RWTH Aachen **SS 2013**
Lehrgebiet Theoretische Informatik Problem Set 6
Reidl—Rossmanith—Sánchez—Sikdar 18.06.2013

Parameterized Algorithms Tutorial

Tutorial Exercise T15

The Nemhauser-Trotter kernelization algorithm for Vertex Cover transforms an instance (G', k') into an equivalent instance (G, k) in time $O(k'n+k'^3)$ such that $|V(G)| \leq 2k$ and $k \leq k'$. If we use the 2^k-bounded search-tree algorithm to then decide the kernelized instance, the total time taken to decide (G', k') is: $O(k'n + k'^3 + 2^k \cdot k^2)$. If we use the technique of *interleaving* whereby we branch and kernelize repeatedly, it seems intuitive that one can obtain a faster algorithm. Analyze the running time of an algorithm that interleaves kernelization and branching.

Proposed Solution

Let $p(n, k)$ denote the time taken to run the Nemhauser-Trotter kernelization algorithm on an instance (G, k) with n vertices. Since $p(n, k) = kn + k^3$, for an instance that is already kernelized, we may write this as $c' \cdot k^3$. Suppose that (G', k') is the input to the problem and that (G, k) is the instance obtained by the Nemhauser-Trotter kernelization algorithm. Then $|V(G)| \leq 2k$ and $k \leq k'$ and the question is whether G has a vertex cover of size at most k. Fix an integer $0 \leq r \leq k$ (whose value will be determined later). The idea is branch and kernelize repeatedly until the parameter drops to r in which case the instances associated with the nodes of the search tree have at most $2r$ vertices. Note that every time we make a binary decision on an edge, we create two induced subgraphs. The Nemhauser-Trotter kernelization also creates induced subgraphs. Therefore the instance obtained after branching and kernelization at every node of the search tree is an *induced subgraph* of the original instance. If we therefore store solutions of induced subgraphs with at most $2r$ vertices in a database, then we need not branch any further and we can look up the database for solutions.

Since the instance that we started branching on had at most $2k$ vertices, there are at most

$$\binom{2k}{2r}$$

possible induced subgraphs with $2r$ vertices. Constructing the set of all possible solutions (which are of size at most r) for each of these induced subgraphs takes time proportional to 2^r and hence the total time taken to construct the database is

$$\binom{2k}{2r} \cdot 2^r.$$

Every time we branch, the parameter drops by one and therefore when the value of the parameter is r, the depth of the search tree is $k - r$. The time taken at each node of the search tree is the time taken to branch and kernelize and then recurse. Let the instance we are branching on is (G, k) where $|V(G)| \leq 2k$, and let $T(k)$ be the total time taken to process (G, k). Then

$$T(k) = \begin{cases} 2T(k-1) + c \cdot k^3 & \text{for } k > r \\ s(r) & \text{for } k \leq r, \end{cases}$$

where c is some constant and $s(r)$ is the time taken for searching the database for solutions to an induced subgraph with $2r$ vertices. A solution to a non-homogeneous recurrence such as this is a linear combination of the solution to the homogenous part of the recuurence and one solution (a particular solution) that satisfies the recurrence. The solution to the homogenous part is 2^k and when the inhomogeneity is a polynomial $P(k)$, then a particular solution of the recurrence is a polynomial of the same degree as $P(k)$. In this case, a particular solution is $a_3k^3 + a_2k^2 + a_1k + a_0$, for some constants a_0, a_1, a_2, a_3. Substituting the polynomial $a_3k^3 + a_2k^2 + a_1k + a_0$ in the recurrence $T(k) = 2T(k-1) + ck^3$, we have:

$$a_3k^3 + a_2k^2 + a_1k + a_0 = 2(a_3(k-1)^3 + a_2(k-1)^2 + a_1(k-1) + a_0) + ck^3,$$

from which we obtain the values $a_3 = -c$, $a_2 = -6c$, $a_1 = -18c$, and $a_0 = -22c$. The total time taken to decide an instance (G', k') is at most

$$p(n, k) + \binom{2k}{2r} \cdot 2^r + 2^{k-r} + \max\{c \cdot k^3, s(r)\}.$$

All that remains is to choose r appropriately so that this expression is minimized. Since the exponential part of the running time dominates, it is sufficient to choose r such that $\binom{2k}{2r} \cdot 2^r + 2^{k-r}$ is minimized.

Tutorial Exercise T16

Consider the $n \times n$ sliding puzzle which consists of a frame with $n^2 - 1$ square tiles with numbers on them; the frame has one missing tile and this enables the others to move horizontally and vertically. See, for instance, the following 4×4 puzzle with numbers in hexadecimal:

1	3	6	4
5		7	8
9	2	F	B
D	A	E	C

The puzzle is *solved*, if all numbers are sorted both row-, and column-wise, with the smallest number appearing in the top left corner of the frame, and the empty space on the bottom right corner.

The SLIDING PUZZLE problem is defined as follows: given an $n \times n$ sliding puzzle and an integer k, decide whether one can solve the puzzle in at most k moves. A move consists in moving a tile either horizontally or vertically one step.

- Show that this problem is in FPT by giving a bounded search-tree algorithm for it.
- Give an algorithm that constructs a kernel of polynomial size in polynomial time.

Proposed Solution

The key observation here is that the *hole* in the puzzle can move in at most four ways (less if we reach the border of the puzzle). This immediately gives us a $O(4^k\,poly(n))$ branching algorithm.

Similarly, the hole cannot move farther than k steps in any direction. We therefore look at a rectangle of size $2(k+1) \times 2(k+1)$ around the initial position of the hole: every piece *outside* that rectangle cannot be possibly moved in k steps. Therefore, if any piece outside this rectangle is *not* at its correct place, we can output a trivial no-instance of the puzzle. Otherwise we restrict ourselves to the rectangle itself by using an appropriate relabeling of the pieces. It follows that the sliding puzzle has kernel of size $O(k^2)$.

Homework H11

Complete the solution to Tutorial Exercise T15. Use numerical tools such as GNU Octave to find out the value of r for which the value of

$$\binom{2k}{2r} \cdot 2^r + 2^{k-r}$$

is minimized. What is the running time of the algorithm for this value of r?

[Hint: At the minimum, $2^{k-r} = \binom{2k}{2r} \cdot 2^r$. Also, if $r = 2\alpha k$, then $\binom{2k}{2\alpha k} \approx \left(\alpha^\alpha (1-\alpha)^{(1-\alpha)}\right)^{-2k}$].

[10 points]

Homework H12

Given a graph $G = (V, E)$, an *induced matching* of G is a set of edges $F \subseteq E$, such that the edge set of the induced subgraph $G[V(F)]$ is F itself. The *size* of an induced matching is the number of edges in it. The Induced Matching problem is given a graph G and an integer k, to decide whether G has an induced matching of size at least k. Design a linear kernel for this problem on graphs of maximum degree d, where d is a constant. [Hint: Start with any maximal induced matching and then play around with the degree bound.]

[10 points]