

Parameterized Algorithms Tutorial

Tutorial Exercise T21

In this exercise we will use the technique of color-coding to design a randomized FPT-algorithm for the k -PATH problem which is defined as follows: given a graph G and an integer k as input, decide whether G has a path on at least k vertices.

1. Show that if G does indeed have a path \mathcal{P} on k vertices, then by assigning colors to the vertices of G from the set $\{1, \dots, k\}$ uniformly at random, one colors the vertices of \mathcal{P} with *distinct* colors with probability at least e^{-k} .
2. Now assume that we are given a graph whose vertices are colored using colors from the set $\{1, \dots, k\}$. Show that one can find a path with k distinctly colored vertices in time $O(2^k \cdot k \cdot |E(G)|)$.
3. Use the above facts to design a randomized FPT-algorithm for the k -PATH problem. What is the expected running time of the algorithm? Can you modify the algorithm for the k -CYCLE problem?

Proposed Solution

1. Let v_1, \dots, v_k be a k -path in the graph G . The probability that these k vertices are assigned distinct colors is:

$$\frac{k^{n-k} k!}{k^n} = \frac{k!}{k^k} > \frac{1}{k^k} \cdot \left(\frac{k}{e}\right)^k = e^{-k}.$$

The inequality used above is Stirling's approximation: $k! \approx \sqrt{2\pi k} \cdot (k/e)^k$.

2. First add a new vertex s , assign it color 0 and make it adjacent to all vertices of G . We will use dynamic programming to detect a colorful path on $k+1$ vertices starting at s in the time specified, if one exists. For each vertex v , we maintain a table with indices $1, \dots, k+1$; corresponding to index i , the table stores the set of colors on colorful paths on i vertices that start at s and end at v . There can be at most $\binom{k+1}{i}$ sets for entry i and the table has at most 2^{k+1} . Note that we are *not* storing the number of paths from s to v of length i of which there could be n^i . The tables are initialized as follows: for each vertex v , set the entry $v[2]$ to $\{0, \text{col}(v)\}$; all other entries are set to \emptyset . For updating the tables: for each vertex v , if table entry $v[i]$ contains a set C and if v has a neighbor u with $\text{col}(u) \notin C$, then update the entry $u[i+1] = C \cup \text{col}(u)$. The graph G has a path on k vertices iff there exists a vertex v whose $k+1$ st table entry is non-empty. The total time taken is $O(\sum_v \sum_{i=0}^{k+1} i \binom{k+1}{i} \text{deg}(v)) = O(2^k \cdot k \cdot |E|)$.

3. The randomized algorithm performs the following steps in an infinite loop: randomly color the vertices of the graph using colors $\{1, \dots, k\}$; use dynamic programming to check whether there exists a colorful path of length $k + 1$. If there is a path of length k , it is properly colored with probability at least e^{-k} and hence the expected number of iterations is e^k and the expected running time is $O((2e)^k \cdot k \cdot |E|)$.
4. The graph G has a k -cycle iff there exist vertices u, v such that there is a path on k vertices starting at u and ending at v and $\{u, v\} \in E(G)$. Modify the dynamic programming algorithm so that it does not add a new vertex s but “starts” at a vertex u of the graph and checks whether there exists a path on k vertices starting at u . For each random coloring, repeat the modified dynamic programming algorithm $|V|$ times, with each vertex as a possible start vertex. That is, for each vertex $u \in V$, start at u and check whether there exists k -path from u ending at v such that $\{u, v\} \in E(G)$. The total expected time taken is $O((2e)^k \cdot k \cdot |E| \cdot |V|)$.

Tutorial Exercise T22

We now consider the following idea of using only two colors to obtain a randomized FPT-algorithm for k -PATH. As before assume that the input graph G has a path \mathcal{P} on k vertices. Color the vertices of the graph using two colors, say, black and white uniformly at random. Show that the probability of obtaining a coloring where \mathcal{P} is split into two roughly equal parts G_1 and G_2 is 2^{-k} . Use this idea and recursion to design an algorithm for k -PATH. Analyze its running time.

Proposed Solution

Suppose that there exists a path on k vertices starting at u and ending at v . The probability that a random coloring $c: V \rightarrow \{1, 2\}$ assigns the first $\lfloor k/2 \rfloor$ vertices the color 1 and the remaining $\lceil k/2 \rceil$ vertices the color 2 is $2^{n-k}/2^n = 2^{-k}$. Call such a coloring *proper*. Intuitively, the algorithm repeatedly colors the vertex sets using two colors and for each partition $V = V_1 \cup V_2$, recursively checks whether there exists vertices $u, v \in V_1$ and $w, x \in V_2$ such that there exist paths of length $k/2$ from u to v in $G[V_1]$ and from w to x in $G[V_2]$ and such that $\{v, w\} \in E(G)$. Strictly speaking, the algorithm outlined below actually outputs all vertex pairs (u, v) such that there is a path on k vertices from u to v .

