Randomized Computation

K. Subramani[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

March 27, 2009

Outline

1. Randomized Algorithms
   - Three paradigmatic problems
   - 2SAT
   - Min-Cut
   - Non-singularity of a Symbolic Matrix
2. Randomized Complexity Classes

Outline

1. Randomized Algorithms
   - Three paradigmatic problems
   - 2SAT
   - Min-Cut
   - Non-singularity of a Symbolic Matrix
2. Randomized Complexity Classes

Outline

1. Randomized Algorithms
   - Three paradigmatic problems
   - 2SAT
   - Min-Cut
   - Non-singularity of a Symbolic Matrix
2. Randomized Complexity Classes

Three paradigmatic problems

How useful is randomized computation?

(i) 2SAT.
(ii) Min-Cut.
(iii) Non-singularity of a symbolic square matrix.

Outline

1. Randomized Algorithms
   - Three paradigmatic problems
   - 2SAT
   - Min-Cut
   - Non-singularity of a Symbolic Matrix
2. Randomized Complexity Classes

Problem Description

Goal

Let $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ denote a boolean formula in CNF over the boolean variables $\{x_1, x_2, \ldots, x_n\}$, such that each clause C_i has exactly two variables. Determine whether ϕ is satisfiable.

Note

2SAT can be solved in $O(m+n)$ time using Tarjan's connected components algorithm. This algorithm is a variant of the reachability method discussed in class.

Problem Description

Goal

Let $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ denote a boolean formula in CNF over the boolean variables $\{x_1, x_2, \ldots, x_n\}$, such that each clause C_i has exactly two variables. Determine whether ϕ is satisfiable.

Note

2SAT can be solved in $O(m + n)$ time using Tarjan's connected components algorithm. This algorithm is a variant of the reachability method discussed in class.

The 2CNF Algorithm

Function SATISFIABILITY-TESTING(ϕ)

```
Start with an arbitrary assignment to the variables.
while (the current assignment is not satisfying) do
    Pick an unsatisfied clause.
    Uniformly and at random flip the value assigned to one of its two literals
    (variables).
end while
```

Algorithm 2.1: Papadimitrious's randomized algorithm for 2CNF Satisfiability

The 2CNF Algorithm

Function SATISFIABILITY-TESTING(ϕ)

1: Start with an arbitrary assignment to the variables.
2: **while** (the current assignment is not satisfying) **do**
3: Pick an unsatisfied clause.
4: Uniformly and at random flip the value assigned to one of its two literals (variables).
5: **end while**

Algorithm 2.2: Papadimitrious's randomized algorithm for 2CNF Satisfiability

The 2CNF Algorithm

```
Function SATISFIABILITY-TESTING(φ)
Start with an arbitrary assignment to the variables.
while (the current assignment is not satisfying) do
    Pick an unsatisfied clause.
    Uniformly and at random flip the value assigned to one of its two literals
    (variables).
end while
```

Algorithm 2.3: Papadimitrious's randomized algorithm for 2CNF Satisfiability

The 2CNF Algorithm

Function SATISFIABILITY-TESTING(ϕ)

1: Start with an arbitrary assignment to the variables.
2: **while** (the current assignment is not satisfying) **do**
3: Pick an unsatisfied clause.
4: Uniformly and at random flip the value assigned to one of its two literals (variables).
5: **end while**

Algorithm 2.4: Papadimitrious's randomized algorithm for 2CNF Satisfiability

The 2CNF Algorithm

Function SATISFIABILITY-TESTING(ϕ)

1: Start with an arbitrary assignment to the variables.
2: **while** (the current assignment is not satisfying) **do**
3: Pick an unsatisfied clause.
4: Uniformly and at random flip the value assigned to one of its two literals (variables).
5: **end while**

Algorithm 2.5: Papadimitrious's randomized algorithm for 2CNF Satisfiability

Mathematical Preliminaries

Theorem

Let X and Y be two random variables. Then $\mathbf{E}[X] = \mathbf{E}[\mathbf{E}[X|Y]]$.

Theorem (Markov)

Let X be a non-negative random variable and let $c > 0$ denote a constant. Then $\mathbf{Pr}(X \geq c \cdot \mathbf{E}[X]) \leq \frac{1}{c}$.

Mathematical Preliminaries

Theorem

Let X and Y be two random variables. Then $\mathbf{E}[X] = \mathbf{E}[\mathbf{E}[X|Y]]$.

Theorem (Markov)

Let X be a non-negative random variable and let $c > 0$ denote a constant. Then $\mathbf{Pr}(X \geq c \cdot \mathbf{E}[X]) \leq \frac{1}{c}$.

Analysis

Modeling as a random walk

Assume that ϕ is satisfiable and focus on a particular satisfying assignment $\hat{T}$. Let T denote the current assignment. We want to bound the expected number of steps before T is transformed into $\hat{T}$.

Let $t(i)$ denote the expected number of flips for T to become $\hat{T}$, assuming that T differs from $\hat{T}$ in exactly i variables. It follows that,

$$\begin{aligned} t(0) &= 0 \\ t(n) &= 1 + t(n-1) \\ t(i) &\leq \frac{1}{2}t(i-1) + \frac{1}{2}t(i+1) + 1, 0 < i < n \end{aligned}$$

Observation

The above system can be solved to get $t(n) \leq n^2$. From Markov's inequality it follows that the probability that T is not transformed into $\hat{T}$ in at most $2 \cdot n^2$ flips is less than one-half. Running time is $O(n^2 \cdot (m+n))$, which is hardly impressive.

Analysis

Modeling as a random walk

Assume that ϕ is satisfiable and focus on a particular satisfying assignment $\hat{T}$. Let T denote the current assignment. We want to bound the expected number of steps before T is transformed into $\hat{T}$.

Let $t(i)$ denote the expected number of flips for T to become $\hat{T}$, assuming that T differs from $\hat{T}$ in exactly i variables. It follows that,

$$\begin{aligned} t(0) &= 0 \\ t(n) &= 1 + t(n-1) \\ t(i) &\leq \frac{1}{2}t(i-1) + \frac{1}{2}t(i+1) + 1, 0 < i < n \end{aligned}$$

Observation

The above system can be solved to get $t(n) \leq n^2$. From Markov's inequality it follows that the probability that T is not transformed into $\hat{T}$ in at most $2 \cdot n^2$ flips is less than one-half. Running time is $O(n^2 \cdot (m+n))$, which is hardly impressive.

