Lecture 5: "The Principle of Deferred Decisions. Chernoff Bounds"

Sotiris Nikoletseas
Associate Professor

CEID - ETY Course
2013 - 2014

Overview

A1. The Principle of Deferred Decisions

A2. The Proposal Algorithm for the Stable Marriage Problem

B1. Chernoff Bounds

B2. A Randomized Algorithm for Dominating Sets

A1. The Principle of Deferred Decisions

The Clock Solitaire game:

- randomly shuffle a standard pack of 52 cards
- split the cards into 13 piles of 4 cards each; label piles as A, 2, . . . , 10, J, Q, K
- take first card from "K" pile
- take next card from pile "X", where X is the value of the previous card taken
- repeat until:
  - either all cards removed ("win")
  - or you get stuck ("lose")

We want to evaluate the probability of "win".

Key features - game termination

Remark 1. The last card we take before the game ends (either winning or loosing) is a "K".

Proof:

- Let us assume that at iteration j we draw card X but pile X is empty (thus the game terminates).
- Let $X \neq K$ (i.e. we lose). Because pile X is empty and $X \neq K$, we must have already drawn (prior to draw j) 4 X cards. But then we can not draw an X card at the jth iteration, a contradiction. □

Note: There is no contradiction if the last card is a "K" and all other cards have been already removed (in that case the game terminates with win).

Key features - win

Remark 2. We win if the fourth "K" card is drawn at the 52 iteration.

Proof:

- whenever we draw for the 1st, 2nd or 3rd time a"K" card, the game does not terminate because the K pile is not empty so we can continue (see remark 1).
- when the fourth K is drawn at the 52nd iteration then all cards are removed and the game's result is "win".

□

The probability of win

- Because of remark 2, it is:
  $\Pr\{\text{win}\} = \Pr\{\text{4th "K" at the 52nd iteration}\} =$
  $= \frac{\#\text{game evolutions: 52nd card = 4th "K"}}{\#\text{all game evolutions}}$
- Note: Considering all possible game evolutions is a rather naive approach since we have to count all ways to partition the 52 cards into 13 distinct piles, with an ordering on the 4 cards in each pile. This complicates the probability evaluation because of the dependence introduced by each random draw of a card.

$\Rightarrow$ we define another probability space that better captures the random dynamics of the game evolution.

The principle of deferred decisions

- Basic idea: rather than fix (end enumerate) the entire set of potential random choices in advance, instead let the random choices unfold with the progress of the random experiment.
- In this particular game at each draw any card not drawn yet is equally likely to be drawn.
- A winning game corresponds to a dynamics where the first 51 random draws include 3 "K" cards exactly.
- This is equivalent to draw the 4th "K" at the 52nd iteration.
- So we "forget" how the first 51 draws came out and focus on the 52nd draw, which must be a "K".
- But the latter probability is $\frac{1}{13}$ because of symmetry (e.g. the type of the 52nd card is random uniform among all 13 types).

The probability of win

- Thus we have proved the following:
  Theorem: The probability of win at the clock solitaire is $\frac{1}{13}$.
- An alternative approach:
  - we actually have 13x4=52 distinct positions (13 piles, 4 positions each) where 52 distinct cards are placed. This gives a total of 52! different placements.
  - each game evolution actually corresponds to an ordered permutation of the 52 cards.
  - The winning permutations are those where the 52nd card is a "K" (4 ways) and the 51 preceding cards are arbitrarily chosen (51!). Thus:
    $$\Pr\{\text{win}\} = \frac{4 \cdot 51!}{52!} = \frac{4}{52} = \frac{1}{13}$$
    (the idea was to defer, i.e. first consider the last choice and then conditionally the previous ones!) In other words, the principle does not assume that the entire set of random choices is made in advance. Rather, at each step of the process we fix only the random choices that must be revealed.

A2. The Proposal Algorithm for the Stable Matching Problem

The Stable Matching Problem. Consider n women $(w_1, \ldots, w_n)$ and n men $(m_1, \ldots, m_n)$.

- A matching is a 1-1 correspondence between the men and the women (i.e. we assume monogamous, heterosexual matchings)
- Each person has a strict preference list of the members of the other sex.
- A matching is unstable iff there exist w_i and m_j such that:
  - w_i and m_j are not matched together
  - w_i prefers m_j to her match
  - m_j prefers w_i to his match
- a matching which is not unstable is stable

Many applications (e.g. assigning teachers to schools they want to serve at, doctors to hospitals, etc.)

Questions

- does a stable matching always exists? (i.e. for all choices of preference lists?)
- can we find one efficiently?

