RWTH Aachen **WS 2018**
Lehrgebiet Theoretische Informatik Problem Set 7
Dreier—Kuinke—Rossmanith 03.12.2018

Parameterized Algorithms Tutorial

Tutorial Exercise T1

Show that Dominating Set $\leq_{\text{FPT}}$ Hitting Set.

Proposed Solution

Let (G, k) be an instance of Dominating Set. Create an instance $(\mathcal{U}, \mathcal{F}, k')$ of Hitting Set as follows. Set $\mathcal{U} = V(G)$ and $\mathcal{F} = \{N[u] \mid u \in V(G)\}$, and $k' = k$. One can easily verify that G has a k-dominating set iff $(\mathcal{U}, \mathcal{F})$ has a k-hitting set.

Tutorial Exercise T2

Given a graph $G = (V, E)$, a *perfect code* for G is a vertex set $S \subseteq V(G)$ such that for all $v \in V(G)$ there is exactly one vertex in $N[v] \cap S$. The Perfect Code problem is defined as follows: given a graph $G = (V, E)$ and an integer parameter k, decide whether G has a perfect code with k vertices. This problem is W[1]-complete on general graphs. Show that this problem is fixed-parameter tractable if we assume that the input graph is planar. Use the fact that every planar graph has a vertex of degree at most five.

Proposed Solution

We want to employ a simple branching algorithm, however, we somehow need to preserve the information that a) a vertex has a neighbour part of the *perfect code* and b) that a vertex cannot be picked as this would cover some neighbour multiple times.

To this end, we mark vertices as *covered* if they have a neighbour that is part of the perfect code and *forbidden* if it is not covered itself but has a covered neighbour.

Now, our recursive algorithm proceeds as follows: pick a vertex v of degree at most five from the graph and branch on the at most six possibilities of including one vertex of $N[v]$ in the solution—of course avoiding vertices marked as *covered* or *forbidden*. In each branch, if vertex $w \in N[v]$ is picked, we remove w from the graph and mark all vertices in $N(w)$ as *covered* and all vertices in $N^2(w)$ (the set of all vertices with distance 2 to w) as *forbidden*.

The correctness of the above algorithm can be verified easily.

Tutorial Exercise T3

The r-Regular Vertex Deletion problem is defined as follows: given a graph G and an integer k, decide whether there is a set $S \subseteq V(G)$ of size at most k whose deletion results in an r-regular graph. A graph is r-regular if every vertex has degree exactly r. Show that this problem admits an algorithm with running time $O((r+2)^k \cdot \text{poly}(n))$.

Proposed Solution

First observe that any vertex of degree $< r$ must necessarily be taken into the solution as we cannot increase the degree of any vertex by removing other vertices. This reduction will be applied after each branching step.

The branching itself proceeds as follows: pick any vertex v of degree larger than r. If no such vertex exists we are done; as the above reduction rule already took care of all vertices of degree less than r. Otherwise, pick any set $A \subseteq N(v)$ of $r+1$ neighbours of v. Note that from the set $v \cup A$ we *must* delete at least one vertex to obtain an r-regular graph. Therefore, we can branch on $r+2$ different cases of choosing a vertex from $v \cup A$ to be part of the solution.

This yields the desired $O((r+2)^k \, poly(n))$-algorithm.

Homework H1

Show that Hitting Set $\leq_{\text{FPT}}$ Dominating Set.

Proposed Solution

Given an instance $(\mathcal{U}, \mathcal{F}, k)$ of Hitting Set, construct a graph $G = (V, E)$ as follows. Define $V(G) = \{x, y_1, \ldots, y_{k+1}\} \cup U \cup F$, where $u_i \in U$ for each element $i \in \mathcal{U}$ and $s_j \in F$ for each set $S_j \in \mathcal{F}$, and $x, y_1, \ldots, y_{k+1}$ are special vertices. Vertex $u_i \in U$ is connected to $s_j \in F$ iff $i \in S_j$ and vertex x is connected to every vertex $u_i \in U$ and to the vertices $y_1, \ldots, y_{k+1}$. The graph G has nor more edges.

