Lagrangian Relaxation via Randomized Rounding, *Introduction*

Neal E. Young

UCR, 1/28/04

Lagrangian relaxation algorithms

[1950] von Neumann. Numerical method for determination of the value and the best strategies of a zero-sum two-person game with large numbers of strategies.

[1950] Brown and von Neumann. Solutions of games by differential equations.

[1952] Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations.

[1958] Ford and Fulkerson. A suggested computation for maximal multicommodity flow.

[1960] Dantzig and Wolfe. Decomposition principle for linear programs.

[1962] Benders. Partitioning procedures for solving mixed-variables programming problems.

[1971] Held and Karp. The traveling salesman problem and minimum spanning trees.

[1977] Khachiyan. Convergence rate of the game processes for solving matrix games.

...

[1979] Shapiro. A survey of Lagrangean techniques for discrete optimization.
Annals of Discrete Mathematics, 5:113--138, 1979.

Lagrangian relaxation algorithms

Shahrokhi and Matula. The maximum concurrent flow problem. JACM, 1990.

Klein, Plotkin, Stein, and Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. SICOMP, 1994.

Goldberg. A natural randomization strategy for multicommodity flow and related problems. IPL, 1992.

Awerbuch and Leighton. A simple local-control approximation algorithm for multicommodity flow. FOCS, 1993.

Luby and Nisan. A parallel approximation algorithm for positive linear programming. STOC, 1993.
--- incrementing multiple variables simultaneously

Awerbuch and Leighton. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. STOC, 1994.

Leighton, Makedon, Plotkin, Stein, Tardos, and Tragoudas. Fast approximation algorithms for multicommodity flow problems. JCSS, 1995.

Plotkin, Shmoys, and Tardos. Fast approximation algorithms for fractional packing and covering problems. MOR, 1995.

Radzik. Fast deterministic approximation for the multicommodity flow problem. SODA, 1995.

Lagrangian relaxation algorithms

Grigoriadis and Khachiyan.
A sublinear-time randomized approximation algorithm for matrix games. OR Research Letters, 1995.
An exponential-function reduction method for block-angular convex programs. Networks, 1995.

Young. Randomized rounding without solving the linear program. SODA, 1995.

Karger and Plotkin. Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows. STOC, 1995.

Grigoriadis and Khachiyan.
Coordination complexity of parallel price-directive decomposition. MOR, 1996.
Approximate minimum-cost multicommodity flows in o(knm/ε^2) time. Math. Programming, 1996.

Garg and Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. FOCS, 1998. --- variable-size increments

Konemann. Fast Combinatorial Algorithms for Packing and Covering Problems
PhD thesis, Max-Planck-Institute for Informatik, 2000. --- dropping met covering constraints

Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. SIAM J. Discrete Math, 2000. --- partitioning increments into phases

The probabilistic method

In order to prove the existence of a combinatorial structure with certain properties, we construct an appropriate probability space and show that a randomly chosen element in the space has the desired properties with positive probability.

- Alon, Spencer, Erdos: "**The Probabilistic Method**" (1992)

(applications in combinatorics, graph theory, number theory, combinatorial geometry, computer science.)

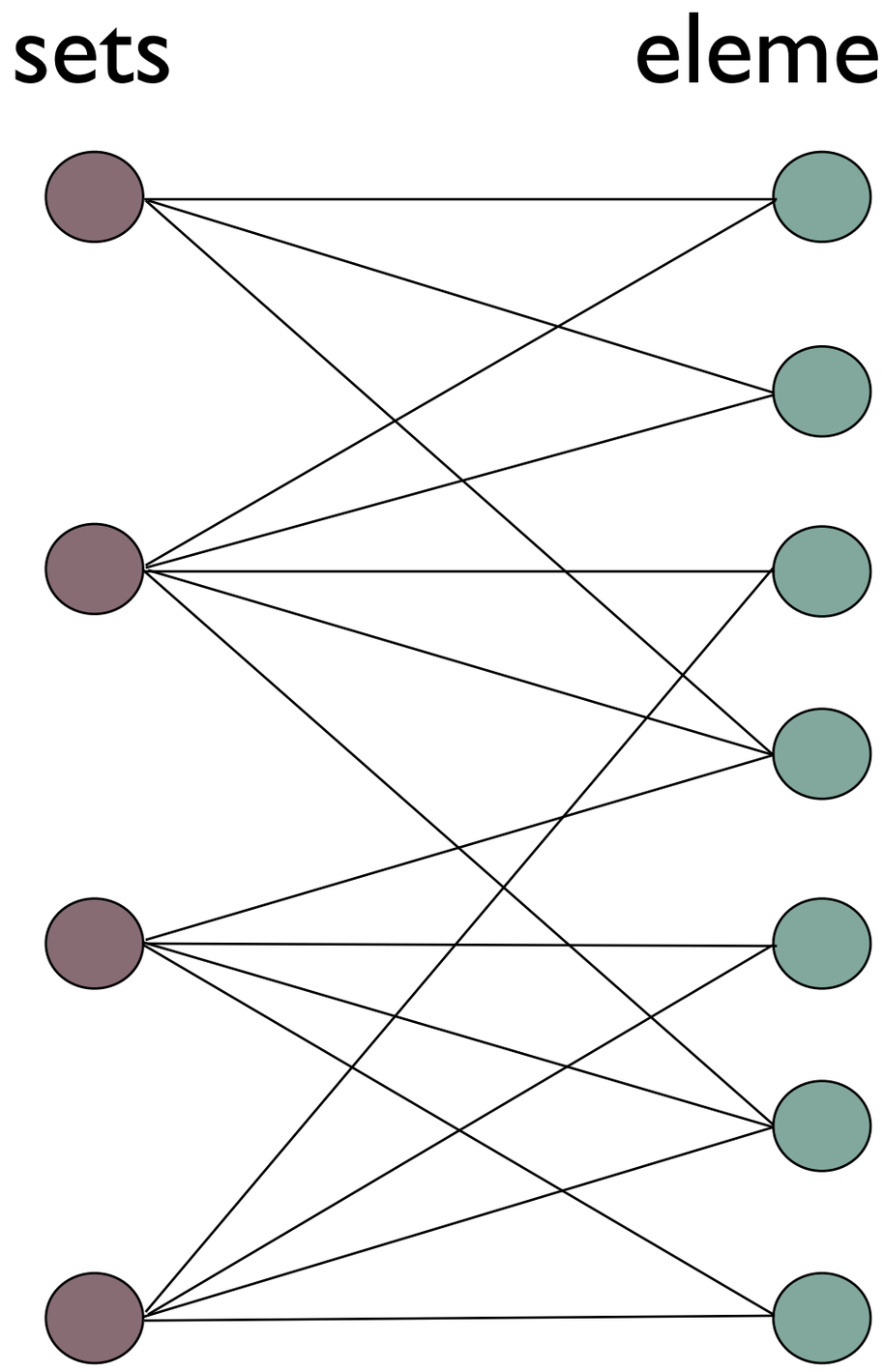
Randomized rounding for approximation algorithms

"For each of the problems we consider, we first show the existence of a provably good approximate solution using the probabilistic method [1]. [We then] show that the probabilistic existence proof can be converted, in a very precise sense, into a deterministic approximation algorithm. To this end we use an interesting "method of conditional probabilities"... We apply our method to integer programs arising in packing, routing, and maximum multicommodity flow...