Answers:

- yes, there is at least one stable matching for every choice of preference lists
- we will prove this by providing an algorithm that finds a stable matching
- this algorithm is randomized (Las Vegas) and needs $O(n \ln n)$ time w.h.p.

The Gale-Shapley "Proposal" Algorithm (I)

- Basic idea: "man proposes, woman disposes". Each currently unattached man proposes to the woman he most desires and has not rejected him already. The woman accepts him if she is currently unattached or she prefers him to her current match.

The Gale-Shapley "Proposal" Algorithm (II)

- Features:
  - Once a woman gets matched, she remains matched forever (though her mates may change)
  - The desirability of her mates (from her perspective) can only increase with time, i.e. at each step either a woman matches for the first time, or she matches to a more desired (to her) mate
  - Unmatched men always have (at least one) an unmatched woman to make proposals to.
  - Unmatched men can propose to currently matched women
  - Men can change status from unmatched to matched and then to unmatched (rejected) and so on, based on the proposals of other men and the womens' choice.

A more formal description

- let us assume some arbitrary ordering of the men
- let i the smallest value such that man m_i is unmatched
- m_i proposes to the most desirable woman (according to his own preference list) that has not already rejected him.
- she accepts him if either a) she is currently unmatched or b) she prefers him to her current match (in that case, her current match becomes unmatched).
- this is repeated until there are no unmatched men left.

Questions:

- does the algorithm terminate?
- is the resulting matching stable?
- how much time it takes?

Is the algorithm well-defined?

Lemma. Whenever there is an unmatched man m_i , there is some woman he has not proposed to (so she cannot have rejected him in the past).

Proof:

- Once a woman becomes matched, she never becomes unmatched in the future
- Since m_i is unmatched currently, all women he has proposed to (if any) so far are matched.
- Thus, if m_i has proposed to all women, then all women are matched, hence all men are matched too - a contradiction.

□

Worst case time complexity

Theorem. The algorithm terminates after $O(n^2)$ iterations (proposals).

Proof:

- For man m_i, let t_i the number of women m_i could still potentially propose.
- At each step (proposal), the sum $\sum_{i=1}^{n} t_i$ decreases by 1

  (three cases actually: a) get accepted by a matched woman so her current mate gets rejected and cannot propose her again b) get accepted by an unmatched woman so he cannot propose her again c) get rejected)
- Initially $\sum_{i=1}^{n} t_i = n^2$, so the number of proposals is at most n^2.

□

Correctness

Theorem: The matching found by the algorithm is stable.

Proof:

- Let us assume the matching is unstable, so there is at least two pairs $m_i - w_j$ and $m_k - w_l$, however with m_i and w_l preferring to be matched together.
- Since m_i prefers w_l to w_j, he must have proposed to w_l before he proposed to w_j.
- But she rejected him, so she must prefer her current match m_k to m_i: a) either she already had a better match at the time m_i proposed to her or b) she matched m_i initially and then got a more desirable proposal. A contradiction. $\square$

Average case analysis of the Proposal Algorithm

- Note: In randomized algorithms random choices are made when processing a"fixed" input.
- In average case analysis, the input is random and we analyze the time complexity (a random variable) of a deterministic algorithm
- In the matching problem, the input's randomness is introduced by assuming that the preference lists are random uniform.

The "Amnesiac" version of the Proposal algorithm

- A simplified modification of the Gale-Shapley algorithm:
  - At each step, m_i proposes to a woman chosen uniformly at a random among all n women (including those he has been rejected by)

Note:

- This does not affect the output of the algorithm, since if m_i was rejected by a woman, he will be rejected again if he proposes her again
- The "Amnesiac" algorithm thus performs more proposals since it includes some "wasted" rejects, so his expected running time is an upper bound on the time of the original algorithm

The expected time complexity

Theorem If the preference lists are chosen in a uniform random manner, the expected number of proposals in the Gale-Shapley algorithm is at most $O(n \log n)$.

