10. Randomized Algorithms: color coding and monotone local search
COMP6741: Parameterized and Exact Computation

Edward Lee²
Semester 2, 2017

Contents

1 Introduction 1
2 Vertex Cover 2
3 Feedback Vertex Set 2
4 Color Coding 4
5 Monotone Local Search 5

1 Introduction

Randomized Algorithms

- Turing machines do not inherently have access to randomness.
- Assume algorithm is also given access apart to a stream of random bits.
- With \( r \) random bits, the probability space is the set of all \( 2^r \) possible strings of random bits (with uniform distribution).

Monte Carlo algorithms

**Definition 1.** A Monte Carlo algorithm is an algorithm whose output is incorrect with probability at most \( p \).
- A one sided error means that an algorithm’s input is incorrect only on true outputs, or false outputs but not both.
- A false negative Monte Carlo algorithm is always correct when it returns false.

Suppose we have an algorithm \( A \) for a decision problem which:

- If no-instance: returns “no”.
- If yes-instance: returns “yes” with probability \( p \).

Algorithm \( A \) is a one-sided Monte Carlo algorithm with false negatives.

**Problem**

Suppose \( A \) is a one-sided Monte Carlo algorithm with false negatives, that with probability \( p \) returns “yes” when the input is a yes-instance. How can we use \( A \) and design an a new algorithm which ensures a new success probability of a constant \( C \)?

Let \( t = -\frac{\ln(1-C)}{p} \) and repeat \( t \) times. Failure probability is

\[
(1-p)^t \leq (e^{-p})^t = \frac{1}{e^{pt}} = 1 - C
\]

via the inequality \( 1 - x \leq e^{-x} \).
Amplification

**Theorem 2.** If a one-sided error Monte Carlo Algorithm has success probability at least $p$, then repeating it independently $\lceil \frac{1}{p} \rceil$ times gives constant success probability. In particular if $p = \frac{1}{f(k)}$ for some computable function $f$, then we get an FPT one-sided error Monte Carlo Algorithm with additional $f(k)$ overhead in the running time bound.

2 Vertex Cover

For a graph $G = (V, E)$ a vertex cover $X \subseteq V$ is a set of vertices such that every edge is adjacent to a vertex in $X$.

<table>
<thead>
<tr>
<th>Vertex Cover</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Graph $G$, integer $k$</td>
</tr>
<tr>
<td><strong>Parameter:</strong> $k$</td>
</tr>
<tr>
<td><strong>Question:</strong> Does $G$ have a vertex cover of size $k$?</td>
</tr>
</tbody>
</table>

**Theorem 3.** There exists a randomized algorithm that, given a Vertex Cover instance $(G, k)$, in time $2^k n^{O(1)}$ either reports a failure or finds a vertex cover on $k$ vertices in $G$. Moreover, if the algorithm is given a yes-instance, it returns a solution with constant probability.

**Solution**

**Proof.**

- Pick an edge at random and then pick one of the endpoints of that edge with probability $\frac{1}{2}$.
- Repeating this $k$ times finds a vertex cover with probability at least $\frac{1}{2^k}$.
- Applying Theorem 2 gives a randomized FPT running time of $2^k \cdot n^{O(1)}$.

3 Feedback Vertex Set

A feedback vertex set of a multigraph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that $G - S$ is acyclic.

<table>
<thead>
<tr>
<th>Feedback Vertex Set</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Multigraph $G$, integer $k$</td>
</tr>
<tr>
<td><strong>Parameter:</strong> $k$</td>
</tr>
<tr>
<td><strong>Question:</strong> Does $G$ have a feedback vertex of size $k$?</td>
</tr>
</tbody>
</table>

- Recall 5 simplification rules for Feedback Vertex Set.

**Solution:** Simplification

1. Loop: If loop at vertex $v$, remove $v$ and decrease $k$ by 1
2. Multiedge: Remove all edges of multiplicity greater than 2, to exactly 2.
3. Degree-1: If $v$ has degree at most 1 then remove $v$.
4. Degree-2: If $v$ has degree 2 with neighbors $u, w$ then delete 2 edges $uv, vw$ and replace with new edge $uw$.
5. Budget: If $k < 0$, terminate algorithm and return no.

