RWTH Aachen **WS 2011/12**
Lehrgebiet Theoretische Informatik Problem Set 2
Langer—Reidl—Rossmanith—Sikdar 28.10.2011

Parameterized Algorithms Tutorial

Tutorial Exercise T4

The Nemhauser–Trotter-Theorem states that given an undirected graph $G = (V, E)$, one can in polynomial time to find two disjoint vertex sets $C_0 \uplus V_0 \subseteq V$ such that

1. If $D \subseteq V_0$ is a vertex cover of $G[V_0]$, then $D \cup C_0$ is a vertex cover of G.
2. There is an optimal vertex cover S of G with $C_0 \subseteq S$.
3. Every vertex cover of $G[V_0]$ has size at least $|V_0|/2$.

Show that:

1. If S' is an optimal vertex cover of $G[V_0]$, then $S' \cup C_0$ is an optimal vertex cover of G.
2. If G has a vertex cover of size k, then $|V_0| + |C_0| \leq 2k$.

Proposed Solution

Let S_0 be an optimal vertex cover of $G[V_0]$. By condidtion 1, we know that $S' = S_0 \cup C_0$ is a vertex cover of G—it remains to show that is also optimal for G. Assume the contrary and let thus S'' be a vertex of G with $|S''| < |S'|$. By condition 2, we can assume that $C_0 \subseteq S''$ holds. Similary, $C_0 \subseteq S'$ holds by construction of S'. It follows that

$$\begin{aligned} & |S''| < |S'| \\ \Rightarrow\ & |S''| - |C_0| < |S'| - |C_0| \\ \Rightarrow\ & |S'' \setminus C_0| < |S' \setminus C_0| = S_0 \end{aligned}$$

and therefore $S'' \cap V_0$ is smaller than S_0. But this means we found a smaller vertex cover for $G[V_0]$, which contradicts our initial assumption. This proves statement 1.

For statement 2, assume S is a vertex cover of G with $|S| = k$. Again we assume that $C_0 \subseteq S$. By condition 3 we know that $|S \cap V_0| \geq |V_0|/2$. Putting this together we obtain

$$\begin{aligned} |V_0| + |C_0| &= |V_0| + |S \cap C_0| \leq 2|S \cap V_0| + |S \cap C_0| \\ &= 2|S \cap V_0| + (|S| - |S \cap V_0|) \\ &= |S \cap V_0| + |S| \leq 2k \end{aligned}$$

Tutorial Exercise T5

Let Π be a decidable parameterized problem. Show that Π is fixed-parameter tractable if and only if there exists a kernelization algorithm for it.

Proposed Solution

It is easy to see that a kernel implies a fixed-parameter algorithm, as a brute-force approach on a kernelized instance still runs in time $O(f(k)\,poly(n))$.

For the other direction, assume we have an algorithm that decides Π in time $O(f(k)\,poly(n))$. We will restrict ourselves to computable functions f. The following case distinction will give us a kernel for any instance (I, k):

1. Assume $f(k) < n$. Then the algorithm runs in polynomial time, therefore we can solve (I, k) and output a trivial yes- or no-instance accordingly
2. Assume $f(k) > n$. This already bounds the input size n by some function of k, therefore (I, k) itself is already a problem kernel

Of course this approach usually only gives us exponentially-sized kernels.

Tutorial Exercise T6

Consider the Planar Independent Set problem: Given a planar graph G and an integer k, decide whether G has an independent set of size at least k. Design an algorithm that takes as input (G, k), where G is planar, and outputs in polynomial time an equivalent instance (G', k') such that $|V(G')| = O(k)$.

Proposed Solution

As we learned in T2, a planar graph always has a vertex of degree at most five. We obtain a kernel by using the following *greedy* algorithm: find a vertex v of degree at most five and put it into the greedy solution I_{greedy}, remove $N[v]$ from the graph. Repeat this until the graph is empty.

Now, if $|I_{greedy}| \geq k$, we found an appropriate solution and can output a trivial yes-instance to the problem. Otherwise, note that the number of removed vertices is at most $6|I_{greedy}|$ and therefore $n \leq 6|I_{greedy}| \leq 6k$. In both cases we obtain a problem kernel.

We can restate this as follows: an instance of Planar Independent Set can always be solved by the above greedy algorithm if $n > 6k$, therefore all interesting instance have size at most $6k$.

Homework H3

Consider the $n \times n$ sliding puzzle which consists of a frame with $n^2 - 1$ square tiles with numbers on them; the frame has one missing tile and this enables the others to move horizontally and vertically. See, for instance, the following 4×4 puzzle with numbers in hexadecimal:

1	3	6	4
5		7	8
9	2	F	B
D	A	E	C

The puzzle is *solved*, if all numbers are sorted both row-, and column-wise, with the smallest number appearing in the top left corner of the frame, and the empty space on the bottom right corner.

The Sliding Puzzle problem is defined as follows: given an $n \times n$ sliding puzzle and an integer k, decide whether one can solve the puzzle in at most k moves. A move consists in moving a tile either horizontally or vertically one step.

- Show that this problem is in FPT by giving a bounded search-tree algorithm for it.
- Give an algorithm that constructs a kernel of polynomial size in polynomial time.

Proposed Solution

The key observation here is that the *hole* in the puzzle can move in at most four ways (less if we reach the border of the puzzle). This immediately gives us a $O(4^k\, poly(n))$ branching algorithm.

Similarly, the hole cannot move farther than k steps in any direction. We therefore look at a rectangle of size $2(k+1) \times 2(k+1)$ around the initial position of the hole: every pieces *outside* of that rectangle cannot be possibly moved with only k steps. Therefore, if any piece outside this rectangle is *not* at its correct place, we can output a trivial no-instance of the puzzle. Otherwise we restrict ourselve to the rectangle itself by using an appropriate relabeling of the pieces. It follows that the sliding puzzle has kernel of size $O(k^2)$.

Homework H4

Given a graph $G = (V, E)$, an *induced matching* of G is a set of edges $F \subseteq E$, such that the edge set of the induced subgraph $G[V(F)]$ is F itself. The *size* of an induced matching is the number of edges in it. The Induced Matching problem is given a graph G and an integer k, to decide whether G has an induced matching of size at least k.

Design a linear kernel for this problem on graphs of maximum degree d, where d is a constant.

Proposed Solution

We proceed similary as with the kernel in T6, but first we need to remove all vertices of degree zero. Then we greedily build an induced matching M_{greedy} by taking an arbitrary

edge into the matching and removing the at most $2d$ vertices adjacent to its endpoints. Repeat until the graph has no edges left.

If $|M_{greedy}| \geq k$ we are done and can output a trivial yes-instance, otherwise it follows that $n \leq 2d|M_{greedy}| \leq 2dk$. Therefore this problem admits a $2dk$ kernel on graphs of bounded degree.