Input: A graph $G = (V, E)$ and an integer k .

Output: The set $\{(u, v) \in V^2 \mid u \xrightarrow{k} v\}$

Path($G = (V, E), k$)

if $k = 1$ **then return** $\{(v, v) \mid v \in V\}$;

for $3 \cdot 2^k$ **times do**

Choose $V_1 \in 2^V$ uniformly at random;

$G_1 = G[V_1], G_2 = G[V \setminus V_1]; R = \emptyset$;

for all $u, x, y, v \in V$ **do**

if $(u, x) \in \text{Path}(G_1, \lfloor k/2 \rfloor)$ and $(x, y) \in E$ and $(y, v) \in \text{Path}(G_2, \lceil k/2 \rceil)$

then $R := R \cup (u, v)$;

return R ;

We want to analyze the running time of this algorithm.

1. Suppose that the probability that the algorithm does not return a k -path, if one exists, is p_k . Then we claim that:

$$p_k \leq \left(1 - \frac{1}{2^k} + \frac{p_{\lfloor k/2 \rfloor}}{2^{k-1}}\right)^{3 \cdot 2^k}.$$

The algorithm might fail for two reasons: either the coloring is not proper *or* that the coloring is proper but the recursive detection of the path fails. The first event occurs with probability $1 - 2^{-k}$ and the second with probability $2^{-k}(p_{\lfloor k/2 \rfloor} + p_{\lceil k/2 \rceil})$. Assuming that p_i is an increasing function of i , and since there are $3 \cdot 2^k$ iterations, we have:

$$p_k \leq \left(1 - \frac{1}{2^k} + \frac{p_{\lfloor k/2 \rfloor}}{2^{k-1}}\right)^{3 \cdot 2^k}.$$

2. We now use induction on k to show that $p_k < 1/4$. Now $p_1 = 0$ and for $k \geq 2$ we have:

$$p_k \leq (1 - 2^{-k} + 2^{-k+1}/4)^{3 \cdot 2^k} = (1 - 2^{-(k+1)})^{\frac{3}{2} \cdot 2^{k+1}} \leq e^{-3/2} < 1/4.$$

The second-last inequality follows from the fact that $f(i) = (1 - 1/i)^i$ is strictly increasing in the interval $[1, \infty)$ and converges to e^{-1} .

3. Let $T(k)$ be the number of recursive calls issued by the algorithm. Then

$$T(k) \leq 3 \cdot 2^k (T(\lfloor k/2 \rfloor) + T(\lceil k/2 \rceil)) \leq 3 \cdot 2^{k+1} T(\lceil k/2 \rceil).$$

Using the fact that $k + k/2 + k/4 + \dots + 1 \leq 2k + \log k$, we obtain

$$T(k) \leq 3^{\log k} \cdot 2^{2k + \log k} \leq k^{\log 3} \cdot k \cdot 4^k.$$

The algorithm finds a k -path with probability $3/4$ if one exists. Iterating it c times, lowers the error probability to $1/4^c$ whilst keeping the running time at $O(4^k \cdot \text{poly}(n, k))$.

Homework H17

The TRIANGLE PACKING problem is defined as follows: given a graph $G = (V, E)$ and an integer k , decide whether G has k vertex-disjoint 3-cycles. Use the idea of randomly coloring the vertices of G with k colors to enable easy detection of vertex-disjoint triangles. What is the expected running time of your algorithm?

Proposed Solution

Assume that the graph G has a set of k vertex-disjoint triangles. Call a random coloring $c: V(G) \rightarrow \{1, \dots, k\}$ of the vertices *proper* if all vertices of a triangle receive the same color but different triangles receive different colors. Given a proper coloring of a yes-instance of the problem, one can find the triangles by searching for them in the graphs induced by the vertices of each color class, in turn. The probability that a coloring is proper is at least

$$\frac{k^{n-3k} \cdot k!}{k^n} = \frac{k!}{k^{3k}} > \frac{1}{k^{3k}} \cdot \left(\frac{k}{e}\right)^k = \left(\frac{1}{k^2 e}\right)^k.$$

The randomized algorithm repeats the following steps in an infinite loop: it randomly colors the vertices of the graph using colors $\{1, \dots, k\}$; it then checks. Since one obtains a proper coloring with probability $(k^2 e)^{-k}$, the expected running time is $O((k^2 e)^k \cdot \text{poly}(n))$.

Homework H18

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 $\mathcal{C} = \{C_1, \dots, C_k\}$. Try all possible subsets $\mathcal{C}' \subseteq \mathcal{C}$ such that $\sum_{C \in \mathcal{C}'} |V(C)| \leq k$ and check the number of edges covered by the components in \mathcal{C}' . If this number is at least t , then we output the components in \mathcal{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 $O(2^{kt} \cdot 2^k \cdot \text{poly}(n)) = O(2^{t+t^2} \cdot \text{poly}(n))$.