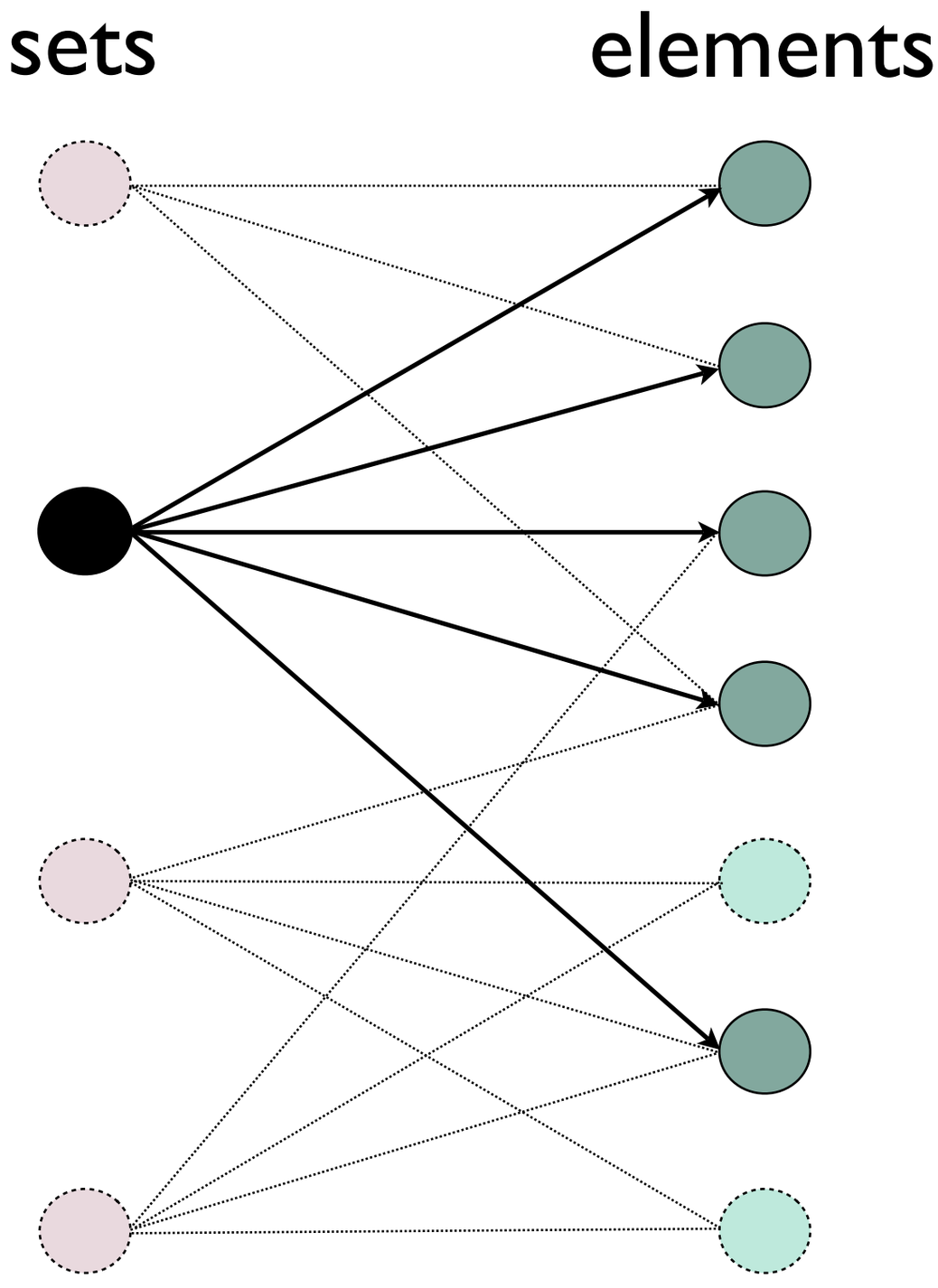
The time taken to solve the linear program relaxations of the integer programs dominates the net running time theoretically (and, most likely, in practice as well)."

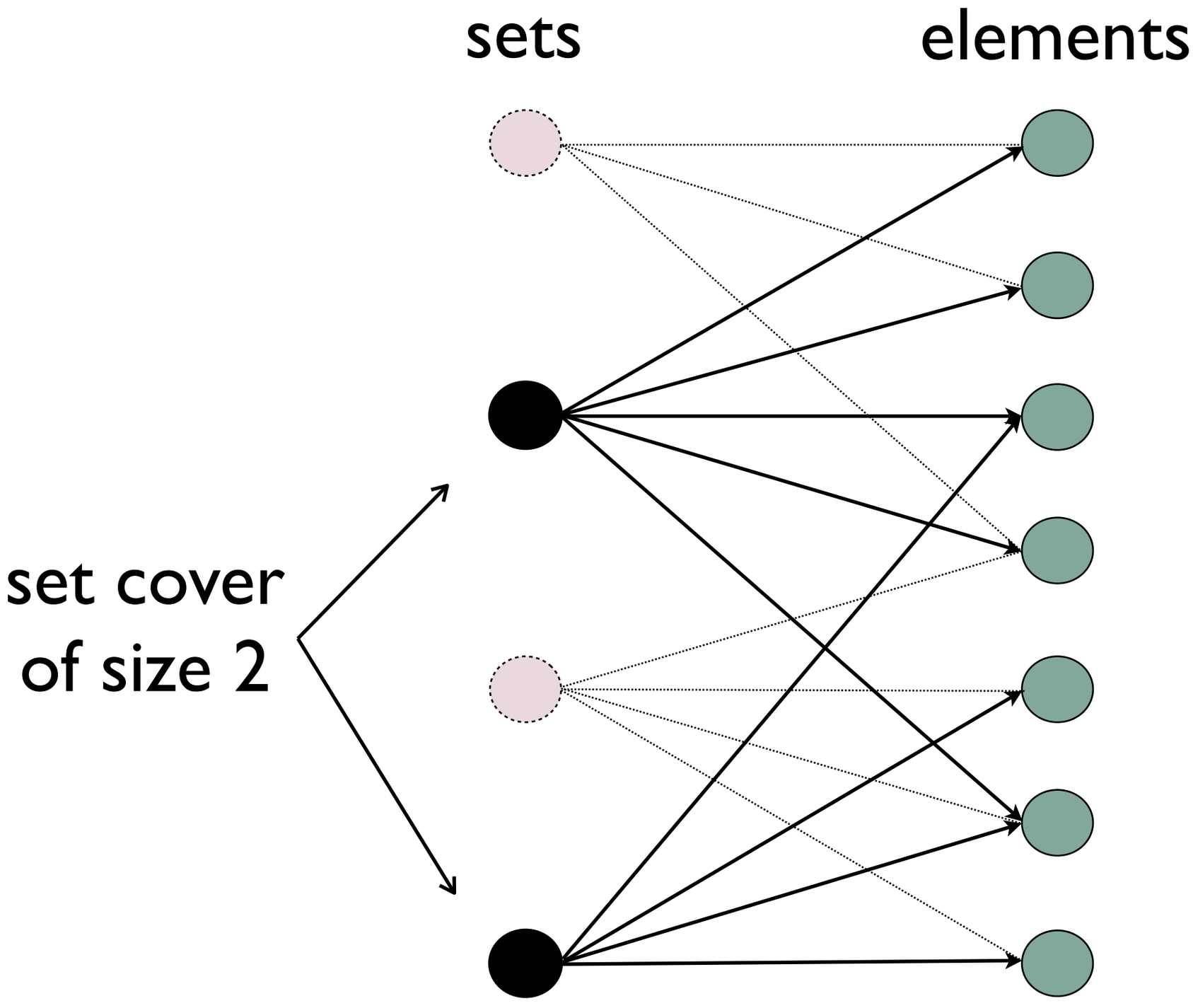
-- Raghavan (1988)