Proof:

- Clearly the algorithm terminates once all women have received at least one proposal.
- So the matching random process is actually a coupon collectors problem, for which we have proved the following bound:

  Coupon Collectors: If $m = n \ln n + cn$ (for any constant $c \in R$) then the time T for collecting all n coupons obeys the following:

  $$\Pr\{T > m\} = 1 - e^{-e^{-c}}$$

  □

The use of the principle of deferred decisions

- We have actually used the principle in the sense that we do not assume that the (random) preference lists are chosen in advance. In fact we somehow (in the average case analysis of the Gale-Shapley algorithm) assume that men do not know their preference lists and each time a man makes a proposal he picks a random woman.
- The only dependency (from the past proposals) left is eliminated in the Amnesiac algorithm by the wasted proposals to women having already rejected a man (i.e. we "forget" the past)

B1. The Chernoffs Bounds

- Tail Bounds on the probability of deviation from the expected value of a random variable.
- Focus on
  sums of independent, indicator (Bernoulli) random variables
- Such sums abstract quantitative results of random choices in randomized algorithms (e.g. the number of comparisons in the randomized quicksort algorithm, where for each two elements i, j the r.v. X_{ij} is 1 if $S_{(i)}, S_{(j)}$ are compared (and 0 otherwise).
- The Chernoff bounds are exponential i.e. much sharper than the Markov, Chebyshev bounds.

Poisson and Bernoulli trials

- Poisson trials:
  - repeated, independent trials
  - two possible outcomes in each trial ("success", "failure")
  - potentially different success probability p_i in each trial i.
  - we take the sum of the corresponding indicator variables X_i for each trial i (it measures the total number of successes).

  i.e. for $1 \leq i \leq n$:

$$X_i = \begin{cases} 1 & \text{(success), with probability } p_i \\ 0 & \text{(failure), with probability } q_i = 1 - p_i \end{cases}$$

$$X = X_1 + \cdots + X_n = \sum_{i=1}^{n} X_i$$

- Bernoulli trials: Poisson trials when $p_i = p, \forall i$.
  (in that case $X \sim B(n, p)$ i.e. it follows the binomial distribution)

The tail probability

- Clearly, the expected value (or mean) of X is:

$$\mu = E(X) = \sum_{i=1}^{n} p_i$$

- Two important questions:

  (1) For a real number $\beta > 0$ what is the probability that X exceeds $(1+\beta)\mu$?
  (i.e. we seek a bound on the tail probability)

  $\rightarrow$ this is useful in the analysis of randomized algorithms (i.e., to show that the probability of failure to achieve a certain expected performance is small).

  (2) How large must β be so that the tail probability is less than a certain value ϵ ? (this is relevant in the design of the algorithm).

The Chernoff bound (for exceeding the mean) (I)

Theorem 1. Let X_i be n independent Poisson(p_i) trials,

$$X = \sum_{i=1}^{n} X_i, \quad \mu = E(X) = \sum_{i=1}^{n} p_i.$$

Then, for any $\beta > 0$, it is :

$$\Pr\{X > (1+\beta)\mu\} < \left[\frac{e^{\beta}}{(1+\beta)^{(1+\beta)}}\right]^{\mu}$$

Proof: For any positive t:

$$\Pr\{X > (1+\beta)\mu\} = \Pr\{e^{tX} > e^{t(1+\beta)\mu}\}$$

By the Markov inequality: $\Pr\{e^{tX} \geq e^{t(1+\beta)\mu}\} < \frac{E[e^{tX}]}{e^{t(1+\beta)\mu}}$ (1)

But $E[e^{tX}] = E\left[e^{t(X_1+\cdots+X_n)}\right] = E\left[\prod_{i=1}^{n} e^{tX_i}\right] = \prod_{i=1}^{n} E[e^{tX_i}]$ (2)

Now $E[e^{tX_i}] = e^t p_i + 1 \cdot (1-p_i) = 1 + p_i(e^t - 1) \leq e^{p_i(e^t-1)}$ (3)

The Chernoff bound (for exceeding the mean) (II)

$$(2) \wedge (3) \Rightarrow E[e^{tX}] = \prod_{i=1}^{n} e^{p_i(e^t-1)} = e^{\sum_{i=1}^{n} p_i(e^t-1)} = e^{(e^t-1)\mu}$$

$$(1) \Rightarrow \Pr\{e^{tX} \geq e^{t(1+\beta)\mu}\} < \frac{e^{(e^t-1)\mu}}{e^{t(1+\beta)\mu}}$$

The right part is minimized when $t = \ln(1+\beta)$
(note that $t > 0 \Leftrightarrow \beta > 0$).

$$t = \ln(1+\beta) \Rightarrow \Pr\{X > (1+\beta)\mu\} < \frac{e^{(e^{\ln(1+\beta)}-1)\mu}}{e^{\ln(1+\beta)(1+\beta)\mu}} =$$
$$= \frac{e^{\beta\mu}}{(1+\beta)^{(1+\beta)\mu}} = \left[\frac{e^{\beta}}{(1+\beta)^{(1+\beta)}}\right]^{\mu}$$

The various Chernoff bounds (I)

Theorem 2. (Chernoff bound for exceeding the mean)