We claim that there exists a hitting set of size k iff G has a dominating set of size $k+1$. Suppose that there exists a hitting set of size k. Choose the "corresponding" vertices from U and, in addition, choose vertex x. These $k+1$ vertices clearly dominate all vertices of G. If G has a dominating set of size $k+1$, then this set must include x. For otherwise, one would have to choose $y_1, \ldots, y_{k+1}$ to dominate all of these. Since x also dominates all vertices in U, clearly this set does not contain any vertex from F. This is because vertices in F can dominate vertices of U only. Hence there exists k vertices in U that dominate all of F. The "corresponding" elements of $\mathcal{U}$ hit all sets in $\mathcal{F}$ and hence there exists a hitting set of size k.

Homework H2

The Partial Vertex Cover problem is defined as follows: given a graph G and integers k and t, decide whether there exists k vertices that cover at least t edges. The parameter is the integer t (when parameterized by k only, the problem is W[1]-complete). The point of this exercise is to use color-coding to obtain a randomized FPT-algorithm for this problem.

1. Show that if $t \leq k$ then the problem is polynomial-time solvable. What happens if the maximum degree of the input graph is at least t?

2. Now use the following idea for coloring the vertices of the graph with two colors, say, green and red. Assume that there exists $S \subseteq V(G)$ of size at most k such that S covers at least t edges. Color each vertex red or green with probability $1/2$. Show that the probability that vertices in S are colored green and all vertices in $\{u \in V(G) \setminus S \mid (u, v) \in E(G) \text{ for some } v \in S\}$ are colored red is a function of k and t. Call such a coloring a *proper coloring*.

3. Given a properly colored graph, we now need to identify a solution quickly. Note that the green vertices decompose the graph into connected components and that these contain the potential solution vertices. Show that in a properly colored graph, the solution is always the union of some green components, that is, the solution either includes all vertices of a green component or none. Hence any green component with k or more vertices that does not cover at least t edges can be discarded. Use this to design an algorithm that identifies a solution set in a proper two-colored graph.

4. Use all the above facts to design a randomized FPT-algorithm for the problem and analyze its time complexity.

Proposed Solution

1. If $t \leq k$, then pick k vertices each of degree at least one, in succession. Such a set will cover at least k edges. If such a set cannot be picked then the given instance is a no-instance. Also if the maximum degree is at least t, then picking a vertex of maximum degree suffices. Hence we may assume that $k < t$ and that the maximum degree is at most $t - 1$.

2. Since each vertex has degree at most $t - 1$, the size of the neighborhood set of S is at most $k(t - 1)$. The probability that a coloring is proper is at least

$$\frac{1}{2^{k+k(t-1)}} = \frac{1}{2^{kt}}$$

3. By the very definition of a proper coloring, the solution is the union of some green components. Now a green component with k vertices or more can be discarded. For the remaining components, we maintain a table with indices $1 \leq i \leq k$. For index i, we store a component C_i with i vertices that covers the maximum number of edges among all components on i vertices. Define $\mathscr{C} = \{C_1, \ldots, C_k\}$. Try all possible subsets $\mathscr{C}' \subseteq \mathscr{C}$ such that $\sum_{C \in \mathscr{C}'} |V(C)| \leq k$ and check the number of edges covered by the components in $\mathscr{C}'$. If this number is at least t, then we output the components in $\mathscr{C}'$. The time taken is $O(2^k \cdot \text{poly}(k, t))$ (this can be achieved much faster, but this suffices for our purposes).

4. The randomized algorithm is now clear. It repeatedly does the following: color the vertices of the graphs using colors red or green and then test whether there exists at most k green vertices which cover at least t edges. If the instance is a yes-instance, the expected number of repetitions required is 2^{kt} and hence the expected running time is $O(2^{kt} \cdot 2^k \cdot \text{poly}(n)) = O(2^{t+t^2} \cdot \text{poly}(n))$.