

1 Introduction

A table of frequently used symbols:

Symbol	Representation
g	the input graph
v	the set of nodes in g
e	the set of edges in g
n	a node in v
G	the constructed metagraph
V	the set of metanodes in G
E	the set of metaedges in G
N	a metanode in V of the form $\{n_i : x_i\}$ where $n_i \in v$ and x_i is the number of atoms at n_i

Each $N \in V$ is composed of i many nodes from v . A metanode takes the form $\{n_i : x_i\}$ where $n_i \in v$ and x_i is the number of atoms at n_i . The $\sum_0^i x_i = k$ at any given N ensuring that all atoms are accounted for at each metanode

2 Algorithms

Algorithm 1: ConstructMetaGraph

Input: $g = (v, e)$: the input graph, $\{s, t\} \in v$: the start and target compounds, k : flow/number of atoms to conserve

Output: $G = (V, E)$: the meta-graph

```

1  $MG \leftarrow (MV = \emptyset, ME = \emptyset)$  /* initialize new metagraph */
2  $stack \leftarrow$  initialize empty stack;
3  $start \leftarrow \{s : k\}$  /* Create a metanode with all  $k$  atoms at the start compound */
4  $V \leftarrow V \cup start$  /* Add the start state to the set of metanodes */
5  $stack.push(start)$  /* Add the start state to the stack to find its neighbors */
6  $G \leftarrow \text{PopulateMetaGraph}(G, stack, target)$  /* Find neighboring metanodes to build  $G$  */
7 return  $G$ 
```

Algorithm 2: PopulateMetaGraph

Input: $G = (V, E)$: the metagraph, $stack$: stack of metanodes that need to be explored, $target$: the metanode state with all k atoms at node t

Output: $G = (V, E)$: metagraph with any newly found metanodes added in

```

1 while  $|stack| > 0$  do
2    $current \leftarrow stack.pop()$  /* Pop off a metanode to explore */
3   if  $current = target$  then
4     continue /* If we've reached the target, no need to find nbrs */
5   else
6      $nbrCount \leftarrow \text{IterativeFindMetaNbrs}(current)$ ;
7     if  $nbrCount = 0$  and  $current \neq target$  then
8        $G \leftarrow \text{Prune}(current)$  /* This metanode is a terminus, so remove it from the metagraph */
9 return  $G$ 
```

Algorithm 3: IterativeFindMetaNbrs

Input: *parent*: the metanode to find nbrs for

Output: $G = (V, E)$: metagraph with any newly found metanodes added in

- 1 for each inner node n in N , for inner nbr m , move 0 to xi flow. For each move, attempt to move the remaining flow at n to the remaining inner nbrs m . Then repeat for each n in N . Check if any of the newly generated states are valid and add them to MG and stack
-