Analysis

Modeling as a random walk

Assume that ϕ is satisfiable and focus on a particular satisfying assignment $\hat{T}$. Let T denote the current assignment. We want to bound the expected number of steps before T is transformed into $\hat{T}$.
Let $t(i)$ denote the expected number of flips for T to become $\hat{T}$, assuming that T differs from $\hat{T}$ in exactly i variables. It follows that,

$$\begin{aligned} t(0) &= 0 \\ t(n) &= 1 + t(n-1) \\ t(i) &\leq \frac{1}{2}t(i-1) + \frac{1}{2}t(i+1) + 1, 0 < i < n \end{aligned}$$

Observation

The above system can be solved to get $t(n) \leq n^2$. From Markov's inequality it follows that the probability that T is not transformed into $\hat{T}$ in at most $2 \cdot n^2$ flips is less than one-half. Running time is $O(n^2 \cdot (m+n))$, which is hardly impressive.

Analysis

Modeling as a random walk

Assume that ϕ is satisfiable and focus on a particular satisfying assignment $\hat{T}$. Let T denote the current assignment. We want to bound the expected number of steps before T is transformed into $\hat{T}$.
Let $t(i)$ denote the expected number of flips for T to become $\hat{T}$, assuming that T differs from $\hat{T}$ in exactly i variables. It follows that,

$$\begin{aligned} t(0) &= 0 \\ t(n) &= 1 + t(n-1) \\ t(i) &\leq \frac{1}{2}t(i-1) + \frac{1}{2}t(i+1) + 1, 0 < i < n \end{aligned}$$

Observation

The above system can be solved to get $t(n) \leq n^2$. From Markov's inequality it follows that the probability that T is not transformed into $\hat{T}$ in at most $2 \cdot n^2$ flips is less than one-half. Running time is $O(n^2 \cdot (m+n))$, which is hardly impressive.

Analysis

Modeling as a random walk

Assume that ϕ is satisfiable and focus on a particular satisfying assignment $\hat{T}$. Let T denote the current assignment. We want to bound the expected number of steps before T is transformed into $\hat{T}$.
Let $t(i)$ denote the expected number of flips for T to become $\hat{T}$, assuming that T differs from $\hat{T}$ in exactly i variables. It follows that,

$$\begin{aligned} t(0) &= 0 \\ t(n) &= 1 + t(n-1) \\ t(i) &\leq \frac{1}{2}t(i-1) + \frac{1}{2}t(i+1) + 1, 0 < i < n \end{aligned}$$

Observation

The above system can be solved to get $t(n) \leq n^2$. From Markov's inequality it follows that the probability that T is not transformed into $\hat{T}$ in at most $2 \cdot n^2$ flips is less than one-half. Running time is $O(n^2 \cdot (m+n))$, which is hardly impressive.

Analysis

Modeling as a random walk

Assume that ϕ is satisfiable and focus on a particular satisfying assignment $\hat{T}$. Let T denote the current assignment. We want to bound the expected number of steps before T is transformed into $\hat{T}$.
Let $t(i)$ denote the expected number of flips for T to become $\hat{T}$, assuming that T differs from $\hat{T}$ in exactly i variables. It follows that,

$$\begin{aligned} t(0) &= 0 \\ t(n) &= 1 + t(n-1) \\ t(i) &\leq \frac{1}{2}t(i-1) + \frac{1}{2}t(i+1) + 1, 0 < i < n \end{aligned}$$

Observation

The above system can be solved to get $t(n) \leq n^2$. From Markov's inequality it follows that the probability that T is not transformed into $\hat{T}$ in at most $2 \cdot n^2$ flips is less than one-half. Running time is $O(n^2 \cdot (m+n))$, which is hardly impressive.

Analysis

Modeling as a random walk

Assume that ϕ is satisfiable and focus on a particular satisfying assignment $\hat{T}$. Let T denote the current assignment. We want to bound the expected number of steps before T is transformed into $\hat{T}$.
Let $t(i)$ denote the expected number of flips for T to become $\hat{T}$, assuming that T differs from $\hat{T}$ in exactly i variables. It follows that,

$$\begin{aligned} t(0) &= 0 \\ t(n) &= 1 + t(n-1) \\ t(i) &\leq \frac{1}{2}t(i-1) + \frac{1}{2}t(i+1) + 1, 0 < i < n \end{aligned}$$

Observation

The above system can be solved to get $t(n) \leq n^2$. From Markov's inequality it follows that the probability that T is not transformed into $\hat{T}$ in at most $2 \cdot n^2$ flips is less than one-half. *Running time is $O(n^2 \cdot (m+n))$, which is hardly impressive.*

Analysis

Modeling as a random walk

Assume that ϕ is satisfiable and focus on a particular satisfying assignment $\hat{T}$. Let T denote the current assignment. We want to bound the expected number of steps before T is transformed into $\hat{T}$.
Let $t(i)$ denote the expected number of flips for T to become $\hat{T}$, assuming that T differs from $\hat{T}$ in exactly i variables. It follows that,

$$\begin{aligned} t(0) &= 0 \\ t(n) &= 1 + t(n-1) \\ t(i) &\leq \frac{1}{2}t(i-1) + \frac{1}{2}t(i+1) + 1, 0 < i < n \end{aligned}$$

Observation

The above system can be solved to get $t(n) \leq n^2$. From Markov's inequality it follows that the probability that T is not transformed into $\hat{T}$ in at most $2 \cdot n^2$ flips is less than one-half. Running time is $O(n^2 \cdot (m+n))$, which is hardly impressive.

Outline

1. Randomized Algorithms
   - Three paradigmatic problems
   - 2SAT
   - Min-Cut
   - Non-singularity of a Symbolic Matrix
2. Randomized Complexity Classes

Problem Description

Goal

Given an unweighted, undirected graph $G = \langle, V, E \rangle$, find the smallest cardinality set $E' \subseteq E$, such that $G = \langle V, E - E' \rangle$ has at least two components. *Also called edge connectivity.*

Note

Min-Cut can be solved in polynomial time using network flow techniques.

Observation

The Min-Cut of a graph is no larger than the degree of the smallest degree vertex.

Problem Description

Goal

Given an unweighted, undirected graph $G = \langle, V, E\rangle$, find the smallest cardinality set $E' \subseteq E$, such that $G = \langle V, E - E'\rangle$ has at least two components. Also called edge connectivity.

Note

Min-Cut can be solved in polynomial time using network flow techniques.

Observation

The Min-Cut of a graph is no larger than the degree of the smallest degree vertex.

Problem Description

Goal

Given an unweighted, undirected graph $G = \langle , V, E \rangle$, find the smallest cardinality set $E' \subseteq E$, such that $G = \langle V, E - E' \rangle$ has at least two components. Also called edge connectivity.

Note

Min-Cut can be solved in polynomial time using network flow techniques.

Observation

The Min-Cut of a graph is no larger than the degree of the smallest degree vertex.

