10. Randomized Algorithms: color coding and monotone local search

COMP6741: Parameterized and Exact Computation

Edward Lee[1,2]

[1] School of Computer Science and Engineering, UNSW Sydney, Australia
[2] Decision Sciences Group, Data61, CSIRO, Australia

Semester 2, 2017

Outline

Outline

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.

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.

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

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?

Problem

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.

Outline

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.

VERTEX COVER	
Input:	Graph G, integer k
Parameter:	k
Question:	Does G have a vertex cover of size k?

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.

Vertex Cover	
Input:	Graph G, integer k
Parameter:	k
Question:	Does G have a vertex cover of size k?

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 $2^k \cdot n^{O(1)}$.

$\square$

Outline

Feedback Vertex Set

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

Feedback Vertex Set

Input:	Multigraph G, integer k
Parameter:	k
Question:	Does G have a feedback vertex of size k?

Feedback Vertex Set

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

FEEDBACK VERTEX SET	
Input:	Multigraph G, integer k
Parameter:	k
Question:	Does G have a feedback vertex of size k?

- 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

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.

Lemma

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 $G \backslash X = F$ 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. □

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'.

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'. So with probability greater than $\frac{1}{2} \times \frac{1}{3} = \frac{1}{6}$, a randomly chosen endpoint of e belongs to X'.

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'. So with probability greater than $\frac{1}{2} \times \frac{1}{3} = \frac{1}{6}$, a randomly chosen endpoint of e belongs to X'.
- By inductive process, a recursive call finds a feedback vertex set in graph $G' - \{v\}$ of size $k' - 1$ with probability $\left(\frac{1}{6}\right)^{k-1}$. Hence X' can be found with probability at least $\left(\frac{1}{6}\right)^{k}$.

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'. So with probability greater than $\frac{1}{2} \times \frac{1}{3} = \frac{1}{6}$, a randomly chosen endpoint of e belongs to X'.
- By inductive process, a recursive call finds a feedback vertex set in graph $G' - \{v\}$ of size $k' - 1$ with probability $\left(\frac{1}{6}\right)^{k-1}$. Hence X' can be found with probability at least $\left(\frac{1}{6}\right)^{k}$.
- Applying Theorem 2 gives a randomized FPT running time of $6^k \cdot n^{O(1)}$.

□

Lemma 2

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.

Lemma 2

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:

$$|E(G) \backslash E(H)| > |V(H)| > |E(H)|$$

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.

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.

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

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 $|V_{\geq 3}| = |V'_{\geq 3}| + 1$ then $|V_{\leq 1}| = |V'_{\leq 1}| + 1$.
- This results in:

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

$\square$

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

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.

Outline

Longest Path

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

Longest Path

Input:	Graph G, integer k
Parameter:	k
Question:	Does G have a simple path of size k?

Longest Path

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

LONGEST PATH

Input:	Graph G, integer k
Parameter:	k
Question:	Does G have a simple path of size k?

Problem

- Show that LONGEST PATH is NP-hard.

Longest Path

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

LONGEST PATH

Input:	Graph G, integer k
Parameter:	k
Question:	Does G have a simple path of size k?

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}*.*

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 χ and $k!k^{n-k}$ of them are injective on X. The lemma follows from the inequality

$$k! > (k/e)^k.$$

$\square$

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 $2^k n^{\mathcal{O}(1)}$ 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 $V_1, ..., V_k$ subsets such that vertices in V_i are colored i.

Solution

Proof.

Parition $V(G)$ into $V_1, ..., V_k$ subsets such that vertices in V_i are colored i. Apply dynamic programming on nonempty $S \subseteq \{1, ..., k\}$. 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.

Solution

Proof.

Parition $V(G)$ into $V_1, ..., V_k$ subsets such that vertices in V_i are colored i. Apply dynamic programming on nonempty $S \subseteq \{1, ..., k\}$. 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 $|S| > 1$

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

Solution

Proof.

Parition $V(G)$ into $V_1, ..., V_k$ subsets such that vertices in V_i are colored i. Apply dynamic programming on nonempty $S \subseteq \{1, ..., k\}$. 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 $|S| > 1$

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

All values of P can be computed in $2^k n^{O(1)}$ 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* $(2e)^k n^{O(1)}$ *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.*

Outline

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: $O(\alpha^n n^{O(1)})$

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: $O(f(k) \cdot n^{O(1)})$

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 Φ 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

Subset Problems

An *implicit set system* is a function Φ 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

Φ-SUBSET

Input:	Instance I
Question:	Is $|\mathcal{F}_I| > 0$

Subset Problems

An *implicit set system* is a function Φ 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

Φ-Subset

Input:	Instance I
Question:	Is $|\mathcal{F}_I| > 0$

Φ-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 Φ-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 $B(I, X, k-t)$

Algorithm

Suppose Φ-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: [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:

$$\frac{k}{n} \cdot \frac{k-1}{n-1} \cdot \ldots \cdot \underbrace{\frac{k-t}{n-t}}_{\displaystyle \| \atop \displaystyle \frac{1}{c}} \cdot \ldots \cdot \frac{2}{n-(k-2)} \frac{1}{n-(k-1)} = \frac{1}{\binom{n}{k}}$$

Theorem

Theorem 12

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

Theorem

Theorem 12

If there exists an algorithm for Φ-EXTENSION *with running time* $c^k n^{O(1)}$ *then there exists a randomized algorithm for* Φ-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

Theorem 12

If there exists an algorithm for Φ-EXTENSION *with running time* $c^k n^{O(1)}$ *then there exists a randomized algorithm for* Φ-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* $\left(2 - \frac{1}{3.592}\right)^n \cdot n^{O(1)} = 1.7217^n \cdot n^{O(1)}$.

References

- Chapter 5, *Randomized methods in parameterized algorithms* by Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, 2015.
- *Exact Algorithms via Monotone Local Search*, Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, Saket Saurabh. ACM symposium on Theory of Computing, 2016.

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

Solution

- 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 χ 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 $\chi : V(G) \to [3k]$, 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 $\frac{3k!}{(3!)^k k!}$ 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.