Advanced algorithms

Exercise sheet #3 (Solutions) – Parametric complexity

October 5, 2022

Exercice 1 (Independent set for planar graphs)**.** Recall the Independent set problem:

Input: A graph $G = (V, E)$; an integer k;
Queston: Is there a subset $S \subseteq V$ such that $\#S \geqslant k$ and for any $a, b \in S$, $\{a, b\}$ is not in E?

It is not known to be FPT with respect to k. However, we will show that it is FPT when restricted to planar graphs.

(a) Show that a planar graph with $n \geqslant 3$ vertices has at most $3n - 6$ edges. *Hint: use Euler's formula for connected planar graphs #vertices + #faces = 2 + #edges (which includes the outter face).*

*Solution — Let $G = (V, E)$ be a maximal planar graph (*i.e. *a graph such that for every $u, v \in V$ with $(u, v) \notin E$ the graph $(V, E \cup \{(u, v)\})$ is not planar anymore) with $n \geq 3$ vertices, m edges and f faces. Note that G must be connected. Therefore, by Euler's formula,*

$$n - m + f = 2.$$

Each edge belongs to exactly two faces. Moreover, each face of G must be formed by exactly 3 edges: If there is a face that is not a triangle, then it can be divided by adding a new edge while keeping G planar, which contradicts the maximality. So

$$2m = 3f.$$

Then, $3n - m = 6$ i.e.

$$m = 3n - 6.$$

(b) Show that a planar graph has a vertex of degree at most 5.

Solution — Let G be a planar graph of n vertices and m edges.

- *If $n < 3$, then all the vertices have degree at most 5.*
- *Otherwise, suppose that all the vertices have degree at least 6. Then, by the degree-sum formula,*

$$2m = \sum_{v \in V} \deg(v) \geq 6n$$

i.e.

$$m \geq 3n > 3n - 6$$

which contradicts the previous question. Therefore G must have a vertex of degree at most 5.

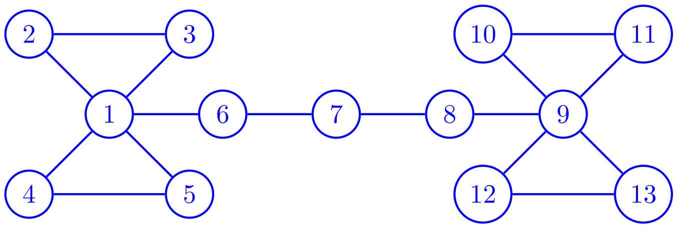
(c) Deduce a $O(6^k n)$-time algorithm for INDEPENDENT SET restricted to planar graph.

Solution —

Before introducing the actual algorithm, let's see why this problem is not so simple. One might think that to get an independent set of size k it is possible to use the following greedy algorithm:

```
procedure WrongIndependentSet(G, k)
    Set S ← ∅
    while #S < k do
        Select one x ∈ V among those of minimal degree;
        S ← S ∪ {x};
        Delete every adjacent vertex and corresponding edge.
    end while
end procedure
```

However, it does not work as can be seen by looking at the following graph:



Vertex 7 has degree 2 which is minimal in the graph, so we might begin with this one. The reader can convince themselves that it will produce the independent set of size 5, while an independent set of size 6 exists.

Instead, we will use a branch and bound approach: Notice for each vertex v, either it belongs to a maximal independent set, or one of its neighbors does. This yields the following algorithm:

- *At each step, we find a vertex minimal degree. It is of degree at most 5.*
- *We pick it or one of its neighbors (nondeterministic choice among at most 6 possibilities).*
- *Then we remove the vertex that we picked and all its neighbors.*
- *We stop if the graph is empty (failure) of if we picked k vertices (success).*

The depth of the tree of all possible computation is at most k and the degree of each node is at most 6. So the size of the tree is $O(6^k)$. At each node, the computation take $O(n)$ operations (finding the small-degree vertex, erasing the vertex, etc.)

Exercice 2 (Hamming center problem (Pâle 2019)).

References

[1] Gramm J, Niedermeier R, Rossmanith P: *Fixed-parameter algorithms for Closest String and related problems.* Algorithmica 2003.

