Side-Chain Modelling

Niclas Theodorsson (nicthe@student.chalmers.se)

19th December 2019

1 Introduction

This report intends to explain different types of algorithms presented in structural bioinformatics in order to find the most probable side-chain conformation in a protein. Finding this combination is potentially an NP-complete problem as there are several angles, those on the backbone of the protein chain that determine the fold of the main-chain, and those of the side-chain. Since angles are continuous, they must be discretized but the number of possible combinations quickly becomes too vast to feasibly search though all combinations (commonly used is a stride of 10-15 degrees). Note that changing one angle will affect the other angles, as side-chain positions affect the fold stability of a protein. In order to reduce the number of possible combinations, different algorithms take different approaches, for instance finding energy maximas and minimas and/or using rotamer libraries which are collections of commonly occuring low-energy side-chain conformations. Using such libraries greatly reduces the number of combinations. This reports looks at two different studies: One paper introducing the famous dead-end elimination theorem (DEE), and one paper extending an already existing program, SCWRL2.95 that implements a similar algorithm to DEE, using a graph partitioning algorithm (SCWRL3.0)

2 Summary and analysis of studies

The dead-end elimination theorem presented in *The dead-end elimination theorem and its use in protein side-chain positioning* by Desmet et al. starts by finding the potential energy of a protein system as well as the global minimum energy conformation (GMEC) based on the rotamers of the library used. The authors then posits that, if a pair of rotamers (i_a, i_b) satisfies the inequality

$$E(i_a) + \sum_j \min_s E(i_a, j_s) > E(i_b) + \sum_j \max_s E(i_b, j_s)$$

where s if the number of rotamers in the library, then the rotamer i_a is not compatible with the GMEC and is marked as dead-ending. After this the authors derive a generalized version of DEE for pairs of rotamers. This is beneficial because, if all pairs of a rotamer are marked as dead-ending, the rotamer will also eventually be marked as dead-ending. Finding pairs that are dead-ending also reduces the search space of $\sum_j \min_s E(i_a, j_s)$.

When the program runs, in the first iteration it will first look through all single rotamers and mark those that do not satisfy the inequality as 'dead-ending', then a generalized version of DEE looks through pairs of rotamers and does the same for reasons mentioned above. This continues until there are no more dead-ends. The authors gives an example where the number of rotamer combinations were reduced from $2.7 * 10^{76}$ to 7200 when running the program on an insulin dimer. The authors also reduce the number of computations by removing "clearly incompatible" rotamers using a 30 kcal mol^{-1} threshold.

After finding all rotamer combinations (23), the lowest energy conformation was said to be the most apt combination, which was verified by comparing the generated structure with the X-ray-determined structure. Using this validation method, they found that 55/76 (72%) residues were correctly determined.

In *A graph-theory algorithm for rapid protein side-chain prediction* by Canutescu et al. the authors builds upon an already existing program, SCWRL2.95, that works as follows: side-chains are placed on the backbone according to the probabilities (given by the rotamer library, in this paper only probabilites $> 90\%$). If a side-chain clashes with another side-chain, they are marked as active, and then placed on every possible rotamer. If a rotamer clashes with a non-active residue, that residue also becomes active. All active residues are then not in their GMEC, while non-active residues are. This means that SCWRL2.95 is practically implementing a dead-end elimination. Similarily to the original DEE theorem, SCWRL2.95 removes "clearly incompatible" rotamers using a 5 kcal mol^{-1} threshold but also only considers rotamer over a 10 kcal mol^{-1} threshold to be interacting with eachother. The authors concludes from this that "SCWRL does not locate the global energy minimum of its own energy function, because many interactions are ignored when setting up the combinatorial searches to make the calculations tractable.".

To expand on this program, the authors interprets this algorithm as a graph, where each residue is a vertex in an undirected graph. If one rotamer in a residue interacts with a rotamer in another residue, an edge is created between the vertexes. This will create a graph of smaller clusters. If a cluster is too large to be feasibly computed, the algorithm will find a residue that, when cut away from one of the clusters, will divide the cluster into two smaller clusters, until the cluster has become a collection of adequately smaller clusters. The authors modify this existing program so that, instead of recursively breaking up the graph into pieces, they break up clusters into biconnected components instead.

By finding and removing vertexes that are "articulation points"(vertexes that appear in more than one biconnected clusters) by using a depth-first search, they can then always create single biconnected subgraphs. An important difference between the original SCWRL2.95 and SCWRL3 by the authors is that each articulation point holds the minimum energy over "all combinations of rotamers of the residues in the component for each rotamer of the articulation point.". These things imply two important things, namely that (1) The accuracy of SCWRL3.0 is potentially better because there is no threshold removing retomers with high energy that *might* be a part of the GMEC and (2) when calculating the GMEC, the complexity is reduced to the largest biconnected subgraph in the cluster. It is worth noting that the authors still apply DEE to remove most obvious incompatibilities after reading in the coordinate data.

In order to compute the minimum energy of each biconnected subgraph, the authors uses a backtracking algorithm that originally searches through all possible combinations in a tree. Some truning techniques are then deployed, such as using a bounding function or sorting the rotamers by their self-energy, which reduces the number of possible combinations when finding the GMEC of each biconnected subgraph. To show the results, the authors run both SCRWL2.95 and SCRWL3 on a set of 180 proteins, showing a slightly improved accuracy of SCRWL3 over SCRWL2.95. When it comes to speed however, the SCRWL3 shows substantial improvement over SCRWL2.95, resulting in atleast 4 times speedup and up to 38 times speedup for different threshold configurations in SCRWL2.95.

As we can see, the dead-end theorem is practically used both in SCRWL2.95 and SCRWL3, but in different ways. In the former case, it is used as a basis for finding the most probable side-chain conformation, and in the later, it is only used to remove the most obviously incompatible conformations.

Since the biconnected subgraphs in SCRWL3 are causally isolated by definition, SCRWL3 inherently enables more parallell computing techniques over SCRWL2.95. For example, there are no data (energy values) dependencies between subgraphs so finding GMEC of each subgraph can be seen as a task dispatched to different threads, enabling computation of all subgraphs at once (with a summation by reduction at the end). As the size (number of vertexes) of the subgraphs are available, one could even dispatch larger subgraphs to better and faster cores, or even cores with architecture dedicated to solving such a problem. Even if such techniques do not reduce the time complexity of solving the problem, it can be useful for increasing speed *and/or* accuracy, as things such as energy or probability thresholds can be further lowered for increased accuracy, or kept the same for additional speedup.