Definition 1 (reminder)**.** *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.*

1 More Kernelization

Definition 2 (reminder)**.** 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.

We will present a useful tool called the *sunflower lemma*; it is presented in the language of set systems, but can be applied to several different settings.

Definition 3. *A* sunflower *is a collection $\mathcal{S} = \{S_1, \ldots, S_t\}$ of sets with identical pairwise intersections, i.e., $S_a \cap S_b = S_c \cap S_d$ for any $S_a, S_b, S_c, S_d \in \mathcal{S}$.*

Note that this implies that there is a common intersection of all sets, called the *core*: If $C = \bigcap_i S_i$, then for any $S_a, S_b \in \mathcal{S}$ we have $S_a \cap S_b = C$. Note that $C = \emptyset$ (i.e., that all sets are disjoint) is allowed.

Theorem 1 (Sunflower Lemma)**.** *Let d be a constant. Let $\mathcal{S}$ be a collection of more than $d!\, k^d$ sets of size d (over any universe). Then $\mathcal{S}$ contains a collection of at least $k + 1$ sets which form a sunflower, and we can find one in polynomial time.*

Given in class: By the sunflower lemma, d-Hitting Set has a kernel of size $\mathcal{O}(k^d)$.

2 Color Coding

The following lies behind the principle of color coding.

Theorem 2. *Let V be a set of n objects, and $X \subseteq V$ with $|X| = k$. Randomly give values between 1 and k to the elements of V. Then the probability that all elements of X get different colors is at least $1/e^k$.*

Note that the set X in the theorem is unknown; the technique can be applied "blindly".

k-Path	**Parameter:** k
Input: Graph $G = (V, E)$, integer k	
Question: Does k contain a (not necessarily induced) path on k vertices?	

Goal: k-Path has a randomized algorithm with running time $\mathcal{O}^*((2e)^k)$.

The role of the color-coding technique is to reduce k-path to the following problem. It is important to note that the coloring is not "proper" – neighbouring vertices can have the same label.

Colorful Path	**Parameter:** k
Input: Graph $G = (V, E)$, a labelling of V with k different labels	
Question: Does G contain a path using all k labels?	

Problem 1. *Show that* Colorful Path *is FPT.*

Problem 2. *Use this to prove the above goal (that k-path is randomized FPT).*

3 Iterative Compression

The problem we will solve here is the following.

BINARY EQUATIONS	**Parameter:** k
Input: A set E of binary equations $(v_i = v_j)$ or $(v_i \neq v_j)$ over a set of variables V; an integer k.	
Question: Is there a set $S \subseteq E$, $\vert S\vert \leq k$, such that $E - S$ is satisfiable?	

For brevity, let us call such a set S a *deletion set.*

Goal: BINARY EQUATIONS has an FPT algorithm with running time $\mathcal{O}^*(3^k)$.

The principle of Iterative Compression is to go via the following form. (Note the change of parameter.)

COMPRESSION BINARY EQUATIONS	**Parameter:** $\vert S\vert$
Input: A set E of binary equations, as before; a deletion set $S \subseteq E$	
Question: Find a deletion set S' such that $\vert S'\vert < \vert S\vert$ (if possible).	

Problem 3. *Show: An FPT algorithm for* COMPRESSION BINARY EQUATIONS *implies an FPT algorithm for the original* BINARY EQUATIONS *problem. (Given in class if time allows.)*

Now we move on to the problem of solving this "compression" form.

Problem 4. *Show that* COMPRESSION BINARY EQUATIONS *can be reduced to a form such that every equation except those in S is an equality $(x_i = x_j)$.*

Problem 5. *Use the previous to show that* COMPRESSION BINARY EQUATIONS *is FPT. (*Hint: *Decide exhaustively what values the endpoints of the equations in S will get.)*

Remark 1. *The problem* BINARY EQUATIONS *can in particular be used to solve* EDGE BIPARTIZATION*: Delete k edges to make a graph bipartite. (In fact,* BINARY EQUATIONS *and* EDGE BIPARTIZATION *are equivalent.)*