

Lecture 7: “The Principle of Deferred Decisions”

Sotiris Nikolettseas
Professor

CEID - ETY Course
2017 - 2018

A1. The Principle of Deferred Decisions

A2. The Proposal Algorithm for the Stable Marriage Problem

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 losing) 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 j th iteration, a contradiction. \square

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:

$$\begin{aligned} \Pr\{\text{win}\} &= \Pr\{\text{4th "K" at the 52nd iteration}\} = \\ &= \frac{\#\text{game evolutions: 52nd card} = \text{4th "K"}}{\#\text{all game evolutions}} \end{aligned}$$

- 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.

⇒ 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 (and 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 $13 \times 4 = 52$ 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, \dots, w_n) and n men (m_1, \dots, 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 exist? (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 . □

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 $c \in \mathbb{R}$) then the time T for collecting all n coupons obeys the following:

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

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)