1 Preview: FPT and Kernelization

We will describe another way of aproaching hard problems – fixed parameter tractability. We want to express the running time of the algorithm not only in the terms of the instance size n, but also some *parameter* k of the input, for example the size of the solution we are looking for, the maximum degree of the input graph or some other structural property of the instance. In other words: we do not want to be efficient on all inputs of size n (because we essentially can't acheive that), but only for those where k is small.

Definition 1 (very informal)**.** A parameterized problem *is a decision problem, whose input comes with an additional integer value k called a parameter. It is nothing inherently different from what you are used to already; if you take the* VERTEX COVER *problem –* "Does the input graph G have a vertex cover of size at most k?" *and decide that you would like to express the running time of your algorithm also in terms of the variable k, you can call this variable a* parameter *and thus define a parameterized problem* VERTEX COVER(k).

Definition 2. *A parameterized problem is* fixed-parameter tractable (FPT) *if it is solved by an algorithm running in time $O(f(k)n^c)$ for some function f and some constant c independent of k.*

Remark 1. *It is usually easy to obtain a running time of n^k (this would probably be the running time of the standard brute-force approach), but $O(f(k)n^c)$ is much better!*

2 Technique: Bounded Search Tree, aka Branching Strikes Back

2.1 Forbidden subgraphs

Problem 1. *Find a fixed-parameter algorithm for* VERTEX COVER(k)*. You should first aim for running time $O(2^k)poly(n)$. Can you still improve this algorithm using the improvements of the exact branching algorithm for the vertex cover problem you have learnt in the previous lecture?*

TRIANGLE DELETION	**Parameter:** k
Input: Graph $G = (V, E)$, integer k	
Question: Decide whether you can delete k vertices to remove all triangles (3-cycles) in G.	

Problem 2. *Find an FPT-algorithm for* TRIANGLE DELETION(k).

Definition 3. *A graph property $\mathcal{P}$ is* hereditary *if for every $G \in \mathcal{P}$ and induced subgraph G' of G, we have $G' \in \mathcal{P}$ as well.*

Remark 2. *Every hereditary property $\mathcal{P}$ can be characterized by a (finite or infinite) set $\mathcal{F}$ of forbidden induced subgraphs.*

Theorem 1 (Deletion to hereditary properties)**.** *If $\mathcal{P}$ is hereditary and can be characterized by a* finite *set $\mathcal{F}$ of forbidden induced subgraphs, then the graph modification (deleting vertices/edges, adding edges) parameterized problems corresponding to $\mathcal{P}$ are FPT.*

Remark 3. *The two problems we have tackled so far fit this setting: the property is being edge-less and triangle-free, respectively.*

CLUSTER EDITING	**Parameter:** k
Input: Graph $G = (V, E)$, integer k	
Question: Decide whether you can add or remove at most k edges of G such that in the resulting graph all connected components are cliques.	

Problem 3. *Find an FPT-algorithm for* CLUSTER EDITING(k). Hint: Is the property *"Every component of G is a clique"* hereditary? Can you find the set of forbidden subgraphs?

3 Technique: Kernelization

Definition 4. Kernelization *is a polynomial-time transformation that maps an instance (I, k) of a parameterized problem to an instance (I', k') of the same problem such that:*

1. *(I, k) is a yes-instance if and only if (I', k') is a yes-instance,*
2. *$k' \leq k$*
3. *$|I'| \leq f(k)$ for some function f.*

The resulting instance (I', k') is called a kernel*.*

Remark 4. *This is a formalization of a more familiar notion of pre-processing. Note that small kernels yield quick algorithms.*

Problem 4. *Find a kernelization algorithm for* Vertex Cover*.* Hint: First think about finding some simple pre-processing/reduction rules that might help you to reduce the size of the problem. Which verices can you ignore, and which ones you always have to take to your solution? Maybe applying these rules exhaustively already produces a provably small instance? Can you achieve a kernel of size $k(k+1)$? What about k^2? Can you get $\frac{2}{3}k^2$?

Covering Points With Lines	**Parameter:** k
Input: A set P of n points on a plane, integer k	
Question: Find k lines covering all points.	

Problem 5. *Find a kernelization algorithm for* Covering Points With Lines*(k).*

Problem 6. *Prove that an NP problem is fixed parameter tractable if and only if it has a kernelization algorithm.* Hint: the kernelization algorithm obtained here does not necessarily have to be *clever*.