Minimum Fill-In

Mikkel Gaub,
Malthe Ettrup Kirkbro,
& Mads Frederik Madsen

June 2, 2017

1 Algorithm

The algorithm REFREF is concerned with finding four-cycles and certain substructures in the graph, called moplexes, which are defined as adhering to the following rules:

Clique Every vertex in the moplex must know all other vertices.

Shared neighbourhood Every vertex in the moplex must have the same neighbourhood.

Maximally inclusive If a vertex can be added to a moplex, without it not being a moplex anymore, that vertex must be included.

Furthermore, simplicial moplexes are moplexes where the neighbourhood of the moplex is also a clique.
MARKINGS

1.1 Kernelization

The kernelization algorithm used, REFREF, is split into three phases:

1. .
2. .
3. .

Since phase 3 is dependent on k, it has to be run for each k-value used by the algorithm...

1.2 Cases

Once the graph has been kernelized, the algorithm is called with an initial k-value. The algorithm then tries to find a solution for the graph with the initial k-value. If no solution with that k-value can be found, k is incremented and the algorithm is run again. The algorithm is recursively called with a the k-value and in each iteration the first of six actions are performed, where the criteria of that case is met. These cases are, in order:

Four-cycle If a four-cycle exists, the program branches on both possible resolution of the four-cycle.

Moplex with marked and unmarked vertices If a moplex containing marked and unmarked vertices exists in the graph, all vertices of the moplex are marked.

Simplicial moplex with only unmarked edges If *any* simplicial moplexes containing only unmarked edges, exist in the graph, they are removed.

Unmarked moplex with a neighbourhood missing only one edge If a moplex containing only unmarked vertices and has a neighbourhood which is only missing one edge in order to be a clique, the missing edge is added.

Only marked moplexes If all moplexes in the graph are marked, the algorithm fails.

Any unmarked moplex If there are any unmarked moplexes in the graph, the program branches into two. In one of the cases, every vertex in the moplex is marked. In the other case, every edge missing between the moplex and its neighbourhood is added.

2 Optimizations

The algorithm has been implemented in C++ to allow for greater optimization.

2.1 Moplex caching

A very large part of the algorithm is in locating every moplex in the graph, since this is used in every case of the algorithm, excluding the case where a four-cycle is found. Because the moplexes change fairly little in each iteration of the algorithm, these are cached in order to reduce the amount of time wasted on finding already found moplexes.

It is accomplished by...

2.2 Component splitting

Since the graph is possibly divided into smaller kernels by the kernelization algorithm, the problem can potentially reduced to several instances with a lower k each, the performance of the program should greatly increase.

This is accomplished by running the core of the algorithm on each found component of the graph, with an initial k-value of 0, or the k returned by the kernelization, if only one component exists in the graph.

2.3 Subgraphs

Many functions used by the algorithm requires subgraphs to be considered, such as the kernelization and the deletion of simplicial moplexes. Since the algorithm is depth-first, meaning each branch is explored exhaustively before the next branch is considered, the changes made must be easily reversible. Since copying or deleting and recreating the graph is expensive, a cheap way to specificy which parts of the graph are currently "active" would be useful. This is a accomplished with a boolean value for each vertex, indicating whether or not it should be regarded.