For X_i, X, μ as in Theorem 1 it is:

$$\forall \beta \in [0,1] : \Pr\{X \geq (1+\beta)\mu\} \leq e^{-\frac{\beta^2 \mu}{3}}$$

Proof: $\forall \beta \in [0,1] : \left(\frac{e^{\beta}}{(1+\beta)^{(1+\beta)}}\right)^{\mu} \leq e^{-\frac{\beta^2 \mu}{3}}$

Theorem 3. (Chernoff bound for the left tail)

For X_i, X, μ as in Theorem 1 it is:

$$\forall \beta \in [0,1] : \Pr\{X \leq (1-\beta)\mu\} \leq e^{-\frac{\beta^2 \mu}{2}}$$

Proof: See the MR book.

The various Chernoff bounds (II)

Theorem 4. (The combined Chernoff bound)

For X_i, X, μ as in Theorem 1 it is:

$$\forall \beta \in [0,1] : \Pr\{X \in (1 \pm \beta)\mu\} \geq 1 - 2e^{-\frac{\beta^2 \mu}{3}}$$

Proof: Follows easily from Theorems 2, 3.

- Important remark:
  - $\mu \to \infty$ (at an arbitrary slow rate) $\Rightarrow$ $\Pr\{X \in (1 \pm \beta)\mu\} \to 1$
  - $\mu = \Omega(\log n) \Rightarrow \Pr\{X \in (1 \pm \beta)\mu\} \geq 1 - 2n^{-\gamma}$, where $\gamma > 1$ i.e. a logarithmic mean guarantees a polynomially fast convergence to 1 of the concentration probability.

B2. A Randomized Algorithm for Dominating Sets

- The dominating set problem: Find a subset of the vertices of a graph such that every vertex not in this set is adjacent to at least one vertex in it. Formally:
  $V' \leq V(G)$ dominating set in $G(V, E)$ iff
  $\forall u \notin V', \exists v \in V' : (u, v) \in E(G)$
- The problem is important and well motivated from real networks, since the dominating set plays a "central" role in the graph.
- Obviously we want to find a dominating set which is as small as possible.

The complexity of the problem / using randomness

- Finding a minimum dominating set is an NP-hard problem.

- Randomness can be used to "attack" the problem:
  - It has been shown that in $G_{n,p}(p = \frac{1}{2})$ random graphs, $\nexists$ dominating set of size $< \log n$ w.h.p.
    (technique: Linearity of expectation + Markov inequality)
  - Also, w.h.p. $\exists$ dominating set of size $\lceil \log n \rceil$
    (technique: the second moment method)
  - We will here present a randomized algorithm that w.h.p. finds a d.s. of size $(1 + \epsilon) \log n$ ($\epsilon > 0$ arbitrarily small), in polynomial time (thus, this algorithm is near-optimal).

The GREEDY-DS algorithm - basic idea

- choose a random vertex and put it in a dominating set under construction
- remove from graph all vertices adjacent to that vertex (they are "covered" by the vertex)
- repeat until the number of vertices left becomes small
- explicitly add those vertices to the dominating set under construction (obviously, the resulting set is a dominating set)

The algorithm - pseudo code

ALGORITHM "GREEDY-DS"

Input: a random instance $G(V, E)$ of $G_{n,p}$ $(p = \frac{1}{2})$

(1) $i \leftarrow 0; V_i \leftarrow V; D \leftarrow \varnothing$

(2) **until** $|V_i| \leq \epsilon \log n$ **do**

begin

choose a random vertex $u_i \in V_i$

$V' \leftarrow V_i - N_i$ (N_i: neighor vertices of u_i)

$D \leftarrow D \cup \{u_i\}$ (add the vertex to the evolving d.s.)

$i \leftarrow i + 1$

$V_i \leftarrow V'$

end

(3) $D \leftarrow D \cup V_i$ (add the vertices left)

(4) **return** D

Analysis - informal

- in each repetition about half (because of $p = \frac{1}{2}$) of the vertices left are "covered" by the random vertex and are removed from the graph
- thus, after $\log n$ repetitions the entire graph is almost covered and the number of vertices left drops to less than $\epsilon \log n$
- Then, these $\epsilon \log n$ vertices are explicity added to the dominating set under construction
- The resulting set has size $\log n + \epsilon \log n = (1 + \epsilon) \log n$ and is obviously a dominating set.

Two comments

(1)

- when we choose a vertex, we "expose" its neighbours, so they can not be assumed "random" anymore (e.g. we know exactly their number, which is not a random variable anymore)
- however these exposed vertices are anyway removed from the graph, so the rest graph (the vertices left) remains random and the randomized analysis remains valid.

