Formal Languages and Automata, 2024 Fall semester

Lec 22. What next?

Eunjung Kim

Unlikely to be P-time, what next?

Popular algorithmic approaches

- Exponential running time, but as fast as possible.
  $\rightsquigarrow$ Exponential-time algorithm.
- Not exact/optimal solution, but as close to optimal as possible.
  $\rightsquigarrow$ Approximation algorithm. runs in P-time.
- Exponential running time, but exponential w.r.t a small parameter k.
  $\rightsquigarrow$ Fixed-parameter algorithm. runs in time $f(k) \cdot n^{O(1)}$.
- Produced intended solution with high probability.
  $\rightsquigarrow$ Randomized algorithm. can be optimal or approximate.

Algorithms beyond the classic set-up

Popular algorithmic approaches

- online algorithm: competitive ratio, memory, update time
- distributed algorithm: communication, data size on each processor

Directed Hamiltonian Path

A Hamiltonian path of a directed graph $G = (V, E)$ is a directed path which visits every vertex of G precisely once.

PROBLEM	DIRECTED HAMILTONIAN PATH
INPUT	a graph G.
QUESTION	does G have a Hamiltonian path?

Dynamic programming for HAM

Naive algorithm: guess all possible tours, $n!$ many of them.

Dynamic programming algorithm in $2^n \cdot n^{O(1)}$ time.

For every vertex subset X and $v \in X$,
$P[X, v] = 1$ if and only if there exists a Hamiltonian path of $G[X]$ ending in v.

1. Initialize: $P[\{w\}, w] = 1$ for every $w \in V$.
2. Inductive step: assume $P[X, v]$ is known for all $|X|$ of size i and $v \in X$. Let's compute $P[X, v]$ for each $|X|$ of size $i + 1$ and $v \in X$.

$$P[X, v] = \bigvee_{w \in X \setminus v} P[X \setminus v, w] \cdot [(w, v) \in E(G)].$$

3. There exists a Hamiltonian path of G if and only if there is a vertex v s.t. $P[V, v] = 1$.

Vertex Cover

A vertex cover of a graph is a vertex subset X of G such that X takes at least one endpoint of every edge in G.

Vertex Cover

Input	a graph G and an integer k.
Question	does G have a vertex cover of size at most k?

Vertex Cover is NP-complete; reduction from Independent Set.

Approximation for Vertex Cover

Greedy algorithm for Vertex Cover

1. Greedily find a maximal matching M.
2. Output $V(M)$.

- The output of the algorithm is indeed a vertex cover. (Why?)
- Analysis:

$$|M| \leq \text{opt vc} \leq \text{output vc} = 2 \cdot |M|.$$

  Therefore, output vc $\leq 2 \cdot$ opt vc.
- 2-approximation: algorithm outputs a solution no larger than twice the optimal.

Parameterized Vertex Cover

Parameterized Vertex Cover

Input a graph G and an integer k.

Parameter k.

Question does G have a vertex cover of size at most k?

FPT-algorithm for Vertex Cover

Algorithm **VC**(G, k)

1. If G has no edge **return** Yes.
2. Else if $k = 0$ **return** No.
3. Else (comment: G has an edge and $k > 0$)
   1. Pick an edge uv.
   2. Return **VC**$(G - u, k - 1)$ or **VC**$(G - v, k - 1)$.

FPT-algorithm for Vertex Cover

Runtime analysis:

- Recursive calls of **VC**(G, k) can be expressed as a branching tree $\mathcal{T}$.
- Parameter k strictly decreases by depth $\rightsquigarrow$ depth at most k.
- $\mathcal{T}$ has at most 2^k leaves $\rightsquigarrow$ runs in time $2^k \cdot \mathit{poly}(n)$.

Randomized approximation for Max Cut

Max Cut

Input	a graph $G = (V, E)$.
Question	find a bipartition of V into V_1, V_2 maximizing the crossing edges between V_1 and V_2.

Max Cut is NP-complete; reduction from 3SAT via a series of reductions.

Randomized approximation for Max Cut

Random greedy algorithm for Max Cut

1. For each vertex v, decide where to put v with prob 0.5, independently at random.
2. Count the crossing edges between V_1 and V_2.

- For arbitrary edge e, probability that e counts as 'crossing' is 1/2.
- $E(\#\text{crossing edges}) = \sum_{e \in E} P(e \text{ is crossing}) = |E|/2$.
- In expectation, the output is at least half the optimal solution.
- Repeat the greedy algorithm several times $\rightsquigarrow$ turn the 'expectation' into probability guarantee.
- Derandomization techniques.