Problem Description

Goal

Given an unweighted, undirected graph $G = \langle, V, E\rangle$, find the smallest cardinality set $E' \subseteq E$, such that $G = \langle V, E - E'\rangle$ has at least two components. Also called edge connectivity.

Note

Min-Cut can be solved in polynomial time using network flow techniques.

Observation

The Min-Cut of a graph is no larger than the degree of the smallest degree vertex.

Edge contraction

Procedure

(i) *Identify the vertices corresponding to an edge, i.e., make them into one large vertex.*

(ii) *Remove all self-loops, if formed.*

(iii) *Maintain all parallel edges, if formed.*

Observation

Contracting an edge does not decrease the Min-Cut of a graph.

Edge contraction

Procedure

(i) *Identify the vertices corresponding to an edge, i.e., make them into one large vertex.*
(ii) *Remove all self-loops, if formed.*
(iii) *Maintain all parallel edges, if formed.*

Observation

Contracting an edge does not decrease the Min-Cut of a graph.

Edge contraction

Procedure

(i) *Identify the vertices corresponding to an edge, i.e., make them into one large vertex.*

(ii) *Remove all self-loops, if formed.*

(iii) *Maintain all parallel edges, if formed.*

Observation

Contracting an edge does not decrease the Min-Cut of a graph.

Edge contraction

Procedure

(i) *Identify the vertices corresponding to an edge, i.e., make them into one large vertex.*

(ii) *Remove all self-loops, if formed.*

(iii) *Maintain all parallel edges, if formed.*

Observation

Contracting an edge does not decrease the Min-Cut of a graph.

Edge contraction

Procedure

(i) *Identify the vertices corresponding to an edge, i.e., make them into one large vertex.*

(ii) *Remove all self-loops, if formed.*

(iii) *Maintain all parallel edges, if formed.*

Observation

Contracting an edge does not decrease the Min-Cut of a graph.

The Min-Cut Algorithm

Function MIN-CUT($G = \langle (V, E \rangle)$)

while (G has more than 2 vertices do) **do**
    Select an edge uniformly and at random, and contract it.
end while
return(The cut determined by the two remaining vertices)

Algorithm 2.6: Karger's Min-Cut Algorithm

The Min-Cut Algorithm

```
Function MIN-CUT(G = ⟨(V, E⟩)
  while (G has more than 2 vertices do) do
      Select an edge uniformly and at random, and contract it.
  end while
  return(The cut determined by the two remaining vertices)
```

Algorithm 2.7: Karger's Min-Cut Algorithm

The Min-Cut Algorithm

Function MIN-CUT($G = \langle (V, E \rangle)$)

1: **while** (G has more than 2 vertices do) **do**
2: Select an edge uniformly and at random, and contract it.
3: **end while**
4: **return**(The cut determined by the two remaining vertices)

Algorithm 2.8: Karger's Min-Cut Algorithm

Mathematical Preliminaries

Theorem

Let $E_1, E_2, \ldots, E_k$ denote a collection of k events on some sample space. Then

$$\mathbf{Pr}(\cap_{i=1}^{n} E_i) = \mathbf{Pr}(E_1) \times \mathbf{Pr}(E_2|E_1) \times \mathbf{Pr}(E_3|(E_1 \cap E_2) \ldots \times \mathbf{Pr}(E_k| \cap_{i=1}^{k-1} E_i).$$

Proof.

By definition,

$$\mathbf{Pr}(E_2|E_1) = \frac{\mathbf{Pr}(E_1 \cap E_2)}{\mathbf{Pr}(E_1)}$$

$$\Rightarrow \mathbf{Pr}(E_1 \cap E_2) = \mathbf{Pr}(E_1) \cdot \mathbf{Pr}(E_2|E_1).$$

Now use induction! □

Mathematical Preliminaries

Theorem

Let $E_1, E_2, \ldots, E_k$ denote a collection of k events on some sample space. Then

$$\mathbf{Pr}(\cap_{i=1}^{n} E_i) = \mathbf{Pr}(E_1) \times \mathbf{Pr}(E_2|E_1) \times \mathbf{Pr}(E_3|(E_1 \cap E_2) \ldots \times \mathbf{Pr}(E_k| \cap_{i=1}^{k-1} E_i).$$

Proof.

By definition,

$$\begin{aligned} \mathbf{Pr}(E_2|E_1) &= \frac{\mathbf{Pr}(E_1 \cap E_2)}{\mathbf{Pr}(E_1)} \\ \Rightarrow \mathbf{Pr}(E_1 \cap E_2) &= \mathbf{Pr}(E_1) \cdot \mathbf{Pr}(E_2|E_1). \end{aligned}$$

Now use induction! □

Mathematical Preliminaries

Theorem

Let $E_1, E_2, \ldots, E_k$ denote a collection of k events on some sample space. Then

$$\mathbf{Pr}(\cap_{i=1}^{n} E_i) = \mathbf{Pr}(E_1) \times \mathbf{Pr}(E_2|E_1) \times \mathbf{Pr}(E_3|(E_1 \cap E_2) \ldots \times \mathbf{Pr}(E_k| \cap_{i=1}^{k-1} E_i).$$

Proof.

By definition,

$$\begin{aligned} \mathbf{Pr}(E_2|E_1) &= \frac{\mathbf{Pr}(E_1 \cap E_2)}{\mathbf{Pr}(E_1)} \\ \Rightarrow \mathbf{Pr}(E_1 \cap E_2) &= \mathbf{Pr}(E_1) \cdot \mathbf{Pr}(E_2|E_1). \end{aligned}$$

Now use induction! □

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least $\frac{kn}{2}$ edges.*

(iii) *Let E_i denote the event that no edge of C is picked for contraction during the i^{th} iteration.*

(iv) *Thus, $E = \cap_{i=1}^{n-1} E_i$ denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round 1 is in C is at most $\frac{k}{\frac{kn}{2}}$.*

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound $\mathbf{Pr}(E_2|E_1)$. If E_1 has occurred, then after round 1, the graph has at least $\frac{k\cdot(n-1)}{2}$ edges. $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.*

(vii) *Working in identical fashion, $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.*

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least $\frac{kn}{2}$ edges.*

(iii) *Let E_i denote the event that no edge of C is picked for contraction during the i^{th} iteration.*

(iv) *Thus, $E = \cap_{i=1}^{n-1} E_i$ denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round 1 is in C is at most $\frac{k}{\frac{kn}{2}}$.*

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound $\mathbf{Pr}(E_2|E_1)$. If E_1 has occurred, then after round 1, the graph has at least $\frac{k \cdot (n-1)}{2}$ edges. $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.*

(vii) *Working in identical fashion, $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.*

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least* $\frac{kn}{2}$ *edges.*

(iii) *Let* E_i *denote the event that no edge of C is picked for contraction during the* i^{th} *iteration.*

(iv) *Thus,* $E = \cap_{i=1}^{n-1} E_i$ *denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round* 1 *is in C is at most* $\frac{k}{\frac{kn}{2}}$. $\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound* $\mathbf{Pr}(E_2|E_1)$. *If* E_1 *has occurred, then after round* 1, *the graph has at least* $\frac{k \cdot (n-1)}{2}$ *edges.* $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.