goal: minimum-size
set cover

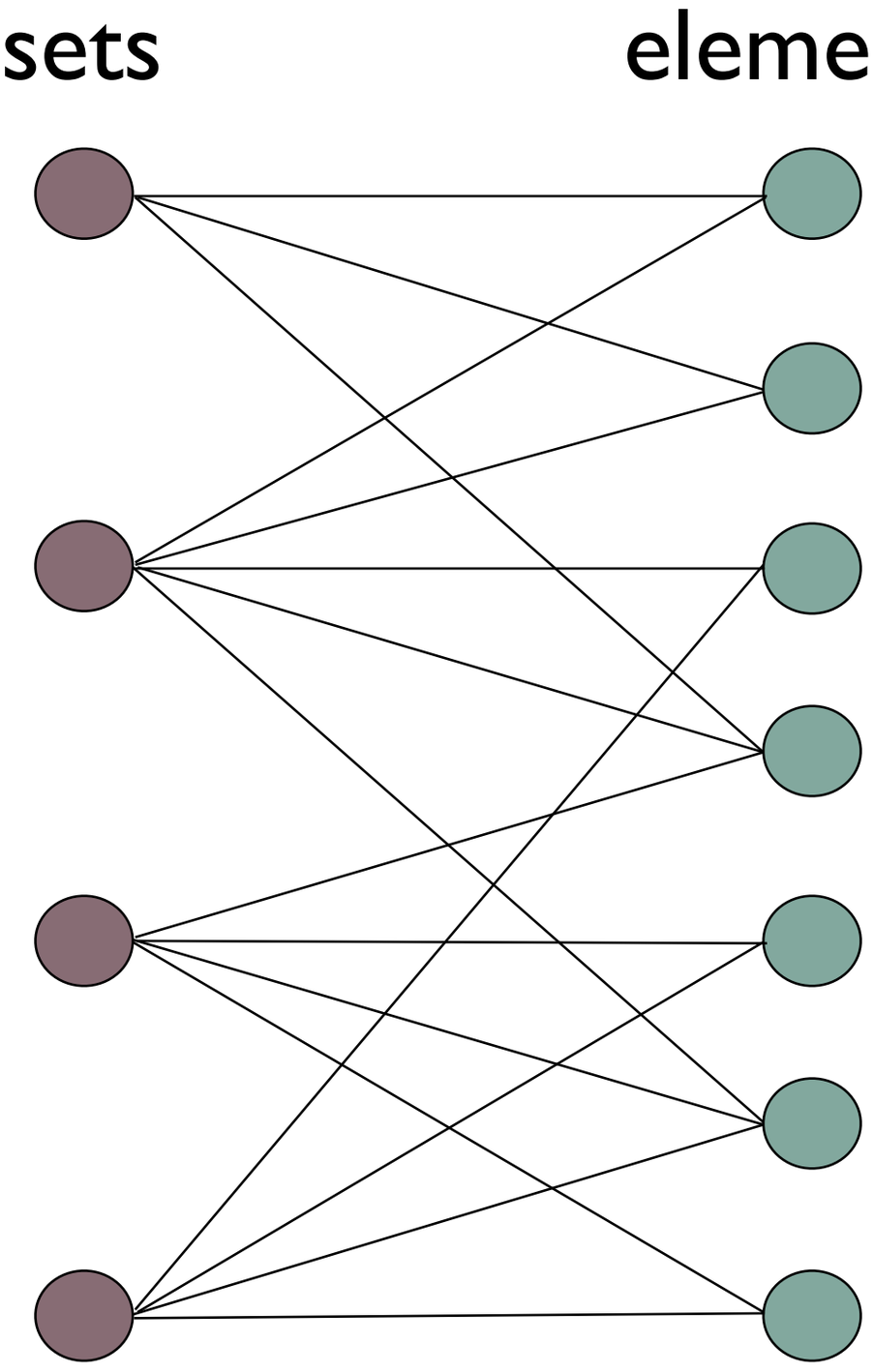


set cover problem

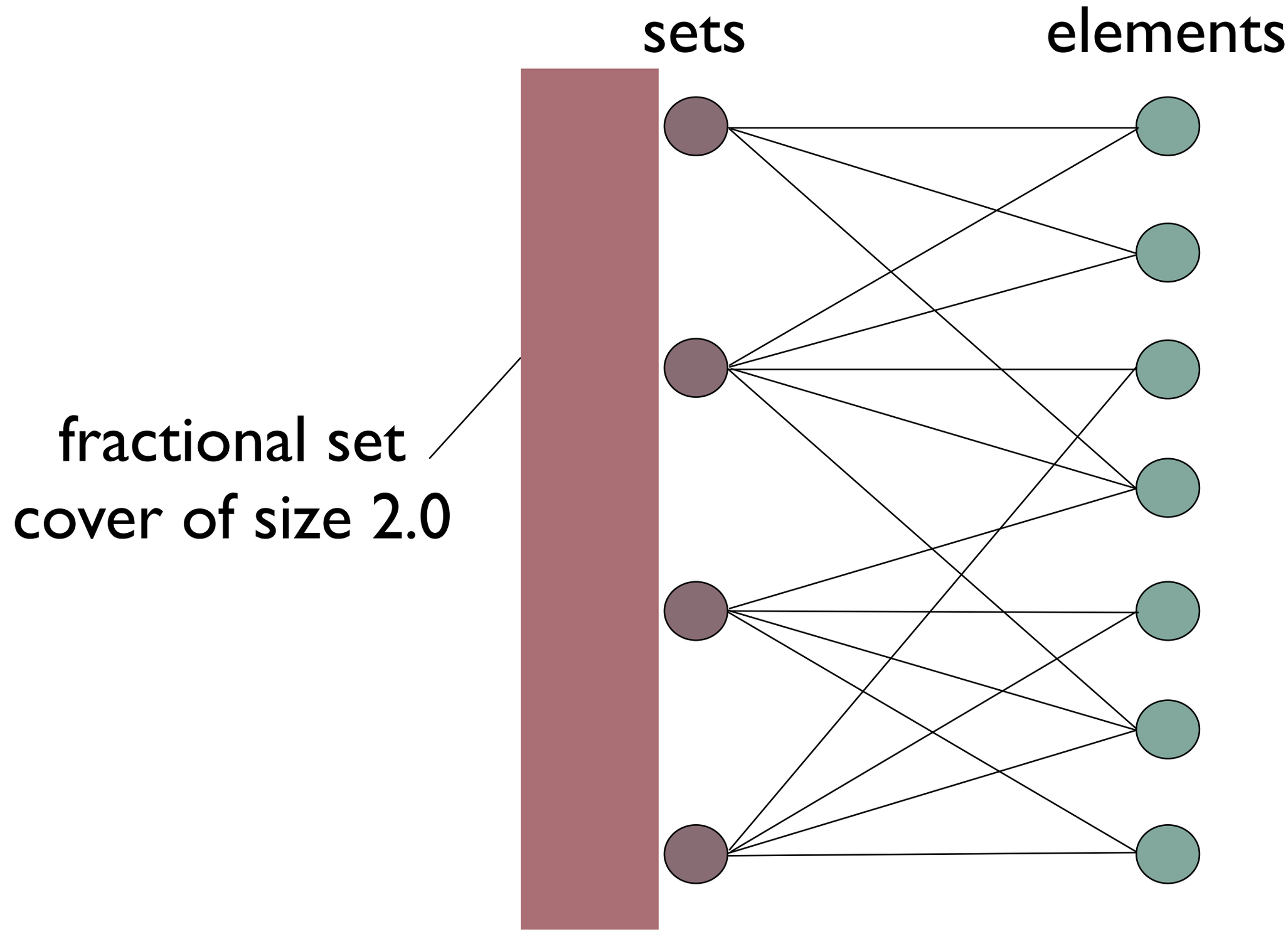

sets
elements


sets
elements
set cover
of size 2

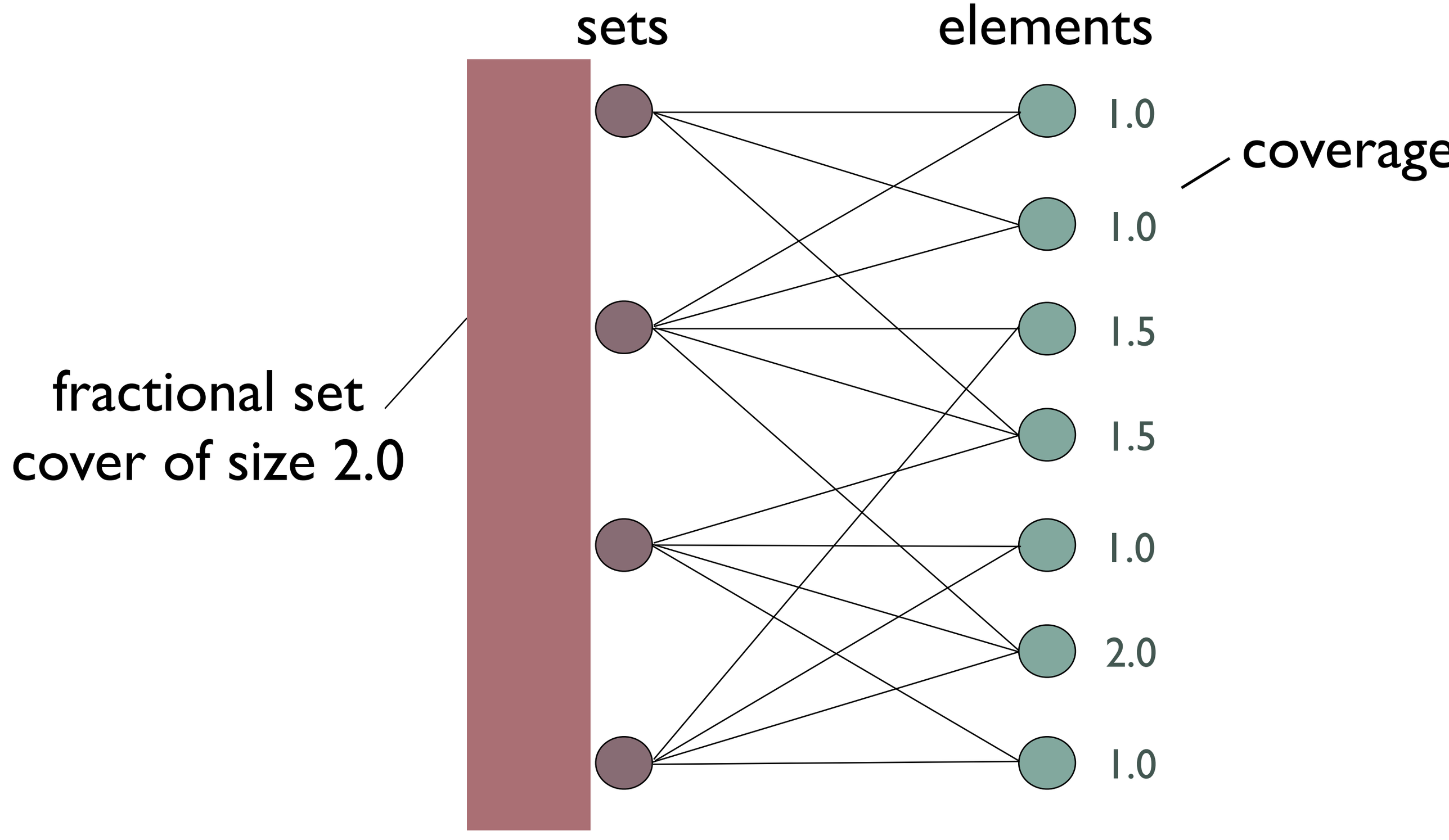
goal: minimum-size
fractional
set cover



fractional set cover problem



fractional set cover problem



fractional set cover problem

approximation algorithm for set cover

randomized rounding scheme

1. Compute optimal fractional set cover x*.
2. Randomly round x* to get collection S of sets.
3. Return S.

how?

analysis

With non-zero probability:

S is a set cover, and

size(S) ≤ log(n) size(x*) ≤ log(n) size(OPT).

