RWTH Aachen **WS 2011/12**
Lehrgebiet Theoretische Informatik Problem Set 7
Langer—Reidl—Rossmanith—Sikdar 09.12.2011

Parameterized Algorithms Tutorial

Tutorial Exercise T18

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 T19

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 T20

Consider the following variant of Vertex Cover:

Half Partial Vertex Cover

Input: A graph $G = (V, E)$, an integer k.
Parameter: The integer k.
Question: Can k vertices in G cover at least $|E|/2$ edges?

Show that Half Partial Vertex Cover is in $W[1]$ by reducing it to STMA.

Proposed Solution

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 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 $i \geq m/2$.

Homework H15
Show that:

1. Hitting Set $\leq_{\text{FPT}}$ Dominating Set.
2. Dominating Set $\leq_{\text{FPT}}$ Hitting Set.

Proposed Solution

1. Hitting Set $\leq_{\text{FPT}}$ Dominating Set. Given an instance $(\mathcal{U}, \mathcal{F}, k)$ of Hitting Set, construct a graph $G = (V, E)$ as follows. Define $V(G) = \{x, y_1, \ldots, y_{k+1}\} \cup U \cup F$, where $u_i \in U$ for each element $i \in \mathcal{U}$ and $s_j \in F$ for each set $S_j \in \mathcal{F}$, and $x, y_1, \ldots, y_{k+1}$ are special vertices. Vertex $u_i \in U$ is connected to $s_j \in F$ iff $i \in S_j$ and vertex x is connected to every vertex $u_i \in U$ and to the vertices $y_1, \ldots, y_{k+1}$. The graph G has nor more edges.

   We claim that there exists a hitting set of size k iff G has a dominating set of size $k+1$. Suppose that there exists a hitting set of size k. Choose the "corresponding" vertices from U and, in addition, choose vertex x. These $k+1$ vertices clearly dominate all vertices of G. If G has a dominating set of size $k+1$, then this set must include x. For otherwise, one would have to choose $y_1, \ldots, y_{k+1}$ to dominate all of these. Since x also dominates all vertices in U, clearly this set does not contain any vertex from F. This is because vertices in F can dominate vertices of U only. Hence there exists k vertices in U that dominate all of F. The "corresponding" elements of $\mathcal{U}$ hit all sets in $\mathcal{F}$ and hence there exists a hitting set of size k.

2. Dominating Set $\leq_{\text{FPT}}$ Hitting Set. Let (G, k) be an instance of Dominating Set. Create an instance $(\mathcal{U}, \mathcal{F}, k')$ of Hitting Set as follows. Set $\mathcal{U} = V(G)$ and $\mathcal{F} = \{N[u] \mid u \in V(G)\}$, and $k' = k$. One can easily verify that G has a k-dominating set iff $(\mathcal{U}, \mathcal{F})$ has a k-hitting set.

Homework H16
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 $W[1]$.

Proposed Solution
As in Tutorial Exercise T20, we reduce to the STMA problem. Given an instance $(\mathcal{U}, \mathcal{F}, k)$ of the problem, we construct a non-deterministic Turing machine M which has hardwired into it a three-dimensional array $A[n \times n \times n]$ such that $A[p, q, r] = 1$ iff $\{p, q, r\} \in \mathcal{F}$ (assume that $\mathcal{U} = \{1, \ldots, n\}$). In addition, the machine also has hardwired into it, for each $i \in \mathcal{U}$, the value $m(i) = |\{S \in \mathcal{F} \mid i \in S\}|$. The machine has a special set of $\sum_i^n m(i) \leq n^4$ "count" states q_i which enables it to count the number of sets covered by a candidate solution.

The machine has as input alphabet $1, \ldots, n, \#$ and as tape alphabet the blank symbol and works as follows:

1. First it guesses a set of k numbers from the set $1, \ldots, n$ and writes these on to its tape. It then verfifes whether the numbers chosen are distinct. These represent the vertices of the candidate solution.

2. The machine moves to count state q_0 and starts reading the guessed vertices. If the machine is in count state q_i and sees vertex u, it then moves to count state $q_{i+m(u)}$ and moves its head once cell to the right. At the end of $k+1$ steps, when it has finished reading all guessed vertices, it moves to a count state that denotes the sum of the total number of times a set is covered by one of the candidate vertices.

3. Now for each triple T of distinct numbers from the candidate solution, if $T \in \mathcal{F}$, then the machine decreases it count state by two. The machine writes these sets on its tape.

4. At the end of this procedure, for each tuple T of the candidate solution, if T can be augmented to a set that has not already been written down in the last step, it decreases the count state by one.

5. The machine accepts if its final count state is q_i where $i \geq |\mathcal{F}|/2$.

Steps 1 and 2 take $O(k^2)$ time in total; Step 3 takes $O(k^3)$ time and Step 4 takes $O(k^2 \cdot k^3) = O(k^5)$ (as we might have to read the triples from Step 3) time.