(vii) *Working in identical fashion,* $\mathbf{Pr}(E_i | \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least $\frac{kn}{2}$ edges.*

(iii) *Let E_i denote the event that no edge of C is picked for contraction during the i^{th} iteration.*

(iv) *Thus, $E = \cap_{i=1}^{n-1} E_i$ denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round 1 is in C is at most $\frac{k}{\frac{kn}{2}}$.*

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound $\mathbf{Pr}(E_2|E_1)$. If E_1 has occurred, then after round 1, the graph has at least $\frac{k \cdot (n-1)}{2}$ edges. $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.*

(vii) *Working in identical fashion, $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.*

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least* $\frac{kn}{2}$ *edges.*

(iii) *Let* E_i *denote the event that no edge of C is picked for contraction during the* i^{th} *iteration.*

(iv) *Thus,* $E = \cap_{i=1}^{n-1} E_i$ *denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round* 1 *is in C is at most* $\frac{k}{\frac{kn}{2}}$.

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound* $\mathbf{Pr}(E_2|E_1)$. *If* E_1 *has occurred, then after round* 1, *the graph has at least* $\frac{k\cdot(n-1)}{2}$ *edges.* $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.

(vii) *Working in identical fashion,* $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least* $\frac{kn}{2}$ *edges.*

(iii) *Let* E_i *denote the event that no edge of C is picked for contraction during the* i^{th} *iteration.*

(iv) *Thus,* $E = \cap_{i=1}^{n-1} E_i$ *denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round 1 is in C is at most* $\frac{k}{\frac{kn}{2}}$.

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound* $\mathbf{Pr}(E_2|E_1)$. *If* E_1 *has occurred, then after round 1, the graph has at least* $\frac{k\cdot(n-1)}{2}$ *edges.* $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.

(vii) *Working in identical fashion,* $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least $\frac{kn}{2}$ edges.*

(iii) *Let E_i denote the event that no edge of C is picked for contraction during the i^{th} iteration.*

(iv) *Thus, $E = \cap_{i=1}^{n-1} E_i$ denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round 1 is in C is at most $\frac{k}{\frac{kn}{2}}$.*

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound $\mathbf{Pr}(E_2|E_1)$. If E_1 has occurred, then after round 1, the graph has at least $\frac{k\cdot(n-1)}{2}$ edges. $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.*

(vii) *Working in identical fashion, $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.*

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least $\frac{kn}{2}$ edges.*

(iii) *Let E_i denote the event that no edge of C is picked for contraction during the i^{th} iteration.*

(iv) *Thus, $E = \cap_{i=1}^{n-1} E_i$ denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round 1 is in C is at most $\frac{k}{\frac{kn}{2}}$.*

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound $\mathbf{Pr}(E_2|E_1)$. If E_1 has occurred, then after round 1, the graph has at least $\frac{k\cdot(n-1)}{2}$ edges.* $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.

(vii) *Working in identical fashion, $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.*

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least $\frac{kn}{2}$ edges.*

(iii) *Let E_i denote the event that no edge of C is picked for contraction during the i^{th} iteration.*

(iv) *Thus, $E = \cap_{i=1}^{n-1} E_i$ denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round 1 is in C is at most $\frac{k}{\frac{kn}{2}}$.*

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound $\mathbf{Pr}(E_2|E_1)$. If E_1 has occurred, then after round 1, the graph has at least $\frac{k\cdot(n-1)}{2}$ edges. $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.*

(vii) *Working in identical fashion, $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.*

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least $\frac{kn}{2}$ edges.*

(iii) *Let E_i denote the event that no edge of C is picked for contraction during the i^{th} iteration.*

(iv) *Thus, $E = \cap_{i=1}^{n-1} E_i$ denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round 1 is in C is at most $\frac{k}{\frac{kn}{2}}$.*

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound $\mathbf{Pr}(E_2|E_1)$. If E_1 has occurred, then after round 1, the graph has at least $\frac{k\cdot(n-1)}{2}$ edges. $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.*

(vii) *Working in identical fashion, $\mathbf{Pr}(E_i| \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.*

Analysis

Steps

(i) *Focus on a specific Min-Cut C of G having exactly k edges.*

(ii) *Clearly G must have at least* $\frac{kn}{2}$ *edges.*

(iii) *Let* E_i *denote the event that no edge of C is picked for contraction during the* i^{th} *iteration.*

(iv) *Thus,* $E = \cap_{i=1}^{n-1} E_i$ *denotes the event that no edge of C is touched, i.e., the cut C survives.*

(v) *The probability that an edge picked randomly in round* 1 *is in C is at most* $\frac{k}{\frac{kn}{2}}$.

$\mathbf{Pr}(E_1) \geq (1 - \frac{2}{n})$.

(vi) *Let us now bound* $\mathbf{Pr}(E_2|E_1)$. *If* E_1 *has occurred, then after round* 1, *the graph has at least* $\frac{k\cdot(n-1)}{2}$ *edges.* $\Rightarrow \mathbf{Pr}(E_2|E_1) \geq (1 - \frac{2}{n-1})$.

(vii) *Working in identical fashion,* $\mathbf{Pr}(E_i | \cap_{j=1}^{i-1} E_j) \geq (1 - \frac{2}{(n-i+1)})$.

Analysis (contd.)

Steps

(i) *It follows that*

$$\begin{aligned}\mathbf{Pr}(E) &\geq \mathbf{Pr}(\cap_{i=1}^{n-2} E_i)\\ &= \Pi_{i=1}^{n-2}(1 - \frac{2}{(n-i+1)})\\ &= \frac{2}{n\cdot(n-1)}\\ &\geq \frac{2}{n^2}\end{aligned}$$

(ii) *Thus, the probability that C survives all the contractions is at least* $\frac{2}{n^2}$.

(iii) *Thus, the probability that C does not survive all the contractions is at most* $(1-\frac{2}{n^2})$.

Analysis (contd.)

Steps

(i) *It follows that*

$$\begin{aligned}\mathbf{Pr}(E) &\geq \mathbf{Pr}(\cap_{i=1}^{n-2} E_i)\\ &= \Pi_{i=1}^{n-2}(1 - \frac{2}{(n-i+1)})\\ &= \frac{2}{n \cdot (n-1)}\\ &\geq \frac{2}{n^2}\end{aligned}$$

(ii) *Thus, the probability that C survives all the contractions is at least* $\frac{2}{n^2}$.