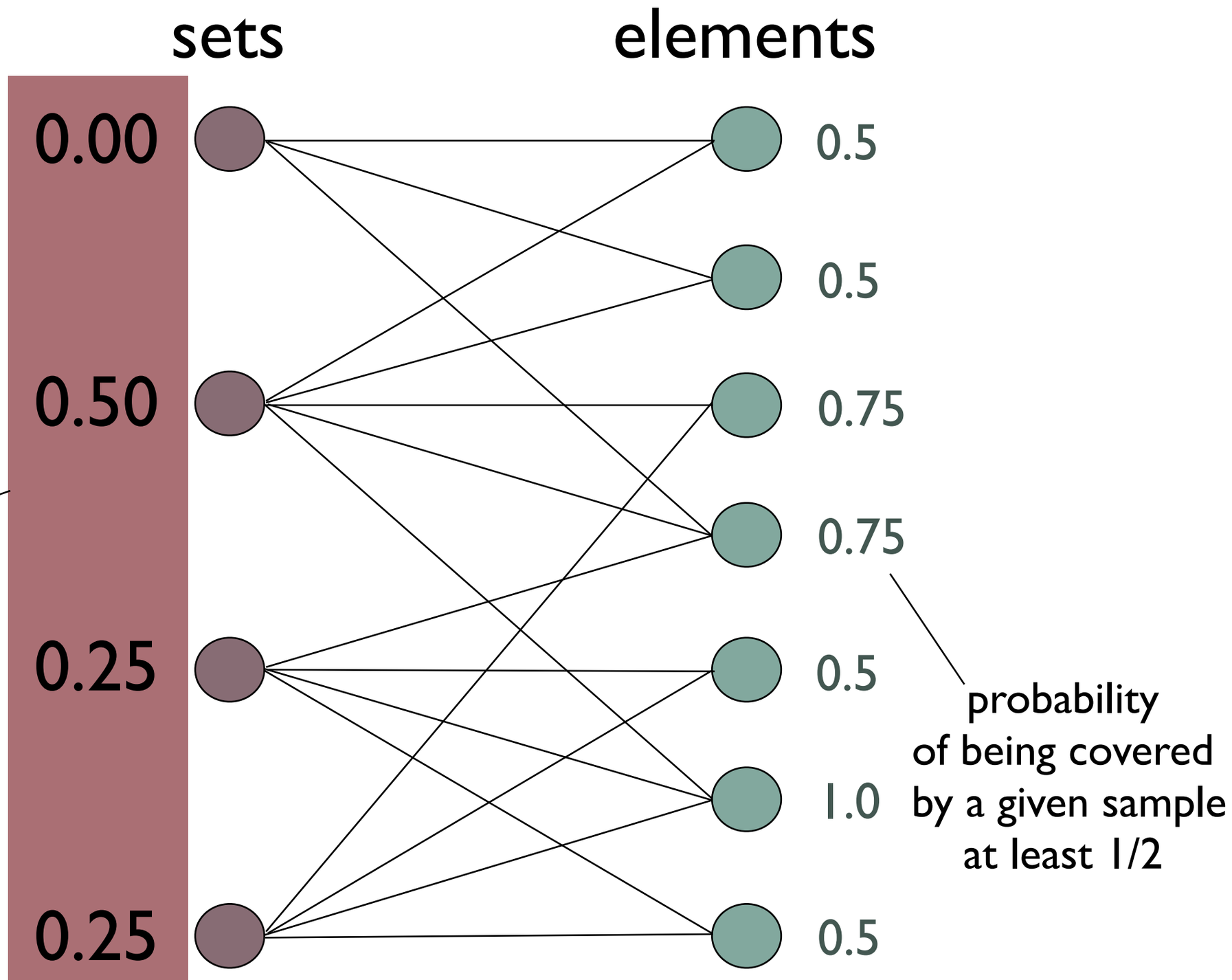
goal: convert fractional cover into true cover

<u>step 1</u>

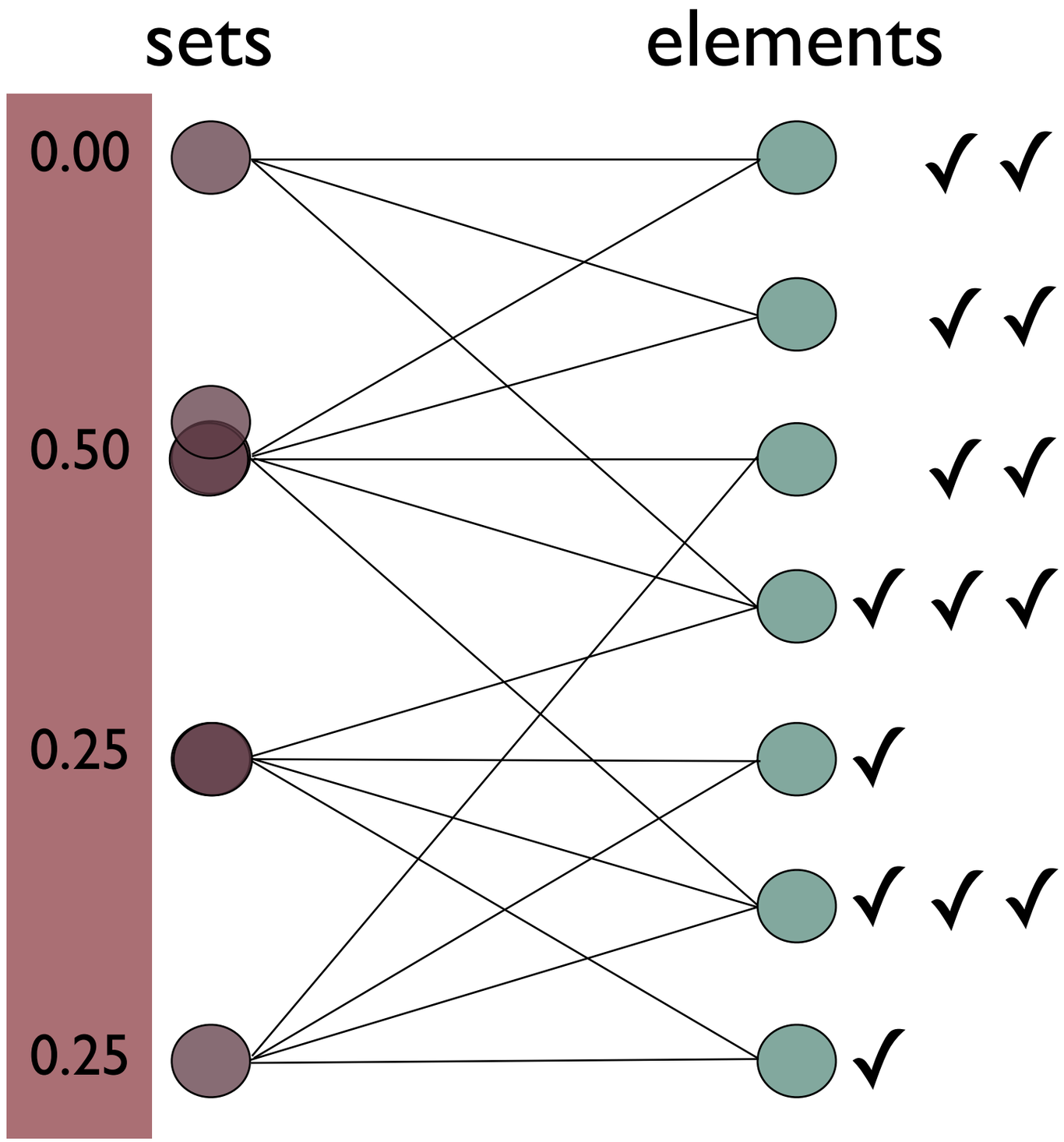
scale fractional cover by 1/2 to get probability distribution p on sets

