RWTH Aachen
Lehrgebiet Theoretische Informatik
Reidl—Rossmanith—Sánchez—Sikdar

SS 2013
Problem Set 8
09.07.2013

Parameterized Algorithms Tutorial

Tutorial Exercise T21

Provide an FPT-reduction from Independent Set to Short Turing Machine Acceptance (STMA).

Proposed Solution

In this solution and the ones that follow, we will not be very formal in the Turing machine constructions. The goal is to get a feeling as to why such a reduction should work and not get bogged down in the details of Turing machine constructions.

Let (G, k) be an instance of the Independent Set problem. Construct a non-deterministic Turing machine T_G whose input alphabet consists of $n + 1$ symbols $\{1, \ldots, n, \#\}$, where $n = |V(G)|$, and whose tape alphabet consists of the blank symbol $\{B\}$ and which works as follows:

1. The machine writes k symbols on its tape from the set $\{1, \ldots, n\}$.
2. It then verifies that the symbols written are distinct.
3. It then constructs the subgraph G' of G induced by these k vertices.
4. Finally, it verfies whether G' has edges and if not, it accepts.

Steps 1 and 2 take time $O(k)$ and $O(k^2)$. Assuming that the graph G is "hardwired" in the machine as an adjacency matrix, Steps 3 and 4 together take time $O(k^2)$. The size of the state space of the machine M and the transition function table can be seen to be polynomial in the size of G. A very rough estimate is as follows: $O(kn)$ states for choosing k vertices; $O(k)$ states for verifying whether the vertices chosen are all distinct; $O(k^2)$ states for constructing the subgraph and another $O(k^2)$ states for verifying whether the subgraph has any edges.

Tutorial Exercise T22

Provide an FPT-reduction from Dominating Set to Short Multi-Tape Turing Machine Acceptance.

Proposed Solution

Given an instance (G, k) of Dominating Set, construct a machine with $n + 1$ tapes, where $n = |V(G)|$. The input alphabet of the machine is $\{1, \ldots, n, \#\}$ and the tape alphabet is $\{B\}$, denoting the blank symbol. The machine works as follows:

1. It first writes down symbol i (denoting the ith vertex) on the ith tape.

2. Over a set of k moves, it chooses k vertices non-deterministically and writes these on to the $n+1$st tape.

3. It verifies that the k vertices chosen are distinct.

4. It then moves the head of the $n+1$st tape to the starting position (where it started writing down the candidate vertices). If vertex i is dominated by the jth vertex on tape $n+1$, and the ith tape head does not see a #, the machine moves the ith tape head one cell to the right and prints a # on that cell. For a fixed j on tape $n+1$, the machine does this simultaenously for all i satisfying this condition.

5. The machine accepts iff all tape heads from 1 to n see a #.

The total time taken by the machine is $O(k^2)$. One can again verify that the number of states and the size of the transition function table is polynomial in n.

Tutorial Exercise T23

Consider the following variant of VERTEX COVER:

PARTIAL VERTEX COVER

Input:	A graph $G = (V, E)$, an integers k and t
Parameter:	The integer k.
Question:	Are there k vertices in G that cover at least t edges?

Show that PARTIAL VERTEX COVER is $W[1]$-complete. Show inclusion in $W[1]$ by reducing it to STMA and hardness by reducing from INDEPENDENT SET.

Proposed Solution

Membership in $W[1]$:

The important point about this reduction is that one has to essentially compute how many edges a set of k candidate vertices covers. One cannot hope to do this in $f(k)$ time for any f, if the machine actually *writes* down the number of edges covered by each vertex and then adds these numbers. What one would do in such a case is observe that the size of the state space of the Turing machine can be polynomial in the input graph size and use the state space to count the number of edges.

Let us assume that in addition to the adjacency matrix of the input graph, the machine has hardwired into it the number of edges covered by each vertex. If the machine guesses k candidate vertices as solution, then one possible way of counting the number of edges covered by such a set is as follows. The machine has a special set of $nm+1$ states $\{q_0, q_1, \ldots, q_{nm}\}$ explicitly for counting. Initially the machine is at "count-state" q_0. If the machine is at count-state q_j on seeing vertex v_i from the candidate solution, the machine moves to state $q_{j+\deg(v_i)}$ and moves the head one cell to the right. The value of $\deg(v_i)$, the degree of vertex v_i, is already hardwired in the machine and this doesn't have to be computed. One can see that this allows computing the total number of edges covered by the candidate solution in $O(k)$ steps. However there's one snag. Edges whose both end-points are in the candidate solution are counted twice. The machine now constructs the subgraph induced by the candidate solution and actually counts the number of edges in it. This is fine as the size of this subgraph is at most $O(k^2)$. For every edge that is counted twice, the machine subtracts one from the count (in the state space). If the final count-state of the machine is q_i then the machine accepts iff $i \geq t$.

$W[1]$-completeness:

Reduction from INDEPENDENT SET. Given an instance $(G = (V, E), k)$ of the INDEPENDENT SET problem, create a graph $G' = (V', E')$ as follows: for every vertex $u \in V$, add $|V| - \deg(v)$ new vertices and connect them to v. The instance of PARTIAL VERTEX COVER is $(G', k, t := k \cdot |V|)$. We claim that G has an independent set of size k iff G' has k vertices that covers at least t edges.

Homework H16

Show that:

1. HITTING SET $\leq_{\text{FPT}}$ DOMINATING SET.
2. DOMINATING SET $\leq_{\text{FPT}}$ HITTING SET.

[10 points]

Homework H17

Consider the following variant of HITTING SET:

HALF 3-HITTING SET

Input:	A finite universe U, a family $\mathcal{F} \subseteq 2^U$ of sets of size exactly three, an integer k.
Parameter:	The integer k.
Question:	Can k elements of U hit at least $\lvert\mathcal{F}\rvert/2$ sets?

Show that HALF 3-HITTING SET is in $W[1]$.

[10 points]