The Meßthaler-Wulff Project

Julia Meßthaler

Blazingly fast code for finding all crystals (subsets of a graph) that can be constructed using only transformations that locally minimize surface energy.

The Problem

Let (G, E) be some graph and $\eta : G \to \wp(G)$ denote the neighbors of a given node, defined as

$$\eta(n) \coloneqq \{n_0 \in G \mid \{n_0, n\} \in E\}$$

Now we can define a crystal as $c \subset G$ with c finite. This allows us to define its complement $\bar{c} \coloneqq G \setminus c$ and the set of all crystals $C \coloneqq \{c \subset G \mid c \text{ finite}\}$.

Now we can define the surface energy of the crystal c (or arbitrary subsets of G) as

$$\xi_c \coloneqq \sum_{n \in c} f_{\bar{c}}(n)$$

where $f_M(n)$ denotes the "friendliness" of the node n or how many friends it has 4defined as

$$f_M(n) \coloneqq \#\{n_0 \in \eta(n) \mid n_0 \in M4\}$$

The idea now is to find crystals c such that $\frac{\xi_c}{\#c}$ is optimal.

The Crystal Graph

We can impose a graph structure on C with the edges

$$T \coloneqq \{\{c, c \setminus \{n\}\} \mid c \in C \text{ and } n \in c\}$$

we call these the transformations and (C, T) the transformation graph.

If we want to discover optimal crystals, then we must efficiently walk this graph. Since this graph is very dense and very large, we must first discuss some optimizations:

- $O(1)$ transformations
- Pruning bad transformations
- Exploiting Symmetries
- Pruning bad crystals

Efficient Transformations

In the code this is achieved using the stateful class `AdditiveSimulation` that walks along the transformation graph. This class keeps track of two important properties, namely $f_c(n)$ and $m(n)$ for (almost) all $n \in G$, in practice $f_c(n)$ will default to 0 and $m(n)$ to 1. $m(n) \in \{0, 1\}$ is called the mode of the node n and is defined as

$$m(n) \coloneqq \begin{cases} 0 & \text{if } n \in c \\ 1 & \text{if } n \in \bar{c} \end{cases}$$

We call 1 the forwards mode and 0 the backwards mode, as nodes that we might add to our current crystal will be in $\bar{c}$ and nodes we might remove are in c.

We can also define the mode sign of a node as $s(n) \coloneqq 2 \cdot m(n) - 1$.

Now let n be the node that defines our transformation, we can update $m(n)$ like so $m'(n) = 1 - m(n)$. The friendliness $f_c(n)$ does not change. Instead, the friendliness of each neighbor must be updated. So for each $n_0 \in \eta(n)$: $f'_c(n_0) = f_c(n_0) + s(n)$. We can also define an update rule for the energy $\xi'_c = \xi_c + s(n) \cdot f_{\bar{c}}(n) - s(n) \cdot f_c(n)$ todo. **TODO: Explanation**

Pruning Bad Transformations

Most transformations are pretty bad, for example in a solid crystal, removing a node in the center will only increase the energy, to combat this we will only consider locally optimal transformations. For this we define $\delta(n) \coloneqq \xi_{c'} - \xi_c = s(n) \cdot f_{\bar{c}}(n)$ todo, the energy difference for this transformation.

Now we look for nodes n such that no nodes n_0 exist with $m(n_0) = m(n)$ and $\delta(n_0) < \delta(n)$. These are the locally optimal transformations.

This is achieved in $O(1)$ using the `PriorityStack` class, a priority queue optimized for this specific use case. We have two instances for our graph walking `AdditiveSimulation`, one for backwards mode and one for forwards mode. Depending on which mode is then queried, we query the appropriate instance for all nodes that have minimal $\delta(n)$.

Exploiting Symmetries

Let H be a group action on G. This is only useful to us if the action commutes with the neighborhood function as in $\eta(h(n)) = h(\eta(n))$ for all $h \in H$. This not only means that the energy is also invariant under H, but also the possible next locally optimal transformations, meaning we can run our simulation only on canonical representatives of each equivalence class of G/H.

We call a function $\chi_H : G \to H$ a characteristic for the group action H if for all $h \in H$ and $g \in G$

$$\chi_H(h(g)) = h \circ \chi_H(g)$$

Pruning Bad Crystals