A *word* of length n on the alphabet A is a sequence of n elements of A. Let $u[i]$ denote the ith letter of u, so that $u = u[1] \dots u[n]$. Given two words u and v of length n, let $d(u, v)$ denote their Hamming distance. It is the number of places at which the two words are different: $d(u, v) = |\{i \in \{1, \dots, n\} \mid u[i] \neq v[i]\}|$.

We consider the problem d-CENTER, inspired from error correcting codes questions:

> Problem d-CENTER
>
> **Input:** k words $w_1, \dots, w_k$, each of length n, and an integer d
> **Question:** is there a word u such that $d(u, w_i) \leqslant d$ for any $i = 1, \dots, k$?

To answer the problem d-CENTER we put the words in a $k \times n$ matrix with entries in A. A column i is *bad* if there are two lines h and ℓ such that $w_h[i] \neq w_\ell[i]$; the column is *good* otherwise.

(a) What can we say about the d-center problem when there are more than kd bad columns?

> *Solution — A word u which is at distance less than d from all the w_i cannot disagree in more than d positions from any word. Therefore, there cannot be more than kd conflicting positions in total. However, any bad column induces a conflict. Therefore, if there are more than kd bad columns, the instance has no solution.*

(b) Assume that we are given a word v and an integer ℓ such that the distance of v to some d-center u of $w_1, \dots, w_k$ is at most ℓ. Assume also that v is not a d-center.

Show how to compute, in linear time, $d + 1$ words $v_1, \dots, v_{d+1}$ such that the distance of one of them to some d-center of $w_1, \dots, w_k$ is at most $\ell - 1$.

> *Solution — We have $d(u, v) \leq \ell$. Since v is not a d-center, there exists a word w_i such that v and w_i differ in at least $d + 1$ positions $i_1, \dots, i_{d+1}$. Moreover, since $d(u, w_i) \leq d$, there exists j^* such that $u[i_{j^*}] = w_i[i_{j^*}]$. For $1 \leq j \leq d + 1$, let v_j be equal to v, except in position i_j where it is equal to $w_i[i_j]$. Then, by construction, $d(v_{j^*}, u) \leq \ell - 1$.*

(c) Deduce an algorithm for solving d-CENTER with complexity $O(kn + k(d + 1)^{d+1})$.

> *Solution —*
>
> *i) First, we check if one of the w_i is a d-center in time kn.*
>
> *ii) Then, we identify all the bad columns, in time kn. If there are more than kd of them, then there is no solution. Otherwise, we proceed to the next step.*
>
> *iii) We set $v \leftarrow w_1$ and $\ell \leftarrow d$ and apply algorithm of question (b) to yield $v_1, \dots, v_{d+1}$.*
>
> *iv) If one of them is a d-center, then were are done, otherwise we set $\ell \leftarrow \ell - 1$ and apply recursively step iii) on each of the v_i, until $\ell = 0$.*

The exploration tree has nodes of degree $d+1$ and a depth of at most d, hence the number recursive calls is upper bounded by $(d+1)^d$, each done in time $O(kd)$.

All in all, this algorithm has a complexity of $O(kn + kd(d+1)^d) = O(kn + k(d+1)^{(d+1)})$.

(d) Is this problem FPT ?

Solution — Yes : The input size is $N \doteq kn$ (we doesn't consider the encoding of A). Since $d+1 = O(n)$, the previous algorithm's complexity is upper bounded by $O(N(1+(d+1)^d))$ so the problem is FPT with respect to d.

Exercice 3 (Covering points by lines)**.** Consider the NP-complete problem Line cover:

Input: A set P of points in the plane; an integer k;
Question: Is P coverable by k lines?

Show that instances of Line cover admit kernels of size $O(k^2)$. (And therefore, Line cover is FPT with respect to k.)

Solution — Notice that if there is a line covering at least $k+1$ points, it is mandatory in the covering if we want to use at most k lines. Indeed, the $k+1$ points on such a line would otherwise require $k+1$ separate lines since any two lines share at most one point. Therefore, the problem is to know whether it is possible to cover the remaining points with at most $k-1$ lines.

