RWTH Aachen **SS 2014**
Lehrgebiet Theoretische Informatik Problem Set 7
Reidl—Rossmanith—Sánchez—Sikdar 27.11.2014

Parameterized Algorithms Tutorial

Tutorial Exercise T1

The problem 3-Colorability is defined as follows: Given a graph G decide if it is possible to assign every node of G one of three colors, such that no two nodes with the same color are adjacent. This problem is fpt parameterized by the treewidth of the graph. Give an algorithm that solves the problem given a tree decomposition of width w in time $O(3^w \cdot w \cdot n)$.

Proposed Solution

First convert the tree decomposition into a nice tree decomposition $\mathcal{T}$ in linear time. If X is a bag of the tree decomposition, then let V_X be all nodes which are contained in X and in bag which is a descendant of X in T. We will solve this in the usual way, by computing a table for every bag which will contain entries as follows: For all possible 3-colorings c of $G[V_X]$ we want an entry $c(X)$, i.e. the colors of c for the nodes in X, in the table for the bag X. Such a table can have at most 3^w entries. Having such a table for the root would provide the solution for the decision problem, since by the definition of the table the graph is 3-colorable iff the table for the root bag is not empty. Assume we want to generate the table for some bag X. Let us do it case by case depending on what X can be:

leaf bag Generate a table that contains three entries, one for every color we can give to the only node in the bag.

forget bag Delete the forgotten node from every entry and delete duplicates.

introduce bag When a node x is introduced go through every coloring c of the nodes of the bag in the table of the child of X. If x does not have a neighbor in the bag with color c' add new a entry extending c with x colored with the color c'. This is correct since by the properties of tree decompositions x can only have neighbors which are contained in X in the graph $G[V_X]$.

join bag Only keep the entries which are also contained in both tables of the child bags of X.

Since these operation suffice to compute the table for the root bag in a bottom up fashion, all operations take time $O(3^w)$ for tables which at most 3^w entries and a nice tree decomposition has at most $w \cdot n$ bags it follows that the running time is $O(3^w \cdot w \cdot n)$.

Tutorial Exercise T2

The problem Dominating Set is defined as follows: Given a graph G find the smallest set $S \subseteq V(G)$ such that every node of G is either in S or has a neighbor in S. This problem is fpt parameterized by the treewidth of the graph. Give an algorithm that solves the problem given a tree decomposition of width w in time $O(9^w \cdot w \cdot n)$.

Proposed Solution

We will use the same notation as in for the solution of T1. We will also solve it in a very similar manner. The interpretation of the tables for the bags will this time be somewhat more complicated: In the table we will have an entry which represent partitions of the current bag X intro three sets S, D and F and are associated with a number s. Such an entry is in the table only if there exists a partial dominating set of $G[V_X]$ of size s where every node not in the bag X is dominated, the nodes in S are in the dominating set, the nodes in D are dominated but not in the dominating set and the nodes in F are not dominated. Because we are trying to minimize the size of the dominating set between two entries with the same sets but different sizes s we will only keep the entry where the size is smaller. This means that, since there are only 3^w ways to partition a set of size w into three sets, the tables have at most 3^w entries. If we can compute such a table we would be able to read the size if a minimal dominating set by taking the minimum size over all the entries where F is empty. We can then again describe the operations needed to compute these tables in a bottom up fashion a nice tree decomposition.

leaf bag Create three entries, such that x is contained only in S, D and F respectively. If the nodes is in S set s to one, otherwise to zero.

forget bag Take the whole table of the child bag. Remove all the entries where the forgotten node x is contained in F. Then remove x from every set in the remaining table. We can do this because we only want entries that represent partial dominating sets that dominate every node which is not in the bag X.

introduce bag Take every entry (S, D, F, s) and create three new entries as follows:

- Put x in S, all the neighbors of x in X in the set D and increase s by one. This is correct since by the properties of tree decompositions x can not dominate anything in $G[V_X]$ which is not in the bag X.
- Put x in D if x has a neighbor in S. Keep s.
- Put x in F if x does not have a neighbor in S. Keep s.

join bag Take the tables t_1 and t_2 from the children bags of X. For every pair of entries $(S_1, D_1, F_1, s_1) \in t_1$ and $(S_2, D_2, F_2, s_2) \in t_2$ generate a new entry (S, D, F, s) where

- $S = S_1 \cup S_2$,
- $D = (D_1 \cup D_2) \setminus (S_1 \cup S_2)$,
- $F = X \setminus (S \cup D)$ and
- $s = s_1 + s_2 - |D_1 \cap D_2|$.

This is correct since by the properties of tree decompositions when merging two entries we only have to take care that the solution agrees on the join bag. Since we have to look at all pairs this step takes time $O((3^w)^2) = O(9^w)$.

By using this algorithm on a nice tree decomposition, since the join operation has the worst running time, we have an algorithm with the desired running time.

Tutorial Exercise T3

The problem Hamiltonian Path is defined as follows: Given a graph G, does there exist a path that visits every node of the graph exactly once? Show there is an algorithm which solves the problem in linear time parameterized by treewidth, i.e. in time $f(w) \cdot n$ where w is the treewidth of G.

