique. O clique de um grafo é um subgrafo em que todos os vértices são induzidos e completos (todos os vértices se conectam através de uma aresta). Propriedades do clique:

- Todo grafo contém pelo menos um clique (de tamanho 2);
- Em um clique de tamanho K, todos os vértices tem grau pelo menos K-1;
- Se em um grafo G o grau máximo é K, então não pode existir um clique de tamanho maior que K+1;
- Se o grafo G tem um clique de tamanho K, então tem cliques de tamanho menor que K no mesmo grafo;
- Se o grafo G não tem clique de tamanho K, então pode não existem cliques de tamanho maior que K.

O clique máximo é procurado no produto modular (Matriz de Associação) H entre os grafos G_1 e G_2, que é um outro grafo. O produto modular de dois grafos G_1 e G_2 é definido no conjunto de vértices $V(G_1)xV(G_2)$ com dois vértices (u_i, v_i) e (u_j, v_j) sendo adjacentes sempre que:

- $(u_iu_j) \in e(G_1)$ e $(v_iv_j) \in e(G_2)$ ou
- $(u_iu_j) \notin e(G_1)$ e $(v_iv_j) \notin e(G_2)$

A Figura 3 mostra um exemplo de produto modular entre dois grafos. De posse da matriz H (ou grafo H) oriunda do produto modular, o problema passa a ser obter o maior clique de H, pois existe uma correspondência entre o clique de H e o maior subgrafo de G_1 e G_2 [1]. Se o maior clique encontrado em H tiver tamanho k, então o maior subgrafo entre G_1 e G_2 também terá tamanho k.

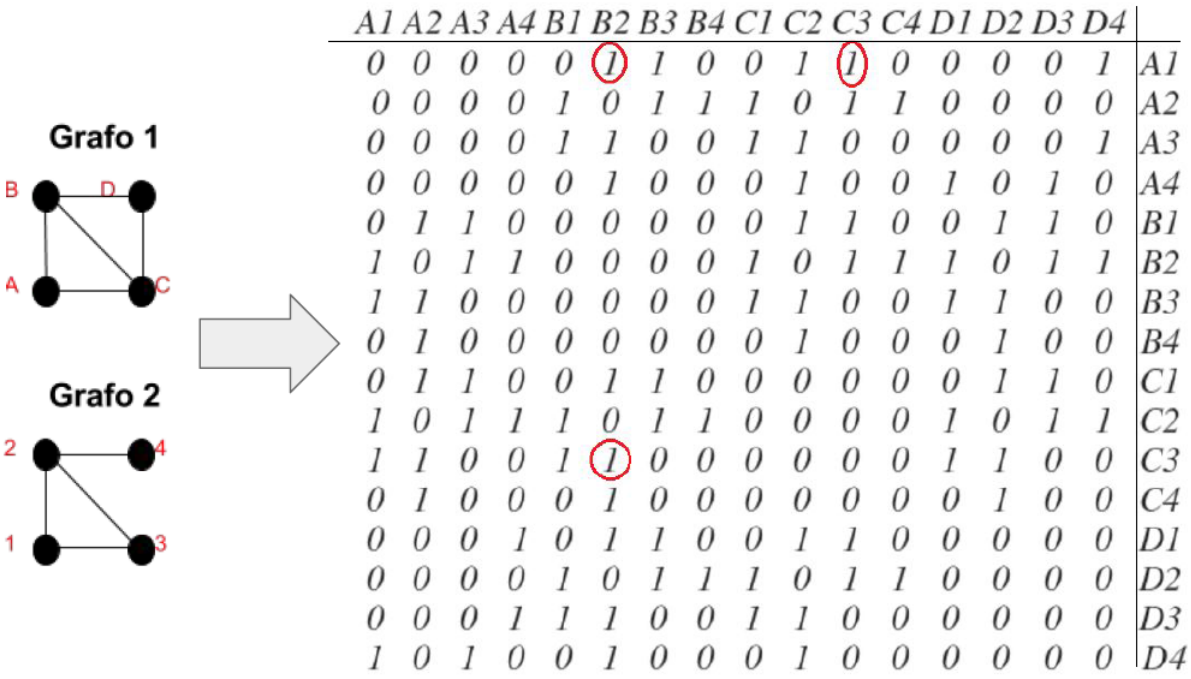

Figura 3. Exemplo de produto modular entre dois grafos. Em vermelho, está destacado um possível clique.