<u>step 2</u>

sample from p T times



randomized rounding scheme



random sampling, T=3

randomized rounding scheme

1. Compute optimal fractional set cover x*.
2. Compute probability distribution p on sets.
3. Repeat T = ln(n) size(x*) times:
4. Choose a random set S according to p.
5. Return chosen sets.

Randomly sample T sets from distribution defined by x.*

Analysis

- Size of S is T (guaranteed). T = ln(n) size(x*).
- Each iteration, each element covered with probability ≥ 1/size(x*).
- Expected number of elements left uncovered after T rounds:

$$n\left[1-\frac{1}{\text{size}(x^*)}\right]^T < n\exp(-T/\text{size}(x^*))$$

$$= 1$$

Therefore, with positive probability, S is a cover

method of conditional probabilities

randomized rounding scheme

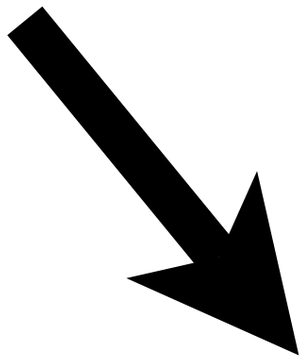
```
1.Compute optimal fractional set cover x*.
2.Randomly round x* to get collection S of sets.
3.Return S.
```

analysis

With non-zero probability:

S is a cover.

size(S) ≤ log(n) size(x*) ≤ log(n) size(OPT).



deterministic algorithm

```
1. Compute optimal fractional set cover x*.
2. Deterministically round x* to get S.
3. Return S.
```