(iii) *Thus, the probability that C does not survive all the contractions is at most* $(1 - \frac{2}{n^2})$.

Analysis (contd.)

Steps

(i) *It follows that*

$$\begin{aligned}
\mathbf{Pr}(E) &\geq \mathbf{Pr}(\cap_{i=1}^{n-2} E_i) \\
&= \Pi_{i=1}^{n-2}(1 - \frac{2}{(n-i+1)}) \\
&= \frac{2}{n \cdot (n-1)} \\
&\geq \frac{2}{n^2}
\end{aligned}$$

(ii) *Thus, the probability that C survives all the contractions is at least* $\frac{2}{n^2}$.

(iii) *Thus, the probability that C does not survive all the contractions is at most* $(1 - \frac{2}{n^2})$.

Analysis (contd.)

Steps

(i) *It follows that*

$$\begin{aligned}\mathbf{Pr}(E) &\geq \mathbf{Pr}(\cap_{i=1}^{n-2} E_i)\\ &= \Pi_{i=1}^{n-2}(1-\frac{2}{(n-i+1)})\\ &= \frac{2}{n\cdot(n-1)}\\ &\geq \frac{2}{n^2}\end{aligned}$$

(ii) *Thus, the probability that C survives all the contractions is at least $\frac{2}{n^2}$.*

(iii) *Thus, the probability that C does not survive all the contractions is at most $(1-\frac{2}{n^2})$.*

Analysis (contd.)

Steps

(i) *It follows that*

$$\begin{aligned}
\mathbf{Pr}(E) &\geq \mathbf{Pr}(\cap_{i=1}^{n-2} E_i) \\
&= \Pi_{i=1}^{n-2}\left(1 - \frac{2}{(n-i+1)}\right) \\
&= \frac{2}{n \cdot (n-1)} \\
&\geq \frac{2}{n^2}
\end{aligned}$$

(ii) *Thus, the probability that C survives all the contractions is at least* $\frac{2}{n^2}$.

(iii) *Thus, the probability that C does not survive all the contractions is at most* $(1 - \frac{2}{n^2})$.

Analysis (contd.)

Steps

(i) *It follows that*

$$
\begin{aligned}
\mathbf{Pr}(E) &\geq \mathbf{Pr}(\cap_{i=1}^{n-2} E_i) \\
&= \Pi_{i=1}^{n-2}(1 - \frac{2}{(n-i+1)}) \\
&= \frac{2}{n \cdot (n-1)} \\
&\geq \frac{2}{n^2}
\end{aligned}
$$

(ii) *Thus, the probability that C survives all the contractions is at least $\frac{2}{n^2}$.*

(iii) *Thus, the probability that C does not survive all the contractions is at most $(1 - \frac{2}{n^2})$.*

Analysis (contd.)

Observation

If Karger's algorithm is run $\frac{n^2}{2}$ times on the same graph, the probability that C does not survive any of the runs is at most $(1-\frac{2}{n^2})^{\frac{n^2}{2}} < \frac{1}{e}$.
In other words, the probability that C is obtained after $\frac{n^2}{2}$ runs is at least $(1-\frac{1}{e})$.

Note

Karger's algorithm is both simpler and faster than any deterministic algorithm for determining the Min-Cut of an undirected graph.

Analysis (contd.)

Observation

If Karger's algorithm is run $\frac{n^2}{2}$ times on the same graph, the probability that C does not survive any of the runs is at most $(1 - \frac{2}{n^2})^{\frac{n^2}{2}} < \frac{1}{e}$.

In other words, the probability that C is obtained after $\frac{n^2}{2}$ runs is at least $(1 - \frac{1}{e})$.

Note

Karger's algorithm is both simpler and faster than any deterministic algorithm for determining the Min-Cut of an undirected graph.

Analysis (contd.)

Observation

If Karger's algorithm is run $\frac{n^2}{2}$ times on the same graph, the probability that C does not survive any of the runs is at most $(1-\frac{2}{n^2})^{\frac{n^2}{2}} < \frac{1}{e}$.
In other words, the probability that C is obtained after $\frac{n^2}{2}$ runs is at least $(1-\frac{1}{e})$.

Note

Karger's algorithm is both simpler and faster than any deterministic algorithm for determining the Min-Cut of an undirected graph.

Analysis (contd.)

Observation

If Karger's algorithm is run $\frac{n^2}{2}$ *times on the same graph, the probability that C does not survive any of the runs is at most* $(1 - \frac{2}{n^2})^{\frac{n^2}{2}} < \frac{1}{e}$.
In other words, the probability that C is obtained after $\frac{n^2}{2}$ *runs is at least* $(1 - \frac{1}{e})$.

Note

Karger's algorithm is both simpler and faster than any deterministic algorithm for determining the Min-Cut of an undirected graph.

Outline

1. Randomized Algorithms
   - Three paradigmatic problems
   - 2SAT
   - Min-Cut
   - Non-singularity of a Symbolic Matrix
2. Randomized Complexity Classes

Problem Description

Definition

Given an $n \times n$ matrix **A**, the determinant of **A** denoted by $|\mathbf{A}|$ is defined as: $\sum_{\pi} \sigma(\pi) \Pi_{i=1}^{n} A_{i,\pi(i)}$, where the summation is over all the permutations of n elements and $\sigma(\pi)$ is $+1$ if π is the product of an even number of transpositions and -1 otherwise. A matrix is said to be singular, if its determinant is identically 0 and non-singular otherwise.

Definition

A symbolic matrix is a matrix whose entries are polynomials, e.g.,

$$\begin{pmatrix} a & a^2 - 1 \\ d + b & e - a \end{pmatrix}$$

Goal

Given a symbolic square matrix, check whether it is identically zero, i.e., regardless of the values of the variables, the determinant always evaluates to zero.

Problem Description

Definition

Given an $n \times n$ matrix **A**, the determinant of **A** denoted by $|\mathbf{A}|$ is defined as: $\sum_{\pi} \sigma(\pi)\Pi_{i=1}^{n} A_{i,\pi(i)}$, where the summation is over all the permutations of n elements and $\sigma(\pi)$ is $+1$ if π is the product of an even number of transpositions and -1 otherwise. A matrix is said to be singular, if its determinant is identically 0 and non-singular otherwise.

Definition

A symbolic matrix is a matrix whose entries are polynomials, e.g.,

$$\begin{pmatrix} a & a^2 - 1 \\ d + b & e - a \end{pmatrix}$$

Goal

Given a symbolic square matrix, check whether it is identically zero, i.e., regardless of the values of the variables, the determinant always evaluates to zero.