2.2.2. Proposta

A heurística proposta funciona da seguinte maneira:

1. Chute inicial k para o valor inicial do clique que será buscado na matriz H. O chute é baseado no valor mediano dos graus dos vértices do grafo 1;
2. Procurar no grafo H pelo k-clique;
3. Se não for achado um k-clique, faz k=k/2 e volta em 2;
4. Se for achado k-cliques:
   a. Ordena os k-cliques de acordo com a quantidade T e calcula a mediana de T;
   b. Do total de k-cliques encontrados, seleciona-se os 50% cujo T seja maior que a mediana;
   c. Desse subconjunto de k-cliques, procura-se por um clique de tamanho k+1;
   d. Se achar, volta em 4;
   e. Se não, termina com o conjunto de k-cliques;

A quantidade T é o tamanho do conjunto de vértices comuns aos k vértices do k-clique que ainda não estão no clique.

3. Análise de Complexidade

Os problemas do Maximum Common Subgraph e Maximum Clique são NP-Completos [4, 3], e portanto, não são conhecidos até o momento algoritmos que os resolva em tempo polinomial.

O MCS, por exemplo, para grafos com V_1 e V_2 vértices pode requerer até $\frac{V_1!V_2!}{(V_1-k)!(V_2-k)!k!}$ comparações para determinar todos os subgrafos com k vértices.

A seguir será mostrada a complexidade do algoritmo de McGregor usado como $baseline$ e o algoritmo com a heurística proposta, ambos para o pior caso. Para tanto, considere V_1 e V_2 como o número de vértices dos grafos G_1 e G_2 respectivamente:

Baseline

- Complexidade de Espaço: $O(V_1)$;
- Complexidade de Tempo: $O(\frac{(V_2+1)!}{(V_2-V_1+1)!})$

Heurística

- Complexidade de Espaço: $O(V_1 * V_2 + E)$
- Complexidade de Tempo: $O((V_1 * V_2)^{med(G1)} log_2(med(G1)))$,em que med(G1) é a mediana dos graus dos vértices do grafo G1.

A complexidade do $baseline$ foi calculada com base no estudo [2]. Para a complexidade da heurística, considera-se que a parte mais custosa do algoritmo é achar o primeiro conjunto de k-clique. O pior caso ocorre quando o chute inicial $k = mediana(G_1)$ tem que ser reduzida até $k = 2$. Com isso, considerou-se buscas por k-cliques com k fixo, sendo o pior caso o primeiro chute $k = mediana(G_1)$ e que essa busca será feita por no máximo $log_2(med(G_1))$ vezes(essa fórmula vem de $\frac{med(G_1)}{2^i} = 2$).

A complexidade de espaço da heurística depende do tamanho do grafo em que o clique será investigado. Esse grafo é construído a partir da matriz de associação e guardado em uma lista de adjacência. Por isso, é a quantidade de vértices ($V_1 * V_2$) mais a quantidade total de arestas (E).

4. Análise de Experimentos

Nessa seção será mostrada a análise de experimentos feita com os algoritmos. Para essas análises fez-se o uso de grafos gerados aleatoriamente no software R. Para

cada instância, foram feitas seis repetições e sua média foi tomada. A acurácia é definida como o tamanho máximo obtido pela heurística dividido pelo tamanho máximo obtido no $baseline$. Os algoritmos foram testados de duas formas: quanto a densidade e o tamanho dos grafos G_1 e G_2.

A Figura 4 mostra uma comparação entre a densidade do grafo com o tempo demandado e a acurácia. Nessas análises o tamanho dos grafos G_1 e G_2 foi fixo em 10. No primeiro gráfico, observa-se que até a densidade de 0.4, o $baseline$ é mais rápido que a heurística; entretanto, a partir desse ponto, a heurística é sempre mais rápida. Na densidade igual a 0.8 o tempo tem um aumento considerável para as duas abordagens. No segundo gráfico é possível observar que a acurácia melhora com o aumento da densidade. Esse comportamento pode ser explicado da seguinte forma: com uma densidade menor, podem existir poucos subgrafos de tamanho k e a heurística acha somente os de no máximo grau k-1; com o aumento da densidade, aumenta a quantidade de subgrafos de tamanho k e, portanto, fica mais fácil para o algoritmo achar pelo menos um desses casos.

