Project description
Core Construction

Advisors: Sebastian Will, Martin Mann

The task in this project is to write a constraint-program that solves the combinatorial problem of constructing all sets of n points in the cubic lattice with a given "compactness". We are interested in maximally compact sets as well as nearly compact sets. Compactness is measured via pairs of points in unit distance. The program should use the C++ constraint-library Gecode. The code shall be documented for the use with the documentation tool doxygen.

1 Background

Compact point sets are the central ingredient of exact structure prediction in HP-lattice models of proteins. There, we want to predict structures with minimal energy. Using compact point sets, one can map a protein sequence to a compact set in order to obtain a structure which has the given point set as its hydrophobic core. The structures on the most compact core, where the sequence fits, will have minimal energy.

2 Formal specs

A contact (p,q) is a pair of points in unit distance. Given a number of elements n and a number of contacts c, the core construction problem is to enumerate, up to translation, rotation, and reflection symmetry, the set of all point sets P subset of Z^3 (i.e., the cubic lattice) such that the cardinality of P is n and the number of contacts, i.e. the cardinality of $\{\{p,q\}|p,q \in P$ and (p,q) forms a contact$\}$, is c.

We will look at a slightly modified version where, for additional constraints, one is given a set of number sequences S. There, a number sequence is a finite sequence of integers $n_1, ..., n_k$. The idea of number sequences is that we can slice a point set into layers along each of the dimensions x,y, or z, i.e. formally for dimension x we look at the sets $\{(p_x, p_y, p_z) \in P|p_x = i\}$. By this we get a sequence of such sets starting from the minimal i where the corresponding set is not empty to the maximal such i. Taking only the cardinalities, we get the number sequence of P in the dimension x. For y and z, number sequence is defined analogously. (Since we are only interested in connected cores, there will be no zeros in number sequences.)

Then the modified problem is to predict the set of all point sets P of size n with c contacts, where additionally the number sequence of P in each dimension x, y, and z is in S. The modified problem is easier than the original problem since the number sequences add strong constraints.

3 Modelling hints

If one knows the lengths of the number sequence in each dimension, this fixes a surrounding cube of the points that can belong to the point set. A good strategy is therefore to first enumerate the lengths and then construct your constraint model. Then, one can model each point within the surrounding cube by a boolean variable that is one iff the point belongs to the "compact" point set.

Start without breaking the rotational and reflection symmetries. (Then ask for further advice.) For breaking translation, one can fix the origin of the surrounding cube to (0,0,0).

If one has more than one number sequence for a dimension, this translates to a disjunction. It is important to infer the constraints on the number of elements in the single layers from the disjunction over number sequences.

4 Input/Output format

Input: the program has arguments for set size n and number of contacts c, additionally it reads number sequences from a file that describes the set S. The file has the format

```
n_11 n_12 ... n_1k(1)
n_21 n_22 ... n_2k(2)
.
.
n_l1 nl2 ... n_lk(l)
```

where the n_{ik} are integers, $n_{i1}, ..., n_{ik}(i)$ is the ith number sequence in the set S. Implicitely, S contains also all reverted sequences of the listed ones. (In a later version, the program could compute a set of sequences for n and c.)

Output: the program writes the set of cores in the following format: for each core one entry that is introduced by the line

```
#COREDATA
```

The header line is followed by a lines that each contain the triple of coordinates of one point

```
 x y z
```

Example input is:

```
n=16, c=28
4 4 4 4
4 6 6
8 8
```

or

```
n=16, c=27
5 5 6
5 6 5
7 9
```

Example output is (size 10, 14 contacts)

```
#COREDATA
 0 0 1
 0 1 0
 0 1 1
 0 1 2
 1 0 0
 1 0 1
 1 0 2
 1 1 0
 1 1 1
 1 1 2

 #COREDATA
 0 0 0
 0 0 1
 0 1 1
 0 1 2
 1 0 0
 1 0 1
 1 0 2
 1 1 0
 1 1 1
 1 1 2

.
.
.
```