8a. Randomized Algorithms

Serge Gaspers

19T3

Contents

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

1 Introduction

Randomized Algorithms

• Turing machines do not inherently have access to randomness.
• Assume algorithm has also access to a stream of random bits drawn uniformly at random.
• With $r$ random bits, the probability space is the set of all $2^r$ possible strings of random bits (with uniform distribution).

Las Vegas algorithms

Definition 1. A Las Vegas algorithm is a randomized algorithm whose output is always correct. Randomness is used to upper bound the expected running time of the algorithm.

Example
Quicksort with random choice of pivot.

Monte Carlo algorithms

Definition 2. A Monte Carlo algorithm is an algorithm whose output is incorrect with probability at most $p$, $0 < p < 1$.

• A Monte Carlo has one sided error if its output is incorrect only on Yes-instances or on No-instances, but not both.

• A one-sided error Monte Carlo algorithm with false negatives answers No for every No-instance, and answers Yes on Yes-instances with probability $p \in (0, 1)$. We say that $p$ is the success probability of the algorithm.

Boosting success probability
Suppose $A$ is a one-sided Monte Carlo algorithm with false negatives with success probability $p$. How can we use $A$ to design a new one-sided Monte Carlo algorithm with success probability $p^* > p$?
Let \( t = \frac{\ln(1-p^\star)}{p} \) and run the algorithm \( t \) times. Return Yes if at least one run of the algorithm returned Yes, and No otherwise. Failure probability is

\[
(1-p)^t \leq (e^{-p})^t = e^{-p \cdot t} = e^{\ln(1-p^\star)} = 1 - p^\star
\]

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

**Definition 3.** A **randomized algorithm** is a one-sided Monte Carlo algorithm with constant success probability.

**Amplification**

**Theorem 4.** 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 we have a polynomial-time one-sided error Monte Carlo algorithm with success probability \( p = \frac{1}{f(k)} \) for some computable function \( f \), then we get a randomized FPT algorithm with running time \( O^*(f(k)) \).

## 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 \( X \).

**Warm-up:** design a randomized algorithm with running time \( O^*(2^k) \).

**Algorithm** \( rvc(G = (V, E), k) \)

\[
S \leftarrow \emptyset \\
\text{while } k > 0 \text{ and } E \neq \emptyset \text{ do} \\
\quad \text{Select an edge } uv \in E \text{ uniformly at random} \\
\quad \text{Select an endpoint } w \in \{u, v\} \text{ uniformly at random} \\
\quad S \leftarrow S \cup \{w\} \\
\quad G \leftarrow G - w \\
\quad k \leftarrow k - 1 \\
\text{if } S \text{ is a vertex cover of } G \text{ then} \\
\quad \text{return Yes} \\
\text{else} \\
\quad \text{return No}
\]

**Success probability**

- Let \( C \) be a minimal vertex cover of \( G \) of size \( k \)
- What is the probability that Algorithm \( rvc \) returns \( C \)?
- When it selects an edge \( uv \in E \), we have that \( \{u, v\} \cap C \neq \emptyset \)
- When it selects a random endpoint \( w \in \{u, v\} \), we have that \( w \in C \) with probability \( \geq 1/2 \)
- It finds \( C \) with probability at least \( 1/2^k \)

**Theorem 5.** **Vertex Cover** has a randomized algorithm with running time \( O^*(2^k) \).

**Proof.**
- If \( G \) has vertex cover number at most \( k \), then Algorithm \( rvc \) finds one with probability at least \( \frac{1}{2^k} \).
- Applying Theorem 4 gives a randomized FPT running time of \( O^*(2^k) \).
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>Input: Multigraph $G$, integer $k$</td>
</tr>
<tr>
<td>Parameter: $k$</td>
</tr>
<tr>
<td>Question: Does $G$ have a feedback vertex of size $k$?</td>
</tr>
</tbody>
</table>

Recall the following simplification rules for Feedback Vertex Set.

**Simplification Rules**

1. Loop: If loop at vertex $v$, remove $v$ and decrease $k$ by 1.
2. Multiedge: Reduce the multiplicity of each edge with multiplicity $\geq 3$ to 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 $uw$.

The solution is incident to a constant fraction of the edges

**Lemma 6.** Let $G$ be a multigraph 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 endpoint in $X$.

**Proof.** Denote by $n$ and $m$ the number of vertices and edges of $G$, respectively. Since $\delta(G) \geq 3$, we have that $m \geq 3n/2$. Let $F := G - X$. Since $F$ has at most $n - 1$ edges, at least $1/3$ of the edges have an endpoint in $X$. 

**Randomized Algorithm**

**Theorem 7.** Feedback Vertex Set has a randomized algorithm with running time $O^*(6^k)$.

We prove the theorem using the following algorithm.

