Project proposal for CS260:
the vertex cover problem

Osayd Abdu (142461), Abdulelah Alneghaimish (159296),
Lukas Larisch (154273), Elaf Islam (142724)

September 18, 2017

1 Introduction

Our course project will cover the vertex cover problem, a problem included in Karp's list of 21 NP-complete problems [1, 2]. We will implement two algorithms for finding a minimal vertex cover on undirected graphs: The first algorithm with exponential running time uses a reduction from the max clique problem [8], the second algorithm solves the problem with help of tree decompositions [4, 5, 6], which makes it possible to experimentally examine the constants hidded in Courcelle's theorem (1990):

Theorem. *Every graph property definable in monadic second-order logic can be decided in linear time on graphs of bounded treewidth.*

Hence, we will test our implementations on graphs of bounded treewidth and random k-trees graphs, as well on general graphs generated by the GNP model and e.g. "named" graphs as included in the SageMath package [7].

2 Problem description

In the following we use standard graph terminology [3].

Definition. *Given a (undirected) graph $G = (V, E)$, a* vertex cover *C of G satisfies the following conditions:*

- $C \subseteq V(G)$
- $\{u, v\} \in E(G) \implies u \in C \vee v \in C$

A minimum vertex cover *is a vertex cover of minimum size among all vertex covers of G.*

Hence, a vertex cover C of a graph G is a subset of the vertices of G, such that all edges of G have an endpoint in C.

3 The algorithms

3.1 Algorithm 1

The first algorithm solves the minimum vertex cover problem using a solution for the maximum clique problem.

Definition. *A* clique C *of a graph* $G = (V, E)$ *satisfies the following conditions:*

- $C \subseteq V(G)$
- $u, v \in C \implies \{u, v\} \in E(G)$

A maximum clique *is a clique of maximum size among all cliques in* G*.*

In other words, a clique C in a graph G induces a complete subgraph in G.

We are going to use the reduction from the vertex cover problem to the maximum clique problem since both problems are NP-complete [8].

The algorithm for solving the maximum clique problem in a graph G that Östergård suggested in 2002 starts from a vertex v_i, $i = n, \ldots, 1$, $n := |V(G)|$, and builds the largest clique that includes v_i from the set of vertices $S_i = \{v_i, ..., v_n\}$. So, $S_n := \{v_n\}$ is considered first (clearly, S_n is the largest clique itself). The algorithm then considers the largest clique in S_{n-1} that contains v_{n-1} and so on. The added advantage of this algorithm is the pruning strategy it uses to reduce the number of possible cliques.

3.2 Algorithm 2

The second algorithm solves the minimum vertex cover problem with help of tree decompositions.

Definition. *A* tree decomposition *of a graph* G *is a pair* (T, β)*, where* T *is a tree and* $\beta : V(T) \to \mathcal{P}(V(G))$ *is a map such that*

(T1) for every edge $e \in E(G)$ *there is a node* $t \in V(T)$ *with* $e \subseteq \beta(t)$*, and*

(T2) for all $v \in V(G)$ *the set* $\beta^{-1}(v) := \{t \in V(T) : v \in \beta(t)\}$ *is non-empty and connected in* T*.*

We refer to the sets $\beta(t)$ *of a tree decomposition as* bags*. The width of a tree decomposition is it's maximal bag size minus 1. The* treewidth *of* G*,* $tw(G)$*, is defined as the minimum width over all tree decompositions of* G*.*

The algorithm is an extension of the approach for solving the VC-problem on trees, i.e. a two-phase dynamic programming algorithm that, now on a treedecomposition, first creates tables for the power set of the vertices contained in a bag. This is initially done for bags corresponding to leaves of a tree decomposition. Then, all vertex covers listed in the created tables will be extended, if possible, to vertex covers of a larger subgraph of the original graph in a bottom-up manner until the root of the tree decomposition is reached. In the second phase, an optimal vertex cover is computed by a top-down run beginning on the root of the tree decomposition which examines the computed tables until a leaf is reached.
The algorithm will work on *nice tree decompositions*, i.e. rooted TD's with at most two children per node and the feature that for nodes of degree 2, one bag is a subset of the "adjacent" bag and the "adjacent" bag contains exactly one more node of the graph.
For such type of TD's it is sufficient to state four rules on how to compute the tables within the algorithm. These four distinct rules are applicable on different types of nodes of a nice treedecomposition, e.g. leafs or nodes with two children (join nodes).
The algorithm has running time $\mathcal{O}(2^{tw(G)+1} \cdot n^{O(1)})$, hence is a polynomial time algorithm for graphs of bounded treewidth.

References

[1] Karp, R. M. (1972). Reducibility Among Combinatorial Problems.. In R. E. Miller & J. W. Thatcher (eds.), Complexity of Computer Computations (p./pp. 85-103), : Plenum Press, New York. ISBN: 0-306-30707-3

[2] The vertex cover problem, `https://en.wikipedia.org/wiki/Vertex_cover`

[3] Diestel, R. (2005). Graph Theory. Springer-Verlag Heidelberg, New York.

[4] H. L. Bodlaender, A tourist guide through treewidth, Acta Cybernetica

[5] Neil Robertson and P.D Seymour (1991), Graph minors. I. Excluding a forest, Journal of Combinatorial Theory, Series B, pp 39 - 61

[6] Stefan Arnborg (1985), Efficient Algorithms for Combinatorial Problems with Bounded Decomposability, BIT, pp 2–23

[7] SageMath, `http://www.sagemath.org/`

[8] Patric R.J. Östergård (2002), A fast algorithm for the maximum clique problem. Discrete Applied Mathematics