(2)

- the Chernoff bound is used to show concentration of the number of "covered" vertices in each repetition around the expected number (which is easy to compute)
- as said, the bound is polynomially fast approaching 1 as long as the mean (of the vertices left) is logarithmic! This is why when this number gets logarithmic, the vertices left are explicitly added to the dominating set.

Analysis - Chernoff bound

Lemma 1 Let $\beta \in [0,1]$. Then for any constant $\epsilon > 0$ and given that $|V_i| \geq \epsilon \log n$ it is:

$$\Pr\left\{|N_i| \geq (1-\beta)\frac{|V_i|}{2}\right\} \geq 1 - n^{-\frac{\beta^2}{4}\epsilon}$$

Proof: The vertices "covered" at repetition i (set N_i) follow a Binomial distribution with parameters $|V_i|, \frac{1}{2}$. In other words, we have $|V_i|$ Bernoulli trials each one with success probability $\frac{1}{2}$. Thus the mean μ_i of their sum ($|N_i|$) is bounded by the Chernoff bound as follows:

$$\Pr\{|N_i| \geq (1-\beta)\mu_i\} \geq 1 - e^{-\frac{\beta^2}{2}\mu_i} = 1 - e^{-\frac{\beta^2}{2}\frac{|V_i|}{2}} \geq$$

$$\geq 1 - e^{-\frac{\beta^2}{2}\frac{\epsilon \log n}{2}} = 1 - n^{-\frac{\beta^2}{4}\epsilon} \qquad \square$$

Analysis - time complexity

- Let the event $\mathcal{E}_i$= "at the i-th random repetition it is $|N_i| \geq (1-\beta)\frac{|V_i|}{2}$"
- Let the event $\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2 \cap \cdots \cap \mathcal{E}_t$ until $|V_t| < \epsilon \log n$
- Lemma 2: Assuming (probabilistically) event $\mathcal{E}$ the number t of repetitions of the random loop of the algorithm is at most $(1+\epsilon')\log n$, where $\epsilon' > 0$ constant.

  Proof: After repetition i:
  $|V_{i+1}| = |V_i| - |N_i| \leq |V_i| - \frac{(1-\beta)}{2}|V_i| = (1-\gamma)|V_i|$ where $\gamma = \frac{1}{1-\beta}$.
  Recursively: $|V_t| \leq (1-\gamma)^t n$, so for $|V_t| \leq \epsilon \log n$ we need $t \geq \frac{\log n}{\log(\frac{1}{1-\gamma})} + \Theta(\log\log n)$. We note that $\frac{1}{1-\gamma} = \frac{2}{1+\beta}$ so by choosing (for any $\epsilon' > 0$) $\beta = 2^{\frac{\epsilon'}{1+\epsilon'}-1}$ we finally get $t \leq (1+\epsilon')\log n$ □

Analysis - continued

- Lemma 3: For any constants $\beta \in [0,1], \epsilon > 0$ it is:
$$\Pr\{\mathcal{E}\} \geq 1 - n^{-\frac{\beta^2}{8}\epsilon}$$

  Proof: From previous lemmata, it is
  $\Pr\{\overline{\mathcal{E}}\} \leq \sum_i \Pr\{\overline{\mathcal{E}_j}\} \leq t \cdot n^{-\frac{\beta^2}{4}\epsilon} \leq (1+\epsilon')\log n \cdot n^{-\frac{\beta^2}{4}\epsilon} \leq n^{-\frac{\beta^2}{8}\epsilon}$

- Time complexity: Clearly, the algorithm needs time $O((1+\epsilon')n\log n)$ with probability at least $1 - n^{-\frac{\beta^2}{8}\epsilon}$.

- The constructed dominating set has size at least $(1+\epsilon'+\epsilon)\log n$

Summary (I)

- The algorithm constructs a near optimal dominating set (i.e. the approximation ratio to the optimal $\log n$ size is $1+\epsilon$, where $\epsilon > 0$ arbitrarily small).
- The time complexity is polynomial (both in expectation and w.h.p.)
- However the probability of good performance, although tending to 1, is not "polynomially large" since in the $1 - n^{-\frac{\beta^2}{8}\epsilon}$ bound the $\frac{\beta^2}{8}\epsilon$ constant is just positive, (not necessarily larger than 1)

Summary (II)

- In the full paper (S. Nikoletseas and P. Spirakis, "Near-Optimal Dominating Sets in Dense Random Graphs in Polynomial Expected Time", in the Proceedings of the 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)) includes an enhanced algorithm (of repetitive trials) boosting this probability to $1 - n^{-a}$, where $a > 1$.