Problem Description

Definition

Given an $n \times n$ matrix $\mathbf{A}$, the determinant of $\mathbf{A}$ denoted by $|\mathbf{A}|$ is defined as: $\sum_{\pi} \sigma(\pi) \Pi_{i=1}^{n} A_{i,\pi(i)}$, where the summation is over all the permutations of n elements and $\sigma(\pi)$ is $+1$ if π is the product of an even number of transpositions and -1 otherwise. A matrix is said to be singular, if its determinant is identically 0 and non-singular otherwise.

Definition

A symbolic matrix is a matrix whose entries are polynomials, e.g.,

$$\begin{pmatrix} a & a^2 - 1 \\ d + b & e - a \end{pmatrix}$$

Goal

Given a symbolic square matrix, check whether it is identically zero, i.e., regardless of the values of the variables, the determinant always evaluates to zero.

Issues involved in non-singularity checking

Issues

(i) Expansion is expensive!

(ii) Gaussian elimination is also expensive.

(iii) What is the complexity of this problem? Why?

Issues involved in non-singularity checking

Issues

(i) Expansion is expensive!

(ii) Gaussian elimination is also expensive.

(iii) What is the complexity of this problem? Why?

Issues involved in non-singularity checking

Issues

(i) Expansion is expensive!

(ii) Gaussian elimination is also expensive.

(iii) What is the complexity of this problem? Why?

Issues involved in non-singularity checking

Issues

(i) Expansion is expensive!
(ii) Gaussian elimination is also expensive.
(iii) What is the complexity of this problem? Why?

Mathematical Preliminaries

Theorem

Let $\phi(x_1, x_2, \ldots, x_m)$ be a polynomial, not identically zero, in m variables, each having degree at most d. Let $M > 0$ denote an integer. Then the number of m-tuples $\langle z_1, z_2, \ldots, z_m \rangle \in \{0, 1, \ldots M-1\}^m$ such that $\phi(z_1, z_2, \ldots z_m) = 0$ is at most $m \cdot d \cdot M^{m-1}$.

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra!
Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,
$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.
Let $\phi(z = \langle z_1, z_2, \ldots, z_m\rangle) = 0$.
Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M$ $= (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra! Assume true for $m - 1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,

$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.

Let $\phi(z = \langle z_1, z_2, \ldots, z_m \rangle) = 0$.

Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M = (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra! Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,

$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.

Let $\phi(z = \langle z_1, z_2, \ldots, z_m \rangle) = 0$.

Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M$ $= (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra!
Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,
$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.
Let $\phi(z = \langle z_1, z_2, \ldots, z_m \rangle) = 0$.
Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M = (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra! Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,
$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.
Let $\phi(z = \langle z_1, z_2, \ldots, z_m\rangle) = 0$.
Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M$ $= (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra! Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,
$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots (\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.
Let $\phi(z = \langle z_1, z_2, \ldots, z_m \rangle) = 0$.
Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M$ $= (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra! Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,

$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.

Let $\phi(z = \langle z_1, z_2, \ldots, z_m \rangle) = 0$.

Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M$ $= (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra! Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,
$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.
Let $\phi(z = \langle z_1, z_2, \ldots, z_m \rangle) = 0$.
Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M$ $= (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra! Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,
$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.
Let $\phi(z = \langle z_1, z_2, \ldots, z_m \rangle) = 0$.
Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M$ $= (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

Mathematical Preliminaries (contd.)

Proof.

If $m = 1$, the theorem is trivially true, as per the fundamental theorem of algebra! Assume true for $m-1$ variables. Rewrite ϕ so that it is a polynomial in x_m with coefficients in $\{x_1, x_2, \ldots, x_{m-1}\}$, i.e.,
$\phi = (\phi_1(x_1, x_2, \ldots, x_{m-1}))x_m^d + (\phi_2(x_1, x_2, \ldots, x_{m-1}))x_m^{d-1} + \ldots(\phi_{d-1}(x_1, x_2, \ldots, x_{m-1}))x_m^1 + (\phi_d(x_1, x_2, \ldots, x_{m-1}))$.
Let $\phi(z = \langle z_1, z_2, \ldots, z_m \rangle) = 0$.
Consider the following two cases:

(i) $\phi_1(z) = 0$. This means that z is a root of ϕ_1 and by induction, there are at most $(m-1) \cdot d \cdot M^{m-2}$ of these. For each of the M possible values of x_m, the first term will be zero. The total number of such possibilities is $(m-1) \cdot d \cdot M^{m-2} \cdot M$ $= (m-1) \cdot d \cdot M^{m-1}$.

(ii) $\phi_1(z) \neq 0$. This means that $\phi(z)$ defines a polynomial in x_m with degree at most d. Observe that for each combination of $x_1, x_2, \ldots, x_{m-1} \in \{0, 1, \ldots, M-1\}$, the resultant polynomial has at most d roots. Thus, the total number of zeros is at most $d \cdot M^{m-1}$.

Thus, the total number of zeros for ϕ is at most $m \cdot d \cdot M^{m-1}$. □

The Non-Singularity Checking Algorithm

Function NON-SING CHECK(**A**)

```
Generate m random integers between 0 and M = 2md.
Compute the resultant determinant of the numeric matrix A′ substituting these
integers into the symbolic matrix A.
if (|A′| ≠ 0) then
    A is not singular.
else
    A is probably singular.
end if
```

Algorithm 2.9: The Non-Singularity Checking Algorithm

The Non-Singularity Checking Algorithm

Function NON-SING CHECK(**A**)

Generate m random integers between 0 and $M = 2md$.
Compute the resultant determinant of the numeric matrix $\mathbf{A}'$ substituting these integers into the symbolic matrix **A**.
if ($|\mathbf{A}'| \neq 0$) **then**
    A is not singular.
else
    A is probably singular.
end if

Algorithm 2.10: The Non-Singularity Checking Algorithm

The Non-Singularity Checking Algorithm

Function NON-SING CHECK(**A**)

```
Generate m random integers between 0 and M = 2md.
Compute the resultant determinant of the numeric matrix A' substituting these
integers into the symbolic matrix A.
if (|A'| ≠ 0) then
    A is not singular.
else
    A is probably singular.
end if
```

Algorithm 2.11: The Non-Singularity Checking Algorithm

Analysis

Error bound

The probability that Algorithm 2.9 declares that a non-singular matrix is singular is precisely $\frac{m \cdot d \cdot (2 \cdot m \cdot d)^{m-1}}{(2md)^m} = \frac{1}{2}$.

Complexity of non-singularity checking in symbolic matrices

Not only is this problem not known to be in **P**, it is rather unlikely that it will be.