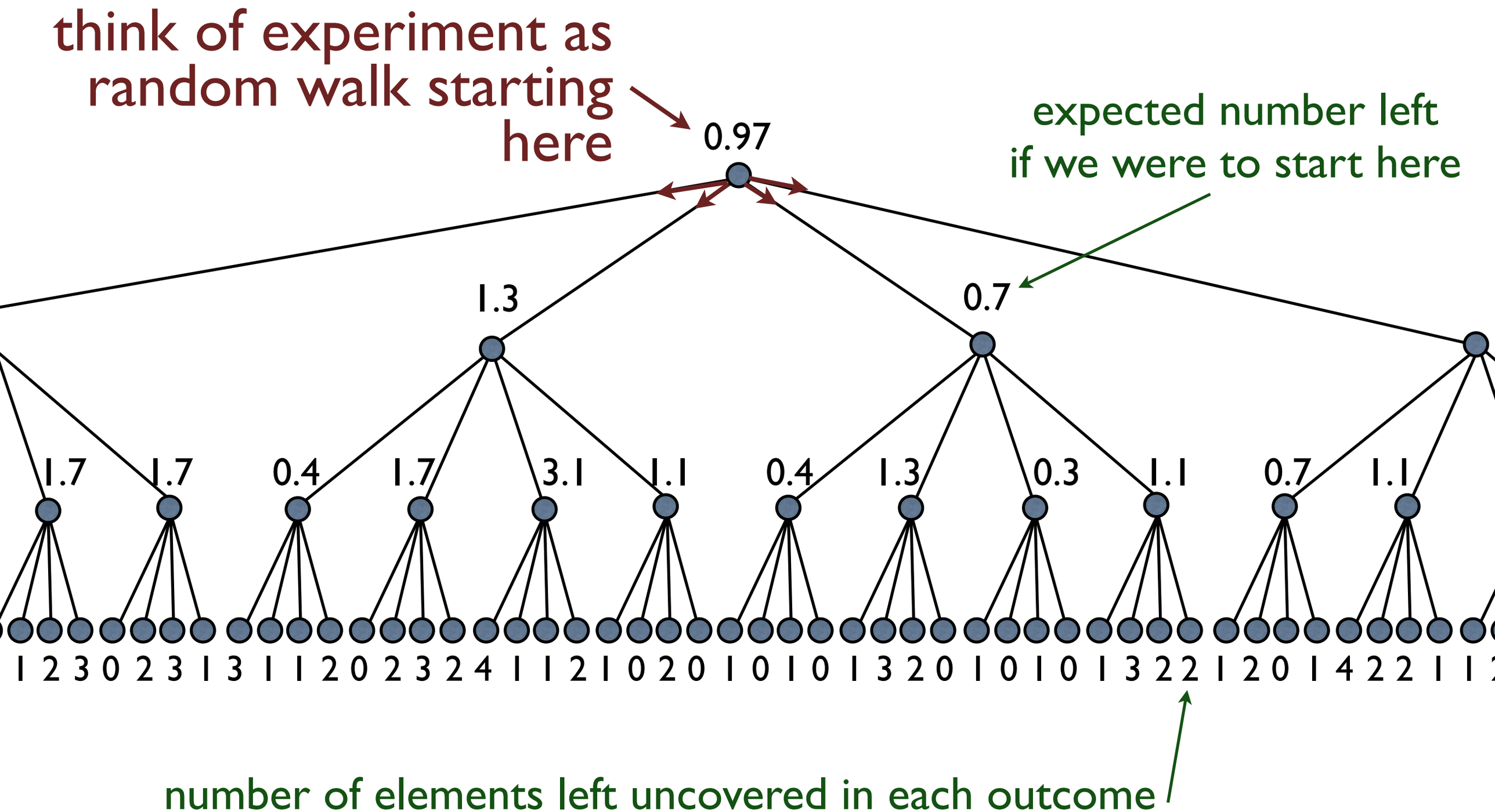
bottleneck

analysis

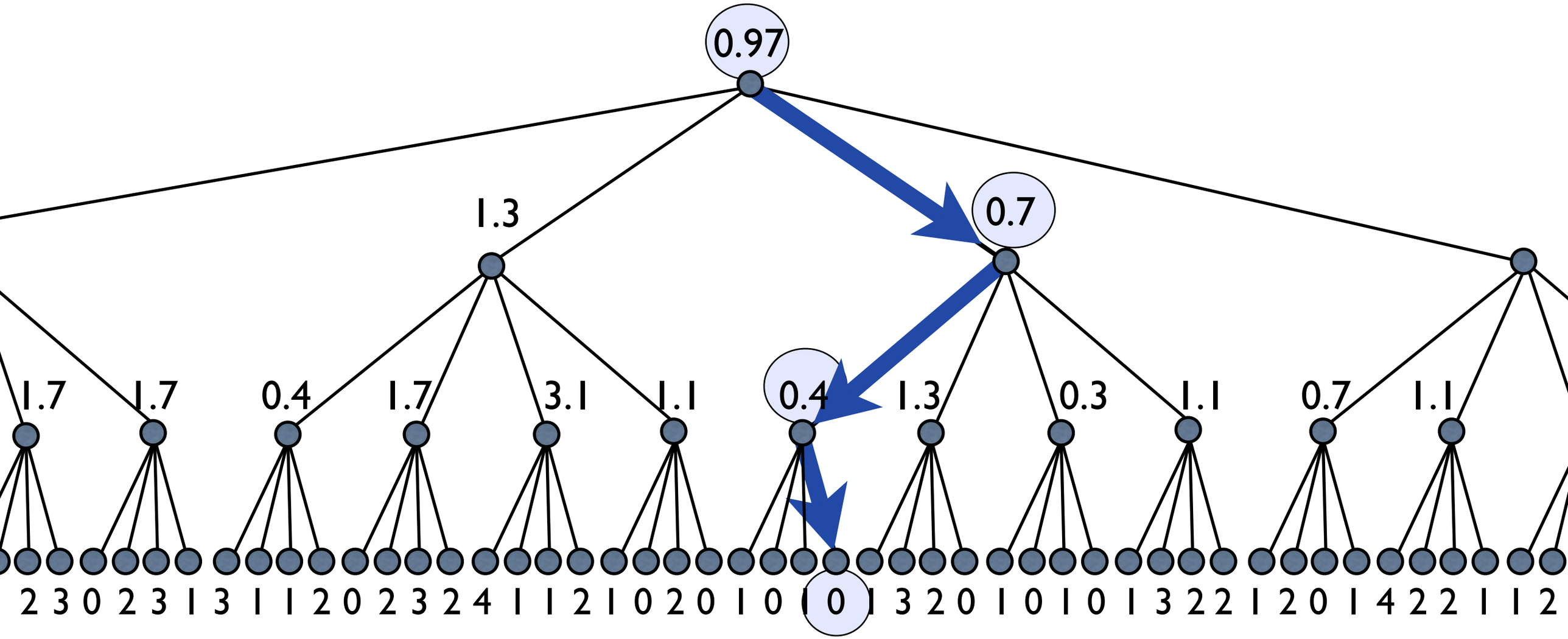
Always!

1. S is a cover.
2. size(S) ≤ log(n) size(x*) ≤ log(n) size(OPT).

method of conditional probabilities



replace each random choice w/ deterministic one



make each choice deterministically
to keep conditional expectation less than 1

Need to compute conditional expectations

Given the choices made in the first t iterations:

- Let n_t be the number of elements not yet covered.
- Expected number of elements left uncovered after T-t more rounds:

$$n_t \left[1 - \frac{1}{\text{size}(x^*)}\right]^{T-t}$$

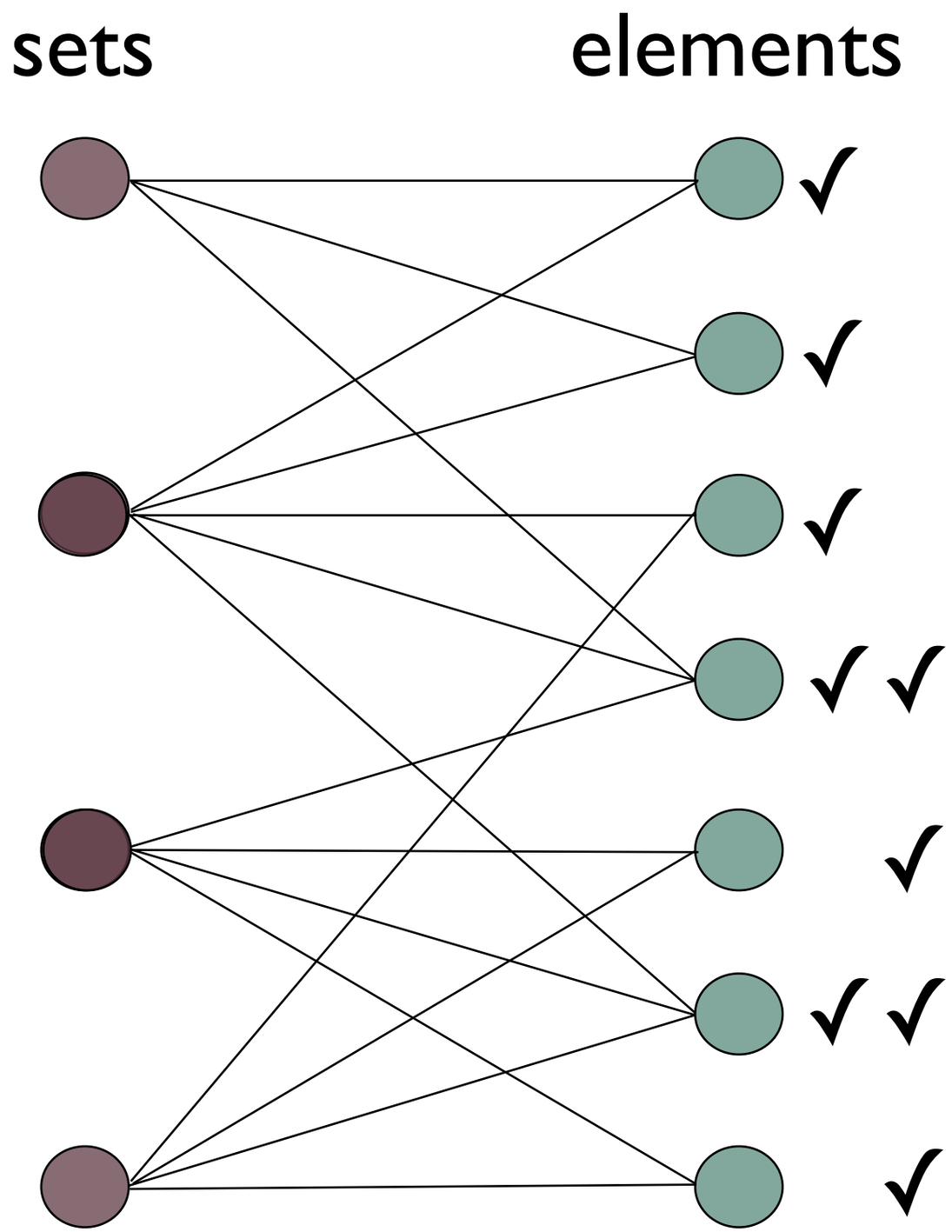
Choose each set to minimize this.
Choose each set to minimize n_t.