Refer to Lecture 6 for soundness of simplification rules.

**Lemma 4.** Let $G$ be a multigraph on $n$ vertices, with minimum degree at least 3. Then, for every feedback vertex set $X$ of $G$, at least $1/3$ of the edges have at least one end point in $X$.

**Proof.** The graph $G$ has minimum degree 3, this means it has at least $3n/2$ edges. Let $G \setminus X = F$ be the forest that remains. There at most $n - 1$ edges in the forest $F$. This means that at least $\frac{1}{3}$ of the edges are in $X$. 

\[ \square \]
Random Algorithm

**Theorem 5.** There is a randomized algorithm that, given a Feedback Vertex Set instance \((G, k)\), in time \(6^k n^{O(1)}\) either reports a failure or finds a feedback vertex set in \(G\) of at most \(k\). Moreover, if the algorithm is given a yes-instance, it returns a solution with constant probability.

**Solution**

**Proof.**

- First apply simplification rules 1-5 in order to obtain a multigraph \(G'\) with minimum degree at least 3 and we wish to find feedback vertex set \(X'\) of size \(k'\).

- Lemma 4 implies with probability greater than \(\frac{1}{3}\), a randomly chosen edge \(e\) has at least one endpoint in \(X'\).

- By inductive process, a recursive call finds a feedback vertex set in graph \(G' - \{v\}\) of size \(k' - 1\) with probability \((\frac{1}{6})^{k-1}\). Hence \(X'\) can be found with probability at least \((\frac{1}{6})^k\).

- Applying Theorem 2 gives a randomized FPT running time of \(6^k \cdot n^{O(1)}\).

**Lemma 6.** Let \(G\) be a multigraph on \(n\) vertices, with minimum degree 3. For every feedback vertex set \(X\), then at least \(\frac{1}{2}\) of the edges of \(G\) have at least one endpoint in \(X\).

**Hint:** Let \(H = G - X\) be a forest. The statement is equivalent to:

\[
|E(G) \setminus E(H)| > |V(H)| > |E(H)|
\]

Let \(J \subseteq E(G)\) denote edges with one endpoint in \(X\), and the other in \(V(H)\). Show:

\[
|J| > |V(H)|
\]

**Solution**

**Proof.**

- Let \(V_{\leq 1}, V_2, V_{\geq 3}\) be set of vertices that have degree at most 1, exactly 2, and at least 3 respectively in \(H\).

- Since \(G\) has min degree 3 then each vertex in \(V_{\leq 1}\) contributes at least 2 edges to \(J\). Each vertex \(V_2\) contributes at least 1 edge to \(J\).

- Note \(H\) is a forest, we inductively show \(|V_{\geq 3}| < |V_{\leq 1}|\).

  - Trivially true for empty forest and single vertex.
  
- Assume true for forests of size \(n - 1\), i.e. \(|V'_{\geq 3}| < |V'_{\leq 1}|\)

- For any forest of size \(n\), consider removing a leaf (which must always exist). If \(|V_{\geq 3}| = \frac{|V'_{\geq 3}| + 1}{2}\) then \(|V_{\leq 1}| = |V'_{\leq 1}| + 1\).

- This results in:

\[
|E(G) \setminus E(H)| \geq |J| \geq 2|V_{\leq 1}| + |V_2| > |V_{\leq 1}| + |V_2| + |V_{\geq 3}| = |V(H)|
\]

**Random Algorithm 2**

**Lemma 7.** There exists a randomized algorithm that, given a Feedback Vertex Set instance \((G,k)\), in time \(4^k n^{O(1)}\) either reports a failure or finds a path on \(k\) vertices in \(G\). Moreover, if the algorithm is given a yes-instance, it returns a solution with constant probability.

**Corollary 8.** Given a Feedback Vertex Set instance \((G,k)\), in time \(4^k n^{O(1)}\) there is an algorithm that either reports a failure or if given a yes-instance finds a feedback vertex set in \(G\) of size at most \(k\) with constant probability.
4 Color Coding

Longest Path
A simple path is a sequence of edges which connect a sequence of distinct vertices.