Analysis

Error bound

The probability that Algorithm 2.9 declares that a non-singular matrix is singular is precisely $\frac{m \cdot d \cdot (2 \cdot m \cdot d)^{m-1}}{(2md)^m} = \frac{1}{2}$.

Complexity of non-singularity checking in symbolic matrices

Not only is this problem not known to be in **P**, it is rather unlikely that it will be.

Analysis

Error bound

The probability that Algorithm 2.9 declares that a non-singular matrix is singular is precisely $\frac{m \cdot d \cdot (2 \cdot m \cdot d)^{m-1}}{(2md)^m} = \frac{1}{2}$.

Complexity of non-singularity checking in symbolic matrices

Not only is this problem not known to be in **P**, it is rather unlikely that it will be.

Randomized Complexity Classes

Note

The following definitions are from [1].

Definition

The class **RP** consists of all languages $L \subseteq \Sigma^*$ that have a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"yes"}] \geq \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"yes"}] = 0$.

Observations

(i) *Rejection is unanimous, acceptance is by majority.*
(ii) *Only positive-sided error is allowed.*
(iii) *The number $\frac{1}{2}$ can be any fixed constant between 0 and 1, without affecting the set of languages in* **RP**.
(iv) *The three paradigmatic problems are in* **RP**.

Randomized Complexity Classes

Note

The following definitions are from [1].

Definition

The class **RP** consists of all languages $L \subseteq \Sigma^*$ that have a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"}yes\text{"}] \geq \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"}yes\text{"}] = 0$.

Observations

(i) *Rejection is unanimous, acceptance is by majority.*

(ii) *Only positive-sided error is allowed.*

(iii) *The number $\frac{1}{2}$ can be any fixed constant between 0 and 1, without affecting the set of languages in* **RP**.

(iv) *The three paradigmatic problems are in* **RP**.

Randomized Complexity Classes

Note

The following definitions are from [1].

Definition

The class **RP** consists of all languages $L \subseteq \Sigma^*$ that have a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] \geq \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] = 0$.

Observations

(i) *Rejection is unanimous, acceptance is by majority.*
(ii) *Only positive-sided error is allowed.*
(iii) *The number $\frac{1}{2}$ can be any fixed constant between 0 and 1, without affecting the set of languages in* **RP**.
(iv) *The three paradigmatic problems are in* **RP**.

Randomized Complexity Classes

Note

The following definitions are from [1].

Definition

The class **RP** consists of all languages $L \subseteq \Sigma^*$ that have a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{``}yes\text{''}] \geq \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{``}yes\text{''}] = 0$.

Observations

(i) *Rejection is unanimous, acceptance is by majority.*
(ii) *Only positive-sided error is allowed.*
(iii) *The number $\frac{1}{2}$ can be any fixed constant between* 0 *and* 1*, without affecting the set of languages in* **RP**.
(iv) *The three paradigmatic problems are in* **RP**.

Randomized Complexity Classes

Note

The following definitions are from [1].

Definition

The class **RP** consists of all languages $L \subseteq \Sigma^*$ that have a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] \geq \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] = 0$.

Observations

(i) *Rejection is unanimous, acceptance is by majority.*

(ii) *Only positive-sided error is allowed.*

(iii) *The number $\frac{1}{2}$ can be any fixed constant between 0 and 1, without affecting the set of languages in* **RP**.

(iv) *The three paradigmatic problems are in* **RP**.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **coRP**, if its complement is in **RP**.

Definition

A language $L \subseteq \Sigma^*$ is in **ZPP** is it is in **RP** $\cap$ **coRP**.

Definition

A language $L \subseteq \Sigma^*$ is in **PP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] > \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] < \frac{1}{2}$.

Note

The problem MAJSAT *is defined as follows: Given a formula in CNF, is it the case that the majority of the* 2^n *assignments satisfy it?* MAJSAT *is the quintessential* **PP** *problem; in fact, it is* **PP-complete**.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **coRP**, if its complement is in **RP**.

Definition

A language $L \subseteq \Sigma^*$ is in **ZPP** is it is in **RP** $\cap$ **coRP**.

Definition

A language $L \subseteq \Sigma^*$ is in **PP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"yes"}] > \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"yes"}] < \frac{1}{2}$.

Note

The problem MAJSAT *is defined as follows: Given a formula in CNF, is it the case that the majority of the* 2^n *assignments satisfy it?* MAJSAT *is the quintessential* **PP** *problem; in fact, it is* **PP-complete**.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **coRP**, if its complement is in **RP**.

Definition

A language $L \subseteq \Sigma^*$ is in **ZPP** is it is in **RP** $\cap$ **coRP**.

Definition

A language $L \subseteq \Sigma^*$ is in **PP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] > \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] < \frac{1}{2}$.

Note

The problem MAJSAT *is defined as follows: Given a formula in CNF, is it the case that the majority of the 2^n assignments satisfy it?* MAJSAT *is the quintessential* **PP** *problem; in fact, it is* **PP-complete**.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **coRP**, if its complement is in **RP**.

Definition

A language $L \subseteq \Sigma^*$ is in **ZPP** is it is in **RP** $\cap$ **coRP**.

Definition

A language $L \subseteq \Sigma^*$ is in **PP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{``}yes\text{''}] > \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{``}yes\text{''}] < \frac{1}{2}$.

Note

The problem MAJSAT *is defined as follows: Given a formula in CNF, is it the case that the majority of the* 2^n *assignments satisfy it?* MAJSAT *is the quintessential* **PP** *problem; in fact, it is* **PP-complete**.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **coRP**, if its complement is in **RP**.

Definition

A language $L \subseteq \Sigma^*$ is in **ZPP** is it is in **RP** $\cap$ **coRP**.

Definition

A language $L \subseteq \Sigma^*$ is in **PP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes''] > \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes''] < \frac{1}{2}$.

Note

The problem MAJSAT *is defined as follows: Given a formula in CNF, is it the case that the majority of the 2^n assignments satisfy it?* MAJSAT *is the quintessential* **PP** *problem; in fact, it is* **PP-complete**.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **coRP**, if its complement is in **RP**.

Definition

A language $L \subseteq \Sigma^*$ is in **ZPP** is it is in **RP** $\cap$ **coRP**.

Definition

A language $L \subseteq \Sigma^*$ is in **PP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"}yes\text{''}] > \frac{1}{2}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"}yes\text{''}] < \frac{1}{2}$.

Note

The problem `MAJSAT` *is defined as follows: Given a formula in CNF, is it the case that the majority of the* 2^n *assignments satisfy it?* `MAJSAT` *is the quintessential* **PP** *problem; in fact, it is* **PP-complete**.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **BPP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"yes"}] \geq \frac{3}{4}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{"yes"}] \leq \frac{1}{4}$.

