CSM Group Separation Algorithm

The CSM program enumerates over permutations during calculation. These are atom permutations – of atoms in the molecule. Instead of going over all n! permutations, CSM divides the atoms into groups of interchangable atoms, going over permutations of each group, instead of the entire n! space.

The separation is done by atom type (if atom types are considreded) and the connectivity of each atom. “Connectivity” is defined recursively – two atoms are equal if the are of the same type, have the same number of neighbors, and all their neighbors are equal. So if A is connected to B and C, and D is connected to E and F, A and B are equal iff B is equal to E or F, and C is equal to F or E. If D is connected to E, F and G, it is not equal to A.

CSM implements an iterative algorithm to find these groups. The algoritm is implemented in the Molecule class, and uses several data structures:

* symbol[i] – Atom i's chemical symbol (C, O, etc...)
* adjacent[i][j] – The jth neighbor of Atom i.
* valency[i] – The number of neighbors for Atom i.
* similar[i] – The group number for Atom i. Items from the same group will have the same number here.
* marked[i] – Used during the iterations – whether the ith atom has been visited this iteration.

The first iteration builds the first grouping of atoms. Each iteration then refines this grouping. The first iteration is as follows:

* Atoms with the same type and same number of neighbors are put in the same group.

At the end of the first iteration, the similar[] array contains these preliminary groups – similar0[i]==similar0[j] iff i and j have the same symbol and same number of neighbors. Here similar0 means similar atoms after the 0th iteration.

This is done in O(N2) worst-case.

For the following iterations, we need to define similari[] - the similar atoms after the ith iteration. This is done in the isSimilar function:

* Two atoms A and B are similari iff each of atom A's neighbors has a similari-1 atom B neighbor.

So each of the iterations after the first looks like this:

* Go over all groups.
* In each group, compare the first element of each group to all the other elements in the group.
* All the elements that are not similari to the first element of the group are split into a new group.

The CSM implementation performs 200 iterations and then stops. It doesn't stop if an iteration did not result in any change, nor does it continue if there are still groups to be divided.

Each iteration calls O(N) times to isSimilar, which again takes O(N) time, giving the total algorithm a worst-case complexity of O(N3), although stopping after 200 iterations means the actual complexity is never worse then O(N2) (at a price of not performing the exact division).

Modifications

Since the number of atoms can be in the thousands, O(N3) can be too much. We want to reduce it.

The first iteration is trivially reduced to O(N) by using a hash table to map <type, #neighbors> to atoms. This by itself does not improve the asymptotic complexity of the algorithm.

IsSimilar (the function that helps build similari based on similari-1 runs in O(N2), too (where N is the number of neighbors of the two atoms). By changing the data structure of similar from a simple array to an array providing the group number of each atom and a hash-based set containing all the atoms of the group, *we can improve this to O(N), reducing the overall complexity to O(N2) without limiting the division to 200 iterations.*