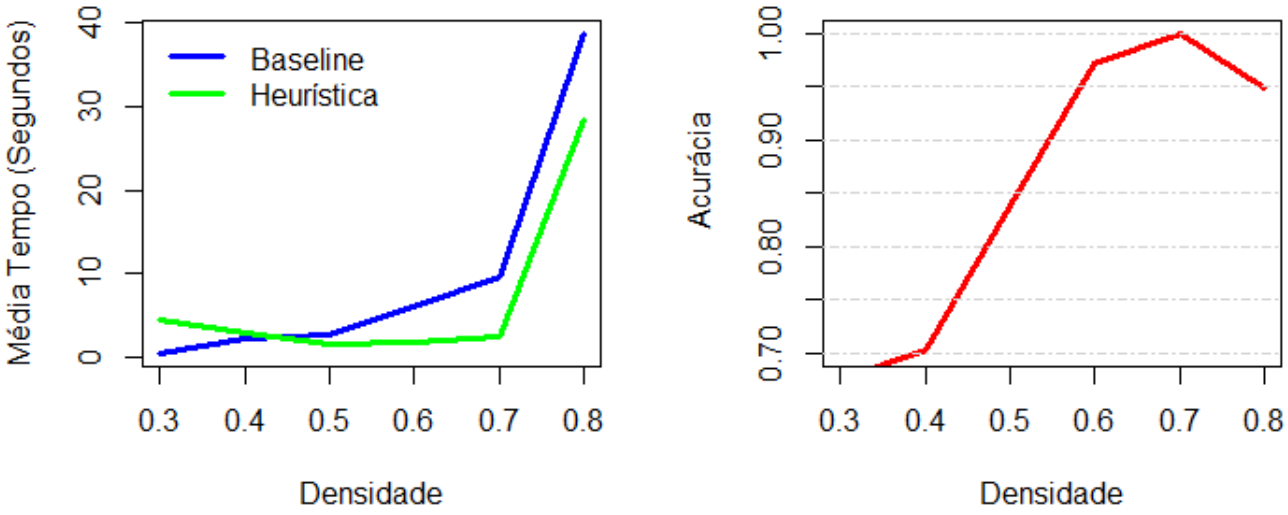

Figura 4. Comparação do desempenho dos algoritmos quanto à densidade.

Na Figura 5 tem-se a comparação do desempenho dos algoritmos quanto ao tama-

nho dos grafos G_1 e G_2. A densidade nesse caso foi fixa em 0,5. No primeiro gráfico, nota-se que em grafos de até 10 vértices, ambas as abordagens são rápidas, mas que a partir de 10 vértices, o $baseline$ passa a ter um crescimento exponencial de tempo, enquanto a heurística tem um aumento bem menor no tempo gasto. No segundo gráfico observa-se que a acurácia fica em torno de 0.8 e 0.95 para todos os tamanhos de grafos iniciais.