Proposed Solution

This problem is MSO_2 expressible:

$$\begin{aligned} hampath =& \exists F \subseteq E \, \exists s \exists t \, path(s,t,V,F) \\ path(s,t,S,F) =& \forall x((x \in S \wedge x \neq s \wedge x \neq t) \\ & \rightarrow \exists a \exists p \forall y(a \in S \wedge p \in S \wedge a \neq p \wedge (xFy \rightarrow y = a \vee y = p))) \end{aligned}$$

It follows by Courcelle's Theorem that this formula can be checked in time $f(w) \cdot n$ given a tree decomposition of width w. If a graph has treewidth w a tree decomposition of width w can be computed in time $2^{O(w^3)}$. Thus by first computing a tree decomposition and then applying Courcelle's Theorem the desired running time is achieved.

Homework H1

Let G be a graph and let $S \subseteq V(G)$ be some vertex subset. Show that the following properties are MSO_2-expressible:

- S is a vertex cover of G
- S is an independent set of G
- G is a connected graph
- S induces a cycle in G
- S induces an even cycle in G

Proposed Solution

- Vertex cover: $vc(S) = \forall x \forall y(\neg xEy \vee x \in S \vee y \in S)$
- Independent set: $is(S) = \forall x \forall y(\neg xEy \vee x \notin S \vee y \notin S)$
- Connected: Let us introduce a slightly more general formula.

  $$con(S) = \forall A \forall B((A \subseteq S \wedge B \subseteq S) \rightarrow \exists a \exists b(a \in A \wedge b \in B \wedge aEb))$$

  Where $con(S)$ means that $G[S]$ is connected ($con(V)$ is the formula we need for this part of the exercise)
- Cycle:

  $$\begin{aligned} cycle(S) =& \, con(S) \wedge \forall x \exists a \exists p \forall y((x \in S \wedge y \in S) \\ & \rightarrow a \neq p \wedge a \in S \wedge p \in S \\ & \wedge (xEy \rightarrow y = a \vee y = p)) \end{aligned}$$

  If we would leave out the connectedness-condition, our formula would also be satisfied by a collection of disjoint cycles.

- Even cycle:

$$\begin{aligned} evencycle(S) =& cycle(S) \wedge bipartite(S) \\ bipartite(S) =& \exists A \exists B \forall a \forall b((a \in S \wedge b \in S \wedge aEb) \\ & \rightarrow (a \in A \wedge b \in B) \vee (b \in A \wedge a \in B)) \end{aligned}$$

Homework H2

The Irredundant Set problem is defined as follows: Given a graph G find a set $S \subseteq V(G)$ of maximal size, such that ever node in $u \in S$ has at least one neighbor $v \notin S$ such that $N(v) \cap S = \{u\}$. We call such a neighbor v a private neighbor. Show this problem is fpt parameterized by the treewidth of the graph without relying on Courcelle's Theorem.

Proposed Solution

We will solve this problem in a way similar to our solutions for T1 and T2 and use the notation introduced in the solution for T1. This time the interpretation of our the entries in our tables is as follows: The entries are just labelings of our bags. The labels which we can use for the nodes in the bags are the following:

S1 We want the node in the irredundant set, we have not assigned it a private neighbor.

S2 The node is in the irredundant set and has been assigned a private neighbor.

P1 The node is a private neighbor but has not been used.

P2 The node is a private neighbor and has been used.

U The node has no role.

For every labeling we also need to keep a record of the size s of the irredundant set. Such an entry means that there exists an irredundant set I of size s in $G[V_X]$ that intersects the bag X only in nodes labeled $S2$ and such that all nodes labeled $P2$ are private neighbor of some node in I. Furthermore we plan to add the nodes labeled $S1$ later and use the nodes labeled $P2$ as private neighbors. Notice that then by definition any node labeled $P1$ can have no neighbors labeled $S1$ or $S2$ and any node labeled $S1$ can have no neighbors labeled $P1$ or $P2$. If we compute a complete table for the root bag we can find out the maximum size of an irredundant set by taking the maximal size of any entry which does not have any nodes labeled $S1$ or $P1$. Now, as we did before, we need to show how to do every step of the bottom up computation on a nice tree decomposition.

leaf bag We have three entries where the unique node in the bag is labeled $S1$, $P1$ and U respectively.

forget bag First we take the table from the child bag. What we do with the forgotten node x depends on its label:

- If x is labeled $S1$ or $P1$ delete the entry.
- If x is labeled anything else, keep the entry without the label for x.

introduce bag When a node x is introduced, for every entry in the table from its child bag we generate the following entries:

- We add the node x labeled U.
- If x has no edge with any node labeled $P1$ or $P2$ we add the node x labeled $S1$. Increase the size s by one.
- If x has no edge with any node labeled $S1$ or $S2$ we add the node x labeled $P1$.
- If there exists a node y in X which is labeled $S1$ such that there is an edge between x and y and no edge between x and any node labeled $S2$, then add x labeled $P2$ and change the label of y to $S2$.
- If there exists a node y in X which is labeled $P1$ such that there is an edge between x and y then add x labeled $S2$ and change the label of y to $P2$. Increase the size s by one.

In this way we can construct any possible solutions while we take care to not construct any invalid ones.

join bag Merge the entries which appear in both child tables for the same labeling. The labeling is kept, the size is updated to be the sum of the size of both entries minus the number of nodes labeled $S1$ or $S2$.

Since the size of the tables is bounded by 5^w this leads to an algorithm which solves the problem given a tree decomposition with a running time of $O(5^w \cdot w \cdot n)$.