midterm report for CS260:
the vertex cover problem

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

date: ?

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:*

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

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. 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 the leafs is reached.
The algorithm will work on *nice tree decompositions*. 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 tree decomposition. 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.

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,* $\mathrm{tw}(G)$*, is defined as the minimum width over all tree decompositions of G.*

Definition. *A tree decomposition (T, β) of a graph G is* nice*, if it satisfies the following conditions:*

- *T is a rooted tree.*
- $d_T(t) \leq 2$ *f.a.* $t \in V(T)$.
- *Every node $t \in V(T)$ is of one of the following types:*

leaf *t is a leaf of T and $|\beta(t)| = 1$.*

introduce *t has exactly one child s, $\beta(s) \subseteq \beta(t)$ and $|\beta(t)| = |\beta(s)| + 1$.*

forget *t has exactly one child, $\beta(t) \subseteq \beta(s)$ and $|\beta(s)| = |\beta(t)| + 1$.*

join *t has exactly two children t_1, t_2 and $\beta(t) = \beta(t_1) = \beta(t_2)$.*

Lemma. *Every graph G with $V(G) \neq \emptyset$ has a nice tree decomposition of width* $\mathrm{tw}(G)$.

sketch. We apply the following transformations as long as possible. We will finally obtain a nice tree decomposition.

Step 1: Choose a root $r \in V(T)$ and orient all edges away from that root.

Step 2: If there is a node $t \in V(T)$ with more that two children, $\{t, s_1\}, \ldots, \{t, s_k\} \in E(T)$ with $k > 2$, perform the following transformation:

- $E(T) := E(T) \setminus \{\{t, s_i\}, i \in \{2, \ldots, k\}\}$.
- subdivide $\{t, s_1\}$ resulting in a new node u and the edges $\{t, u\}, \{u, s_1\}$.

- $\beta(u) = \beta(t)$.
- $E(T) := \{\{u, s_i\}, i \in \{2, \dots, k\}\}$.

Step 3: If there is a node t with exactly 2 children t_1, t_2, such that $\beta(t) \neq \beta(t_1)$ or $\beta(t) \neq \beta(t_2)$, subdivide the edges $\{t, t_i\}, i \in \{1, 2\}$, resulting in two new nodes u_1, u_2. Set $\beta(u_i) = \beta(t), i \in \{1, 2\}$. Now, t is an join node. We can procede with making the u_i's forget or introduce nodes.

Step 4: Let t be a node with exactly one child s. Let $\beta(t) \subseteq \beta(s)$ (the other case can be treated analogously). Let $M := \beta(s) \backslash \beta(t)$. If $|M| > 1$, then subdivide $\{t, s\}$, resulting in a new node u. Set $\beta(u) = \beta(t) \cup \{v\}$ for $v \in M$.

Step 5: Replace leaves by a sequence of introduce nodes: A leaf t with $\beta(t) = \{v_1, \dots, v_k\}$ is replaced by a path $P := (t_1, \dots, t_k)$ with $\beta(t_i) := \{v_i, \dots, v_k\}$. □

Theorem. *Let G be a graph. There is an algorithm that, given a nice tree decomposition (T, β) of G of width $\mathrm{tw}(G) =: k$, computes a minimum vertex cover of G in running time $\mathcal{O}(2^{k+1} \cdot n^{\mathcal{O}(1)})$.*

proof. *Let $t \in V(T)$. By $R_T(t)$, we denote the set of vertices that are reachable from t, including t. Let $\beta(S) := \cup_{t \in S} \beta(t)$ for $S \in V(T)$. For $t \in V(T)$, let $G[t] := G[\beta(R_T(t))]$.*

For $t \in V(T)$ and $X \subseteq \beta(t)$, let $\mathrm{VC}_t(X)$ be the minimum size of a vertex cover X' in $G[t]$ with $X' \cap \beta(t) = X$, if such a X' exists. Otherwise, $\mathrm{VC}_t(X) = \infty$.

Using the propositions below, we can compute $\mathrm{VC}(X)$ f.a. $X \subseteq \beta(t), t \in V(T)$ in time $n^{\mathcal{O}(1)} \cdot 2^{k+1}$. The size of a minimum vertex cover of G is $\min_{X \in \beta(r)} \mathrm{VC}(X)$, were $r \in V(T)$ is the root of T.
We now can construct a minimum vertex cover by tracing back through the tree decomposition.

Proposition. ***leaf:*** *Let t be a leaf of T with $\beta(t) = \{v\}$. Then $\mathrm{VC}(\{v\}) = 1$ and $\mathrm{VC}(\emptyset) = 0$.* □

Proposition. ***forget:*** *Let t be a forget node of T with child s and $\beta(s) \backslash \beta(t) = \{v\}$. Then f.a. $X \in \beta(t), \mathrm{VC}(X) = \min\{\mathrm{VC}(X), \mathrm{VC}(X \cup \{v\})\}$.*

Proof. Let X' be a minimum vertex cover of $G[t] = G[s]$ with $X' \cap \beta(t) = X$. If $x \notin X'$ then $X' \cap beta(s) = X$. Hence, $|X'| \geq \mathrm{VC}_s(X)$. If $x \in X'$, then $|X'| \geq \mathrm{VC}_s(X \cup \{v\})$ for the same reason.
Let X_1, X_2 be the vertex covers that determine $\mathrm{VC}_t(X), \mathrm{VC}_s(X \cup \{v\})$. Both are vertex covers of $G[t]$. Hence, $\mathrm{VC}_s(X) \leq \min\{|X_1, X_2|\}$. □

Proposition. ***introduce:*** *Let t be an introduce node of T with child s and $\beta(t)\backslash\beta(s) = \{v\}$. Then f.a. $X \in \beta(t)$:*

(1) If X is not a vertex cover of $G[\beta(t)]$ then $\mathrm{VC}_t(X) = \infty$.

(2) If X is a vertex cover of $G[\beta(t)]$ and $v \in X$ then $\mathrm{VC}_t(X) = \mathrm{VC}\, s(X\backslash\{v\})+1$.

(3) If X is a vertex cover of $G[\beta(t)]$ and $v \notin X$ then $\mathrm{VC}(X) = \mathrm{VC}(X)$.

Proof. (3): $N_{G[t]}(v) \subseteq \beta(s) \subset X$, since X is a vertex cover of $G[\beta(t)]$. □

Proposition. ***join:*** *Let t be a join node of T with children s_1, s_2. Then f.a. $X \in \beta(t)$, $\mathrm{VC}_t(X) = \mathrm{VC}\, s_1(X) + \mathrm{VC}_{s_2}(X) - |X|$.*

Proof. If X' is a vertex cover of $G[t]$ with $X' \cap \beta(t) = X$, then $X' \cap \beta(R_T(s_i)$ is a vertex cover of $G[s_i], i \in \{1, 2\}$. They share $|X|$ vertices. The vertex covers X_i of $G[s_i]$ that are compatible with X can be combined to a vertex cover of $G[t]$ of stated size.

□

4 Implementations

- C++
- boost graphs
- python interface
- uses exact solver for tree decomposition problem (TdLib)

5 Experiments

5.1 Test graphs

- bounded treewidth
- k-trees
- GNP
- named graphs

5.2 Results

6 Summary

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