Online algorithm for Secretary Problem

Secretary Problem

1. Input: n candidates for a secretary job comes for an interview, in a random order.
2. Goal: find the best candidate
3. Set-up: each secretary i has competence $w(i)$. Once the interview is done, you have to decide immediately to hire the candidate or not. A candidate you once rejected can never be recalled. You know the number of candidates.

Competitive ratio:
w(the chosen candidate)$/w$(the best one among n candidates)

Online algorithm for Secretary Problem

Algorithm

1. Reject the first n/e candidates.
2. Afterwards, hire the first candidate who is as good as the best one among the first n/e.

Two-party communication: equality

Equality

- Alice has $x \in \{0,1\}^n$, Bob has $y \in \{0,1\}^n$.
- They want to decide if $x = y$ with minimum number of bit exchanges.

Two-party communication: Equality

Deterministic communication

1. Alice sends to Bob x; n-bits.
2. Bob compares x and y.

Randomized communication: public randomness

1. Uniform random $z \in \{0, 1\}^n$; both Alice and Bob can access it.
2. Alice computes $x \cdot z$ and sends the outcome to Bob; 1-bits.
3. Bob computes $y \cdot z$ and reject if $y \cdot z \neq x \cdot z$.

Two-party communication: Equality

Deterministic communication

1. Alice sends to Bob x; n-bits.
2. Bob compares x and y.

Randomized communication: public randomness

1. Uniform random $z \in \{0, 1\}^n$; both Alice and Bob can access it.
2. Alice computes $x \cdot z$ and sends the outcome to Bob; 1-bits.
3. Bob computes $y \cdot z$ and reject if $y \cdot z \neq x \cdot z$.

Distributed computation: Leader election

Leader Election in Distributed Computing

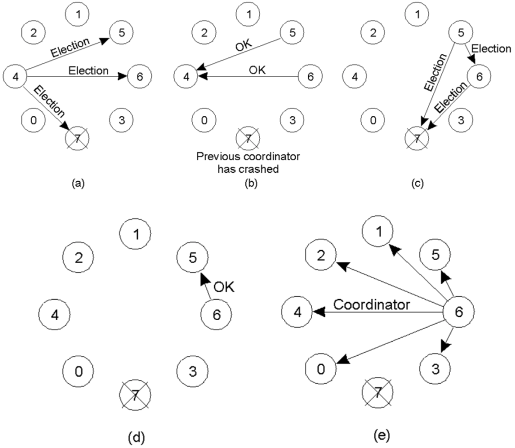
- n nodes (processors) with distinct ID, every processor knows the IP address of every other processor and their IDs.
- Some processors may crash at times.
- Want to make sure that the highest ID node currently alive has the token.
- Protocol so that all processors know the leader (with little communication overhead).

Distributed computation: Leader election

Bully Algorithm

1. Three types of messages: (i) Election started, (ii) I'm alive, (iii) I'm the leader.
2. If a processor with the highest ID wakes up (after crash), it sends "I'm the leader" message.
3. If a node detects a failure of the leader, it sends "Election" to all higher ID nodes.
4. If a node receives "Election" message, it sends "I'm alive" to the sender and sends "Election" message to all higher ID nodes.
5. If a node does not receive "I'm alive" message after sending "Election", it sends "I'm the leader" message to everyone.
6. If a node receives "I'm the leader" message, it considers the sender the leader.

Distributed computation: Leader election



Lecture Note 14, CMPSCI 677 Operating Systems, Prashant Shenoy.

Distributed computation: compact local certificate

ERTY

- You're a processor (node) in a network (graph) with ID of length log n. You are only aware of your neighbors and nothing beyond it.
- An external coordinator can give each node v some data $\ell(v)$.
- Each node can compute whatever they likes (unlimited computing power).
- One round of communication: each node communicates with its neighbors some message.
- After-communication computation: each node, based on what it heard from its neighbors, makes a decision and say "yes" or "no".

Distributed computation: compact local certificate

Can the data $\ell(v)$ be designed (as small size as possible) so that

- All nodes say "yes" if property P holds.
- Some node says "no" if property P does not hold.

$O(1)$-bit local certificate for bipartiteness

Can the nodes collectively decide if the underlying network is bipartite?

$O(\log n)$-bit local certificate for tree

Can the nodes collectively decide if the underlying network is a forest?