<table>
<thead>
<tr>
<th>Longest Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
</tr>
<tr>
<td>Parameter:</td>
</tr>
<tr>
<td>Question:</td>
</tr>
</tbody>
</table>

Problem

- Show that Longest Path is NP-hard.

Reduction from Hamiltonian Path with \( k = n - 1 \).

Color Coding

Lemma 9. Let \( U \) be a set of size \( n \), and let \( X \subseteq U \) be a subset of size \( k \). Let \( \chi : U \to [k] \) be a coloring of the elements of \( U \), chosen uniformly at random. Then the probability that the elements of \( X \) are colored with pairwise distinct colors is at least \( e^{-k} \).

Proof. There are \( k^n \) possible colorings \( \chi \) and \( k!k^{n-k} \) of them are injective on \( X \). The lemma follows from the inequality

\[
 k! > (k/e)^k.
\]

Colorful Path

A path is colorful if all vertices of the path are colored with pairwise distinct colors.

Lemma 10. Let \( G \) be an undirected graph, and let \( \chi : V(G) \to [k] \) be a coloring of its vertices with \( k \) colors. There exists a deterministic algorithm that checks in time \( 2^kn^{O(1)} \) whether \( G \) contains a colorful path on \( k \) vertices and, if this is the case, returns one such path.

Solution

Proof. Parition \( V(G) \) into \( V_1, \ldots, V_k \) subsets such that vertices in \( V_i \) are colored \( i \).

Apply dynamic programming on nonempty \( S \subseteq \{1, \ldots, k\} \). For \( u \in \bigcup_{i \in S} V_i \) let \( P(S,u) = true \) if there is a colorful path with colors from \( S \) and \( u \) as an endpoint. We have the following:

- For \( |S| = 1 \), \( P(S,u) = true \) for \( u \in V(G) \) iff \( S = \{\chi(u)\} \).
- For \( |S| > 1 \)

\[
P(S,u) = \begin{cases} 
  \bigvee_{uv \in E(G)} P(S \setminus \{\chi(u)\}, v) & \text{if } \chi(u) \in S \\
  false & \text{otherwise}
\end{cases}
\]

All values of \( P \) can be computed in \( 2^kn^{O(1)} \) time and there exists a colorful \( k \)-path iff \( P([k], v) \) is true for some vertex \( v \in V(G) \).

Longest Path

Theorem 11. There exists a randomized algorithm that, given a Longest Path instance \((G,k)\), in time \( (2e)^kn^{O(1)} \) either reports a failure or finds a path on \( k \) vertices in \( G \). Moreover, if the algorithm is given a yes-instance, it returns a solution with constant probability.
5 Monotone Local Search

Exact Exponential Algorithms vs Parameterized Algorithms

Exact Exponential Algorithms
- Find exact solutions with respect to parameter \( n \), the input size.
- Feedback Vertex set \( O(1.7347^n) \) [Fomin, Todinca and Villanger 2015]
- Running Time: \( O(\alpha^n n^{O(1)}) \)

Parameterized Algorithms
- Include parameter \( k \), commonly the solution size.
- Feedback Vertex Set: \( O(3.592^k) \) [Kociumaka and Pilipczuk 2013]
- Running Time: \( O(f(k) \cdot n^{O(1)}) \)

Can we use Parameterized Algorithms to design fast Exact Exponential Algorithms?

Subset Problems

An *implicit set system* is a function \( \Phi \) with:
- Input: instance \( I \in \{0,1\}^* \), \(|I| = N\)
- Output: set system \((U_I, \mathcal{F}_I)\):
  - universe \( U_I \), \(|U_I| = n\)
  - family \( \mathcal{F}_I \) of subsets of \( U_I \)

### \( \Phi \)-Subset

**Input:** Instance \( I \)

**Question:** Is \(|\mathcal{F}_I| > 0\)

### \( \Phi \)-Extension

**Input:** Instance \( I \), a set \( X \subseteq U_I \), and an integer \( k \)

**Question:** Does there exist a subset \( S \subseteq (U_I \setminus X) \) such that \( S \cup X \in \mathcal{F}_I \) and \(|S| \leq k\)?

