Submission Track B: exact algorithm for Minimum Fill-In (aka Chordal Completion)

Édouard Bonnet, R. B. Sandeep, and Florian Sikora

May 25, 2017

1 Description of the algorithm

Our implementation follows the exact algorithm of Fomin et al. [3] which is based on listing the minimal separators, the potential maximal cliques, and then performing dynamic programming based on those structures [1, 2]. The algorithm by Fomin et al. refines the approach of Bouchitté and Todinca on some minor aspects (only considering inclusion-minimal minimal separators, trying only the so-called *full blocks* associated to a minimal separator) and also lists the potential maximal cliques in a different way. We realized that this theoretically better enumeration of potential maximal cliques was in practice slower, so for this part we follow Bouchitté and Todinca's algorithm.

This algorithm can be seen as FPT parameterized by the number of potential maximal cliques. Its dependency in the parameter is polynomial (even linear) which does not imply that the problem is polytime solvable since the number of potential maximal cliques (and minimal separators) can be as big as exponential in the number of vertices of the graph.

It is noteworthy that this nice and deep theory (of potential maximal cliques) initiated by Bouchitté and Todinca seems to perform, as is, better than more straightforward approaches (such as the naive branching on all chordalizations of a long induced cycle). This is not always the case and more often than not the intricate and theoretically best algorithms perform quite poorly in practice. This sets Bouchitté and Todinca's algorithm is the regarded class of theoretical work with a direct practical value.

Preprocessing. We preprocess the graph by removing iteratively the simplicial and universal vertices. Also, if X is a clique whose removal separates the graph into at least two connected components $C_1, \ldots, C_\ell$, one could solve independently the graph induced by $X \cup C_i$ (for $i \in [1, \ell]$). The graph $X \cup C_i$ can in turn have a clique separator. Let us call *atoms* the graphs that we obtain by applying this simplification until no clique separator exists. It turns out that the union of atoms is a kernel. One can find the atoms due to a classical algorithm by Tarjan computing all the clique separators [4]. Although, we finally dropped the computation of clique separators from our implementation. The reason for

that is that it was taking some significant amount of time to often report that there is no clique separator. Instead, we are just computing the biconnected components (which can be seen as a coarser notion of atoms).

Listing minimal separators.
We list the minimal separators for each component following Berry et al. [5].

Listing potential maximal cliques. We list the potential maximal cliques (PMCs) based on the list of minimal separators following Bouchitté and Todinca [2].

Dynamic programming based on minimal separators and PMCs. For that part, we rely on the algorithm of Fomin et al. [3]. We compute an upper bound of the solution to prune some tests of potential maximal cliques if the number of fill edges they incur exceeds our upper bound. The upper bound is based on the best of four heuristics preceded by one additional heuristic based on what we call *diamond configurations*. We now describe those heuristics.

Upper bound heuristics. We compute the number of fill edges with the *minimum degree heuristic*: take a minimum degree vertex, fill its neighborhood and remove it from the graph, and the *good neighborhood heuristic*: take a vertex with the least number of non-edges in its neighborhood, fill its neighborhood, and remove it from the graph. We also compute two other upper bounds: one is the *minimum degree heuristic* where we break ties with the *good neighborhood heuristic*, the other is the *good neighborhood heuristic* where we break ties with the *minimum degree heuristic*.

Diamond configurations. We precede the computation of the best upper bound among the four previous heuristics by a shared fifth heuristic. By *shared* we mean that the fill edges obtained at this step will be added at the starting point of each of the other four heuristics. A *diamond configuration of value f* is two non-adjacent vertices u and v with f non-edges in their common neighborhood $N(u) \cap N(v)$. Observe that if uv is *not* a fill edge then at least f edges should be added to chordalize the graph: $N(u) \cap N(v)$ should be cliquified. Another way to see it, is that if f is large, then probably uv should be a fill edge. The heuristic is to add uv when f is larger than some threshold (empirically, we set this threshold to 50). This heuristic is only partial and can immediately stop (if there are no induced C_4 in the graph for instance). Although, in some open instances it places some edges which are very likely to be in an optimum solution.

2 Comments to the organizers

We thank the organizers for their flexibility and for offering us good working conditions and an exciting second edition of PACE. We find that the new problem of this year (Minimum Fill-In) was particularly well chosen, and we wish to congratulate the program and the steering committees for that.

As a soft suggestion for next year, we think it is interesting to have an exact *and* a heuristic contests for all the problems. We believe that it does not add too much work for the organizers. It might even be interesting to have the same set of instances for the exact and heuristic tracks (with less allowed computing time for the heuristics), to compare the approaches and analyze on which types of instances the heuristics are close to optimality. For instance, our upper bound (described above) is giving optimum or very close to optimum values on many instances of the open set. We could not manage to capitalize on that (with say, a branch and bound when Fomin et al.'s algorithm is not quick enough) since good lower bounds are hard to get.

References

[1] Bouchitté, V., Todinca, I. Treewidth and minimum fill-in: Grouping the minimal separators. SIAM Journal on Computing, 31(1), 212-232 (2001).

[2] Bouchitté, V., Todinca, I. Listing all potential maximal cliques of a graph. Theoretical Computer Science, 276(1), 17-32 (2002).

[3] Fomin, F. V., Kratsch, D., Todinca, I., Villanger, Y. Exact algorithms for treewidth and minimum fill-in. SIAM Journal on Computing, 38(3), 1058-1079, (2008).

[4] Tarjan, R. E. Decomposition by clique separators. Discrete mathematics, 55(2), 221-232 (1985).

[5] Berry, A., Bordat, J. P., Cogis, O. Generating all the minimal separators of a graph. International Journal of Foundations of Computer Science, 11(03), 397-403 (2000).