- $S \leftarrow \emptyset$
- Do $k$ times: Apply simplification rules; add a random endpoint of a random edge to $S$.
- If $S$ is a feedback vertex set, return Yes, otherwise return No.

**Proof.** We need to show: each time the algorithm adds a vertex $v$ to $S$, if $(G - S, k - |S|)$ is a Yes-instance, then with probability at least 1/6, the instance $(G - (S \cup \{v\}), k - |S| - 1)$ is also a Yes-instance. Then, by induction, we can conclude that with probability $1/(6^k)$, the algorithm finds a feedback vertex set of size at most $k$ if it is given a Yes-instance.

- Assume $(G - S, k - |S|)$ is a Yes-instance.
- Lemma 6 implies that with probability at least 1/3, a randomly chosen edge $uv$ has at least one endpoint in some feedback vertex set of size $k - |S|$.
- So, with probability at least $\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$, a randomly chosen endpoint of $uv$ belongs some feedback vertex set of size $\leq k - |S|$.
- Applying Theorem 4 gives a randomized FPT running time of $O^*(6^k)$.
Lemma 8. Let $G$ be a multigraph with minimum degree at least 3. For every feedback vertex set $X$, at least $1/2$ of the edges of $G$ have at least one endpoint in $X$.

Note: For a feedback vertex set $X$, consider the forest $F := G - X$. The statement is equivalent to:

$$|E(G) \setminus E(F)| \geq |E(F)|$$

Let $J \subseteq E(G)$ denote the edges with one endpoint in $X$, and the other in $V(F)$. We will show the stronger result:

$$|J| \geq |V(F)|$$

Proof.  
• Let $V_{\leq 1}, V_2, V_{\geq 3}$ be the set of vertices that have degree at most 1, exactly 2, and at least 3, respectively, in $F$.

• Since $\delta(G) \geq 3$, each vertex in $V_{\leq 1}$ contributes at least 2 edges to $J$, and each vertex in $V_2$ contributes at least 1 edge to $J$.

• We show that $|V_{\geq 3}| \leq |V_{\leq 1}|$ by induction on $|V(F)|$.
  
  – Trivially true for forests with at most 1 vertex.
  
  – Assume true for forests with at most $n - 1$ vertices.
  
  – For any forest on $n$ vertices, consider removing a leaf (which must always exist) to obtain $F'$ with the vertex partition $(V'_{\leq 1}, V'_2, V'_{\geq 3})$. If $|V_{\geq 3}| = |V'_{\geq 3}|$, then we have that $|V_{\geq 3}| = |V'_{\geq 3}| \leq |V'_{\leq 1}| \leq |V_{\leq 1}|$.

• Otherwise, $|V_{\geq 3}| = |V'_{\geq 3}| + 1 \leq |V'_{\leq 1}| + 1 = |V_{\leq 1}|$.

• We conclude that:

$$|E(G) \setminus E(F)| \geq |J| \geq 2|V_{\leq 1}| + |V_2| \geq |V_{\leq 1}| + |V_2| + |V_{\geq 3}| = |V(F)|$$

Improved Randomized Algorithm

Theorem 9. Feedback Vertex Set has a randomized algorithm with running time $O^*(4^k)$.

Note
This algorithmic method is applicable whenever the vertex set we seek is incident to a constant fraction of the edges.

4 Color Coding

Longest Path

```
| Longest Path |
| Input:     Graph $G$, integer $k$ |
| Parameter: $k$ |
| Question:  Does $G$ have a path on $k$ vertices as a subgraph? |
```

NP-complete
To show that Longest Path is NP-hard, reduce from Hamiltonian Path by setting $k = n$ and leaving the graph unchanged.
**Color Coding**

**Notation:** $[k] = \{1, 2, \ldots, k\}$

**Lemma 10.** Let $U$ be a set of size $n$, and let $X \subseteq U$ be a subset of size $k$. Let $\chi : U \rightarrow [k]$ 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 $X$. Using the inequality $k! > \left(\frac{k}{e}\right)^k$, the lemma follows since

$$\frac{k! \cdot k^{n-k}}{k^n} > \frac{k! \cdot k^{n-k}}{e^k \cdot k^n} = e^{-k}.$$ 

\[\square\]

**Colorful Path**

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

**Lemma 11.** Let $G$ be an undirected graph, and let $\chi : V(G) \rightarrow [k]$ be a coloring of its vertices with $k$ colors. There is an algorithm that checks in time $O^*(2^k)$ whether $G$ contains a colorful path on $k$ vertices.

**Proof.** Partition $V(G)$ 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 $O^*(2^k)$ time and there exists a colorful $k$-path iff $P([k], v)$ is true for some vertex $v \in V(G)$. 

\[\square\]

**Theorem 12.** **Longest Path** has a randomized algorithm with running time $O^*(2^{k \cdot (\log_2 2)k})$.

**Note**

This algorithmic method is applicable whenever we seek a vertex set $S$ of size $f(k)$ such that $G[S]$ has constant treewidth.