Alternative view of **RP**

RP denotes the set of languages L which can be decided by a polynomially bounded non-deterministic Turing machine N in the following manner: For each input x, if $x \in L$, then at least half the computations of N on x end in accepting leaves and if $x \notin L$, the all computations of N on x end in rejecting leaves. WIthout loss of generality, we may assume that the degree of non-determinism is exactly 2 at each node of the computation tree.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **BPP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] \geq \frac{3}{4}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] \leq \frac{1}{4}$.

Alternative view of **RP**

RP denotes the set of languages L which can be decided by a polynomially bounded non-deterministic Turing machine N in the following manner: For each input x, if $x \in L$, then at least half the computations of N on x end in accepting leaves and if $x \notin L$, the all computations of N on x end in rejecting leaves. WIthout loss of generality, we may assume that the degree of non-determinism is exactly 2 at each node of the computation tree.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **BPP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] \geq \frac{3}{4}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes\text{''}] \leq \frac{1}{4}$.

Alternative view of **RP**

RP denotes the set of languages L which can be decided by a polynomially bounded non-deterministic Turing machine N in the following manner: For each input x, if $x \in L$, then at least half the computations of N on x end in accepting leaves and if $x \notin L$, the all computations of N on x end in rejecting leaves. WIthout loss of generality, we may assume that the degree of non-determinism is exactly 2 at each node of the computation tree.

Randomized Complexity Classes (contd.)

Definition

A language $L \subseteq \Sigma^*$ is in **BPP**, if there exists a randomized algorithm $\mathcal{A}$ running in worst-case polynomial time, such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes''] \geq \frac{3}{4}$.
- $x \notin L \Rightarrow \mathbf{Pr}[\mathcal{A}(x)] = \text{“}yes''] \leq \frac{1}{4}$.

Alternative view of **RP**

RP denotes the set of languages L which can be decided by a polynomially bounded non-deterministic Turing machine N in the following manner: For each input x, if $x \in L$, then at least half the computations of N on x end in accepting leaves and if $x \notin L$, the all computations of N on x end in rejecting leaves. WIthout loss of generality, we may assume that the degree of non-determinism is exactly 2 at each node of the computation tree.

Relations between complexity classes

Observations

(i) **P** $\subseteq$ **RP** $\subseteq$ **NP**.
(ii) **P** $\subseteq$ **coRP** $\subseteq$ **coNP**.
(iii) **RP** $\subseteq$ **BPP** $\subseteq$ **PP**.

Theorem

NP $\subseteq$ **PP**.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

Relations between complexity classes

Observations

(i) **P** ⊆ **RP** ⊆ **NP**.
(ii) **P** ⊆ **coRP** ⊆ **coNP**.
(iii) **RP** ⊆ **BPP** ⊆ **PP**.

Theorem

NP ⊆ **PP**.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

Relations between complexity classes

Observations

(i) $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$.

(ii) $\mathbf{P} \subseteq \mathbf{coRP} \subseteq \mathbf{coNP}$.

(iii) $\mathbf{RP} \subseteq \mathbf{BPP} \subseteq \mathbf{PP}$.

Theorem

$\mathbf{NP} \subseteq \mathbf{PP}$.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

Relations between complexity classes

Observations

(i) **P** $\subseteq$ **RP** $\subseteq$ **NP**.

(ii) **P** $\subseteq$ **coRP** $\subseteq$ **coNP**.

(iii) **RP** $\subseteq$ **BPP** $\subseteq$ **PP**.

Theorem

NP $\subseteq$ **PP**.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

Relations between complexity classes

Observations

(i) **P** $\subseteq$ **RP** $\subseteq$ **NP**.
(ii) **P** $\subseteq$ **coRP** $\subseteq$ **coNP**.
(iii) **RP** $\subseteq$ **BPP** $\subseteq$ **PP**.

Theorem

NP $\subseteq$ **PP**.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

Relations between complexity classes

Observations

(i) **P** $\subseteq$ **RP** $\subseteq$ **NP**.

(ii) **P** $\subseteq$ **coRP** $\subseteq$ **coNP**.

(iii) **RP** $\subseteq$ **BPP** $\subseteq$ **PP**.

Theorem

NP $\subseteq$ **PP**.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

Relations between complexity classes

Observations

(i) **P** $\subseteq$ **RP** $\subseteq$ **NP**.
(ii) **P** $\subseteq$ **coRP** $\subseteq$ **coNP**.
(iii) **RP** $\subseteq$ **BPP** $\subseteq$ **PP**.

Theorem

NP $\subseteq$ **PP**.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

Relations between complexity classes

Observations

(i) **P** $\subseteq$ **RP** $\subseteq$ **NP**.

(ii) **P** $\subseteq$ **coRP** $\subseteq$ **coNP**.

(iii) **RP** $\subseteq$ **BPP** $\subseteq$ **PP**.

Theorem

NP $\subseteq$ **PP**.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

Relations between complexity classes

Observations

(i) $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$.

(ii) $\mathbf{P} \subseteq \mathbf{coRP} \subseteq \mathbf{coNP}$.

(iii) $\mathbf{RP} \subseteq \mathbf{BPP} \subseteq \mathbf{PP}$.

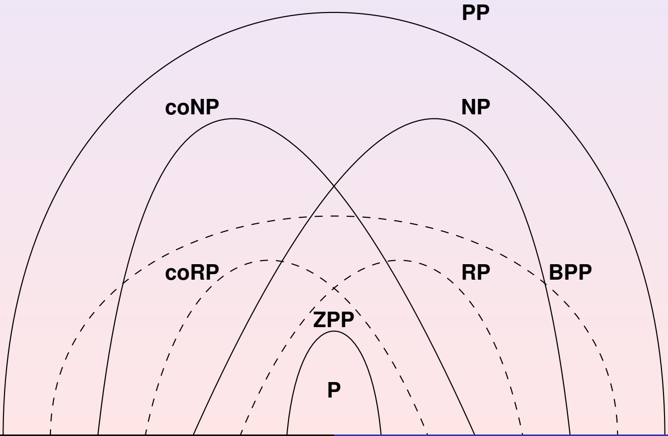
Theorem

$\mathbf{NP} \subseteq \mathbf{PP}$.

Proof.

Let L be accepted by an NDTM N in polynomial time $p()$.
Build an NDTM N' which contains a new initial state, with branching factor 2. One branch moves to N and the other branch which has exactly the same number of computations as N leads only to leaves which are all "accepting". If $x \in L$, N' accepts with clear majority! If $x \notin L$, then $N(x')$ does not have a clear majority of accepting computations and hence N' rejects. □

The Complexity Picture



Rajeev Motwani and Prabhakar Raghavan.
Randomized Algorithms.
Cambridge University Press, Cambridge, England, June 1995.