CMPSCI 690RA: Randomized Algorithms

Lecture 4 – Principle of Deferred Decisions & Stable Matchings

Andrew McGregor

Last Compiled: January 30, 2020

Outline

Clock Solitaire and Principle of Deferred Decisions

Stable Matching Problem

Probabilistic Analysis of Gale-Shapley Algorithm

Recall Last Week's Puzzle

- Take a standard pack of 52 cards that is randomly shuffled.
- Split into 13 piles of 4 and label piles $\{A,2,\ldots,10,J,Q,K\}$.
- Take first card from "K" pile.
- Take next card from "X" pile where X is the face value of the previous card taken.
- Repeat until either all cards are removed (you win) or we get stuck (you lose).

What's the probability you win?

Structural Observations

Lemma

The last card before we terminate (either winning or loosing) is K.

Proof.

- Suppose at iteration j we draw card X but pile "X" is empty.
- If pile 'X" is empty and X$\neq$K then we have already drawn 4 copies of card X prior to iteration j. Contradiction!

□

Lemma

We win iff the fourth K is the 52nd card.

Proof.

- When 1st, 2nd, or 3rd K is seen we don't terminate because "K" pile is non-empty.
- Terminate when 4th K is seen: we win iff it's the 52nd card.

□

Principle of Deferred Decisions

- How do we compute the probability that the fourth K is the 52nd card? $\mathbb{P}$[fourth K is 52nd card] equals:

$$\frac{\#\text{ game configurations such that K is 52nd card revealed}}{\#\text{ game configurations}}$$

- Principle of Deferred Decisions: Let the random choices unfold with the progress of the analysis rather than fixing random events upfront.
- For clock solitaire this means we may assume that at each draw, any unseen card is equally unlikely.

Theorem

The probability we win clock solitaire is $1/13$.

Outline

Stable Matching Problem

There are n job openings $(j_1, \ldots, j_n)$ and n applicants $(a_1, \ldots, a_n)$.

- A matching is a $1-1$ correspondence between jobs and applicants.
- Each job has a (strict) preference list for the applicants and each applicant has a (strict) preference list for the jobs.
- A matching is unstable if there exists job j and applicant a such that
  1. j and a are not matched to each other.
  2. j prefers a to their current employee.
  3. a prefers j to their current job.
- A configuration that is not unstable is stable.

Does a stable matching always exist? Can we find one efficiently?

The Gale-Shapley Algorithm

- Let i be the smallest value such that a_i is unemployed.
- a_i applies to the most desirable employer (according to their list) that hasn't already rejected them
- The jobs accepts then if either a) the job is currently unfilled, or b) a_i is more desirable than the current employee (in which case the current employee becomes unemployed.)
- Repeat until there are no unemployed applicants left.

Does the algorithm terminate? Is the resulting matching stable?

Algorithm is Well-Defined

Lemma

Whenever there's an unemployed candidate a_i, there is a job he/she hasn't applied to.

Proof.

- Once a job has an employee, the job is doesn't become unfilled.
- Therefore, all the jobs to which a_i applied are filled.
- If a_i has applied to all jobs, all the jobs are filled, hence all the applicants are employed. Contradiction!

$\square$

Algorithm is Efficient

Theorem

The algorithm terminates after $O(n^2)$ repeats.

Proof.

- At each stage of the algorithm, let t_i be the number of jobs to which a_i could still potentially apply.
- At each step $\sum_{i \in [n]} t_i$ decreases by 1.
- Initially $\sum_{i \in [n]} t_i = n^2$ so there can be at most n^2 steps.

$\square$

Algorithm is Correct

Theorem

The matching found by the Gale-Shapley algorithm is stable.

Proof.

- Proof by contradiction: Suppose matching includes a–j and a'–j' but a and j' prefer to be matched to each other.
- Since a prefers j' to j, he/she must have applied to j' before he applied to j.
- But at that point, j' must prefer its current match to a: either it already had a better match when a applied or it matched a initially and then got a better proposal. Contradiction!

□

Outline

Probabilistic Analysis

- In randomized algorithms, we consider algorithms that make random choices and investigate what happens when they process a "fixed input." E.g., the 2SAT algorithm from lecture 1.
- In probabilistic analysis, we consider random input and investigate what happens when it's processed by a fixed algorithm. E.g., the Gale-Shapley algorithm when the preference lists are random.

Theorem

If the preference lists are random, the expected number of iterations of Gale-Shapley is $\leq nH_n$.

Probabilistic Analysis of Gale-Shapley Algorithm (1/2)

- Principle of deferred decision: we may assume that at each step a_i applies to a job chosen uniformly at random from the jobs that have no yet rejected the applicant.
- To simplify things, use a modification of the Gale-Shapley algorithm, the "amnesiac" algorithm.
  - At each step, a_i applies to a job uniformly at random from the set of all n jobs.
  - This doesn't change the outcome of the algorithm since, if a_i was rejected a job before, they'll be rejected again.
  - The expected running time of the modified algorithm is an upper bound for the running time of original algorithm.

Probabilistic Analysis of Gale-Shapley Algorithm (2/2)

Theorem

If the preference lists are random, the expected number of iterations of Gale-Shapley is at most nH_n.

Proof.

Since the algorithm terminates once all jobs have received at least one applicant, the random process is analogous to the coupon collector problem. $\square$