## 5 Monotone Local Search

### Exponential-time algorithms

- Algorithms for NP-hard problems
- Beat brute-force & improve
- Running time measured in the size of the universe $n$
- $O(2^n \cdot n)$, $O(1.5086^n)$, $O(1.0892^n)$

### Parameterized algorithms

- Algorithms for NP-hard problems
- Use a parameter $k$
  (often $k$ is the solution size)
- Algorithms with running time $f(k) \cdot n^c$
- $k^k n^{O(1)}$, $5^k n^{O(1)}$, $O(1.2738^k + kn)$

Can we use Parameterized algorithms to design fast Exponential-time algorithms?
Example: Feedback Vertex Set

$S \subseteq V$ is a feedback vertex set in a graph $G = (V, E)$ if $G - S$ is acyclic.

**Feedback Vertex Set**

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

Parameter: $k$

Question: Does $G$ have a feedback vertex set of size at most $k$?

---

**Exponential-time algorithms**

- $O^\ast(2^n)$ trivial
- $O(1.7548^n)$ [Fom+08]
- $O(1.7347^n)$ [FV10]
- $O(1.7266^n)$ [XN15]

**Parameterized algorithms**

- $O^\ast((17k^4)!)$ [Bod94]
- $O^\ast((2k + 1)^k)$ [DF99]
- $O^\ast((2k + 1)^k)$ [FV10]
- $O^\ast(3.460^k)$ deterministic [IK19]
- $O^\ast(2.7^k)$ randomized [LN19]

**Exponential-time algorithms via parameterized algorithms**

**Binomial coefficients**

$$\arg \max_{0 \leq k \leq n} \binom{n}{k} = n/2 \quad \text{and} \quad \binom{n}{n/2} = \Theta(2^n/\sqrt{n})$$

**Algorithm for Feedback Vertex Set**

- Set $t = 0.60909 \cdot n$
- If $k \leq t$, run $O^\ast(3^k)$ algorithm
- Else check all $\binom{n}{k}$ vertex subsets of size $k$

Running time: $O^\ast \left( \max \left( 3^t, \binom{n}{t} \right) \right) = O^\ast(1.9526^n)$

This approach gives algorithms faster than $O^\ast(2^n)$ for subset problems with a parameterized algorithm faster than $O^\ast(4^k)$.

**Subset Problems**

An implicit set system is a function $\Phi$ with:

- Input: instance $I \in \{0, 1\}^\ast$, $|I| = N$
- Output: set system $(U_I, F_I)$:
  - universe $U_I$, $|U_I| = n$
  - family $F_I$ of subsets of $U_I$

**$\Phi$-SUBSET**

Input: Instance $I$

Question: Is $|F_I| > 0$?

**$\Phi$-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 $S \cup X \in F_I$ and $|S| \leq k$?
Algorithm

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

Algorithm for checking whether $\mathcal{F}_I$ 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: $O^* \left( \binom{n}{k} \cdot c^{k-t} \right) = O^* \left( \frac{1}{c} \right)^n$

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:

\[
\frac{k}{n} \cdot \frac{k-1}{n-1} \cdot \frac{k-t}{n-t} \cdots \frac{2}{n-(k-2)} \cdot \frac{1}{n-(k-1)} = \frac{1}{\binom{n}{k}} \leq \frac{1}{c}
\]

**Theorem 13** ([Fom+19]). If there exists a (randomized) algorithm for $\Phi$-EXTENSION with running time $O^*(c^k)$ then there exists a randomized algorithm for $\Phi$-SUBSET with running time $(2 - \frac{1}{c})^n \cdot N^{O(1)}$.

**Theorem 14** ([Fom+19]). FEEDBACK VERTEX SET has a randomized algorithm with running time $O^* \left( (2 - \frac{1}{2.7})^n \right) \subseteq O(1.6297^n)$.

Derandomization

Derandomization at the expense of a subexponential factor in the running time.

**Theorem 15** ([Fom+19]). If there exists an algorithm for $\Phi$-EXTENSION with running time $O^*(c^k)$ then there exists an algorithm for $\Phi$-SUBSET with running time $(2 - \frac{1}{c})^{n+o(n)} \cdot N^{O(1)}$.

**Theorem 16** ([Fom+19]). FEEDBACK VERTEX SET has an algorithm with running time $O^* \left( (2 - \frac{1}{3.460})^n \right) \subseteq O(1.7110^n)$.

Further Reading

- Chapter 5, Randomized methods in parameterized algorithms by [Cyg+15]
- Exact Algorithms via Monotone Local Search [Fom+19]
References