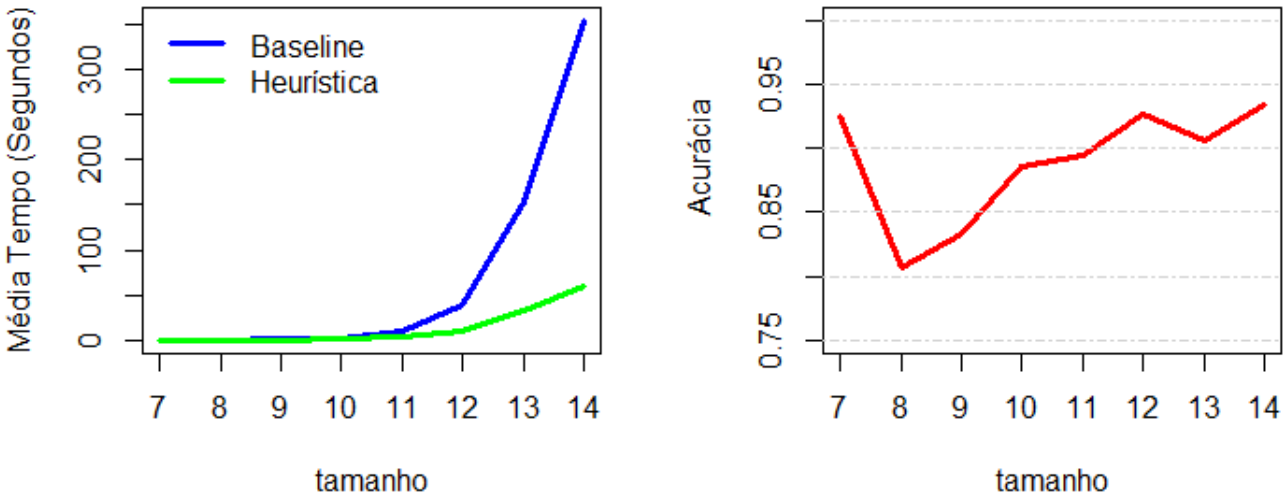

Figura 5. Comparação do desempenho dos algoritmos quanto ao tamanho.

5. Uso e Compilação

O $input$ dos dois algoritmos são dois arquivos $.txt$, em que a primeira linha tem o tamanho do grafo e as demais são a representação do grafo em uma matriz. O $output$ é o tamanho máximo dos subgrafos encontrados entre os dois grafos dados como input. Os dois algoritmos foram inplementados em C++, o $baseline$ faz o uso da biblioteca $boost$ e o compilador utilizado foi o g++.

Os testes foram feitos no notebook Samsung, com 3.7GB de memória RAM, processador Inter(R) Core(TM) i3-3110M CPU @2.400 GHz*4, com o sistema operacional Ubuntu 14.04 LTS.

6. Conclusão

O objetivo do trabalho é propor um problema cuja obtenção da solução exata seja inviável que possa ser modelado através de um grafo, e resolve-lo de duas maneiras: através de um $baseline$ e de uma heurística. Como pôde ser visto ao longo desse relatório, esse objetivo foi cumprido de maneira satisfatória. O problema proposto foi o de obter a maior sub-estrutura comum entre duas estruturas químicas, que foi transformado em grafo associando cada átomo a um vértice e cada ligação a uma arestas. Dessa forma o trabalho se tornou o *Maximum Common Subgraph*(MCS), que é um problema NP-completo. Sua inviabilidade vem do número fatorial de comparações que devem ser feitas para obter a solução exata.

O problema foi resolvido de duas maneiras, uma usando o $baseline$ que foi o algoritmo de McGregor e outra usando a heurística proposta. O algoritmo de McGregor foi escolhido por ser muito usado na literatura e por retornar um valor exato, que foi útil para comparar com a solução aproximada dada pela heurística. Entretanto, a complexidade do algoritmo de McGregor, por ser exato, tem complexidade exponencial no pior caso.

Na avaliação de experimentos foi possível ver que, mantendo o tamanho dos grafos fixos, a heurística melhora sua acurácia com o aumento da densidade; mas mantendo a densidade constante em 0.5 e variando o tamanho dos grafos, ela permanece oscilando em torno de 0.9.

De modo geral, pode-se dizer que a experiência em fazer esse trabalho foi muito proveitosa. Primeiro, por conhecer e explorar um problema que não era conhecido por mim, e segundo, por incentivar a criação ou modificação de uma heurística. Essa segunda parte é relevante pois em algum momento de nossa vida acadêmica ou profissional podemos nos deparar com algum problema NP-completo e ter essa experiência prévia de como explora-lo através de heurísticas pode ser muito útil.

Referências

[1] M. M. Cone, R. Venkataraghavan, and F. W. McLafferty. Computer-aided interpretation of mass spectra. 20. molecular structure comparison program for the identification of maximal common substructures. *Journal of the American Chemical Society*, 99(23):7668–7671, 1977.

[2] D. Conte, P. Foggia, and M. Vento. Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs. *J. Graph Algorithms Appl.*, 11(1):99–143, 2007.

[3] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 29. wh freeman, 2002.

[4] V. Kann. On the approximability of the maximum common subgraph problem. In *STACS 92*, pages 375–388. Springer, 1992.

[5] J. J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software: Practice and Experience*, 12(1):23–34, 1982.

[6] J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of computer-aided molecular design*, 16(7):521–533, 2002.