Algorithm

Suppose \( \Phi \)-Extension has a \( O^*(c^k) \) time algorithm \( B \).

**Algorithm for checking whether contains a set of size \( k \)**

- Set \( t = \max \left( 0, \frac{ck-n}{c-1} \right) \)
- Uniformly at random select a subset \( X \subseteq U_I \) of size \( t \)
- Run \( B(I, X, k-t) \)

Running time: [Fomin, Gaspers, Lokshtanov & Saurabh 2016]

\[
O^* \left( \frac{\binom{n}{t}}{\binom{k}{t}} \cdot c^{k-t} \right) = O^* \left( 2 - \frac{1}{c} \right)^n
\]

Intuition

**Brute-force randomized algorithm**

- Pick \( k \) elements of the universe one-by-one.
- Suppose \( \mathcal{F}_I \) contains a set of size \( k \).
Success probability:
\[
\frac{k \cdot k - 1}{n \cdot n - 1} \cdot \frac{k - t}{n - t} \cdot \cdots \cdot \frac{2}{n - (k - 2)} \cdot \frac{1}{n - (k - 1)} = \frac{1}{\binom{n}{k}}
\]

Theorem 12. If there exists an algorithm for $\Phi$-EXTENSION with running time $c^k n^{O(1)}$ then there exists a randomized algorithm for $\Phi$-SUBSET with running time $(2 - \frac{1}{c})^n \cdot n^{O(1)}$.

- Can be derandomized at the expense of a multiplicative $2^{o(1)}$ factor in the running time.

Theorem 13. For a graph $G$ there exists a randomized algorithm which finds a smallest feedback vertex set in time $(2 - \frac{1}{3.592})^n \cdot n^{O(1)} = 1.7217^n \cdot n^{O(1)}$.

References

Exercise 1

**1-Regular Deletion**

**Input:** Graph $G = (V, E)$, integer $k$

**Parameter:** $k$

**Question:** Does there exist $X \subseteq V$ with $|X| \leq k$ such that $G - X$ is 1-regular?

- Design a randomized FPT algorithm with running time $O^*(4^k)$

**Solution 1**

- If there is a vertex with degree 0, then remove it and reduce $k$ by 1.
- If $v$ has degree 1, remove all vertices at distance at most 2 from $v$, and reducing $k$ by the number of vertices at distance 2 from $v$.
- Graph now has minimum degree 2. If yes-instance then deletion set $X$ is incident to at least $\frac{|E|}{2}$ edges.
- Choose edge at random and then an endpoint of the chosen at at random for a $\frac{1}{4}$ probability of selecting a vertex in $X$.

Exercise 2

**Triangle Packing**

**Input:** Graph $G$, integer $k$

**Parameter:** $k$

**Question:** Does $G$ have $k$-vertex disjoint triangles?

- Design a randomized FPT algorithm for Triangle Packing.
Solution 2

- By considering a random $3k$ coloring $\chi$ of the vertices, Lemma 9 provides an algorithm to return a subset $X$ of size $3k$ are pairwise distinct with $e^{-3k}$ success probability.

- For a graph $G$ and coloring $\chi : V(G) \rightarrow [3k]$, in a similar manner to Lemma 10 we design an algorithm that checks whether $G$ contains a triangle packing on $3k$ vertices such that all vertices are pairwise distinctly colored. We do the following:
  - Enumerate though all possible ways of partitioning $3k$ colors into $k$ bags of exactly 3 colors each. There are exactly $\frac{3^k}{(3!)^k}$ of these ways.
  - For a bag, let these colors be $i, j, k$ and consider the vertex partition $V_i, V_j, V_k$. Using these vertices we check if there exists a triangle using vertices from $V_i \cup V_j \cup V_k$ such that each vertex is a different color. This can be computed in time $n^3$. Repeating this for all $k$ bags only requires $k \cdot n^3$ time.
  - Running time of this algorithm is still FPT.