More formally, consider the following reduction: If there is a line covering a set S of at least $k+1$ points, then we remove them and consider the new instance $(P \setminus S, k-1)$.

We have to check that

1. *This really is a reduction,* i.e. *$(P \setminus S, k-1)$ has a solution if and only if (P, k) has one.*
2. *Computing S, if it exists, requires only polynomial time.*
3. *The kernel (that is the instance obtained after reducing as much as possible) has size $\leqslant k^2$ or has no solution.*

Indeed,

1. *If $P \setminus S$ can be covered by a set $\mathcal{C}$ of at most $k-1$ lines, then considering the line L passing through all the points in S, $\mathcal{C} \cup \{L\}$ is a covering of P by at most k lines. Therefore, if $(P \setminus S, k)$ is accepted, then so is (P, k). Conversely, if (P, k) is accepted, let $\mathcal{C}$ be a covering of P by at most k lines. Then, by the remark above, the line L passing through S must be in $\mathcal{C}$. Therefore, $\mathcal{C} \setminus \{L\}$ is a covering of $P \setminus S$ by at most $k-1$ lines, and $(P \setminus S, k-1)$ is accepted.*
2. *We may assume that each line covers at least 2 points (Or is horizontal), therefore there are at most $n(n-1)/2 + n = \mathcal{O}(n^2)$ candidate lines to test. Which can be done in polynomial time.*
3. *If no line contains more than k points, then k lines can cover at most k^2 points. Thus, if $n > k^2$ and the reduction can't be applied anymore, there is no solution. Therefore, this problem has a kernel of size k^2.*

Exercice 4 (Kernel for VERTEX COVER)**.** We aim at proving that VERTEX COVER can be solved with $O(kn + 5^{k/4}k^3)$ operations, where k is the size of the cover and n is the number of vertices of the input graph.

(a) Show that instances of VERTEX COVER have kernels of size $\leqslant k^2$. *Hint: Consider points with large degree, if any.*

> *Solution — Notice that if a graph G has a vertex v of degree at least $k+1$, then it* must *belong to* any *covering of size k. Therefore, we can remove it and consider the instance $(G \setminus \{v\}, k-1)$.*
>
> *Otherwise, k vertices can only cover k^2 edges, therefore if there are more than k^2 edges, the instance has no solution.*

(b) For any graph G, let $\Delta(G)$ be the maximum degree of a vertex of G. Show that VERTEX COVER restricted to graphs with $\Delta(G) \leqslant 2$ is easily solvable.

> *Solution — If $\Delta(G) = 2$ then G is a disjoint union of paths and cycles. A path with n vertices has a minimal cover with $\lfloor \frac{n}{2} \rfloor$ elements, while a cycle with n vertices has a minimal cover with $\lceil \frac{n}{2} \rceil$ elements. This gives a linear time algorithm.*

(c) Use a *branch and bound* approach to show that VERTEX COVER is solvable in $O(1.5^k(n+m))$ operations, where n is the number of vertices and m the number of edges.

> *Solution — The main idea is the following: if C is a vertex cover, then for every vertex v, either v is in C or all its neighbors are, which we denote by $N(v)$.*
>
> - *If there is a vertex v of degree at least 3, this can be found in linear time, and we call recursively the algorithm on $(G \setminus v, k-1)$ and $(G \setminus N(v), k - \deg(v))$.*
> - *When we reach a graph of maximum degree 2, we apply the linear-time algorithm from Question (b).*
>
> *The depth of the* branch and bound *tree is bounded by a number $T(k)$ which satisfies $T(k) \leq T(k-1) + T(k-3)$ (at each node, we pick one vertex or we pick at least 3). Then we can proove by induction that $T(k) = O((5^{1/4})^k)$.*
>
> *The computations at each node and each leaf of the tree take linear time.*

(d) Conclude.

> *Solution — We first compute a kernel of size $\leqslant k^2$ in time $O(kn)$. The kernel has $\leqslant k^2$ vertices of degree $\leqslant k$, so it has $\leqslant k^3$ edges. Then we then apply the* branch and bound *algorithm, which gives the desired bound.*