derandomized algorithm

1. Repeat T times (or until all elements covered):
2. Choose set to minimize
   number of elements not yet covered.
3. Return chosen sets.

Choose sets to minimize conditional expectation.

≡ *greedy algorithm [Johnson, Lovász 1974]*



greedy algorithm

method of conditional probabilities

randomized rounding scheme

```
1.Compute optimal fractional set cover x*.
2.Randomly round x* to get collection S of sets.
3.Return S.
```

analysis

With non-zero probability:

S is a cover.

size(S) ≤ log(n) size(x*) ≤ log(n) size(OPT).

the big mystery

deterministic algorithm

~~1. Compute optimal fractional set cover x*.~~

2. Deterministically round x* to get S.
3. Return S.

analysis

Always:

1. S is a cover.
2. size(S) ≤ log(n) size(x*) ≤ log(n) size(OPT).

randomized rounding scheme based on random sampling

method of conditional probabilities

greedy algorithm

randomized rounding scheme based on random sampling

existence proof

method of conditional probabilities

greedy algorithm

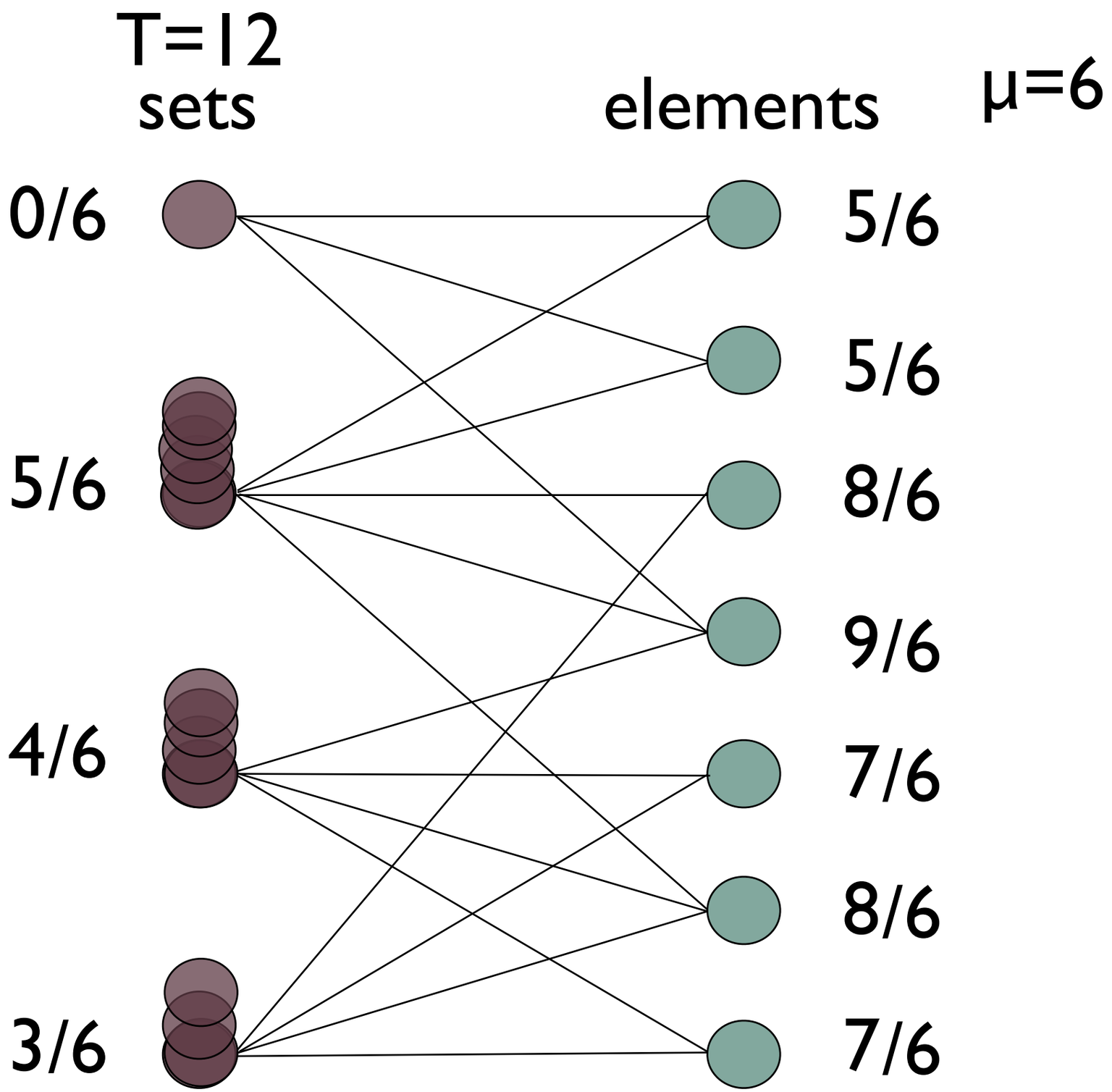
performance guarantee

rounding scheme for *fractional* set cover

1. Compute optimal fractional set cover x*.
2. Compute probability distribution p on sets.
3. Let $\mu = 2\ln(n)/\epsilon^2$.
4. Repeat $T = \text{size}(x^*)\mu$ times:
5. Choose a random set S according to p.
6. Let x(S) = (#times S chosen)/μ.
7. Return x.

Randomly sample T sets from distribution defined by x.*
(Expect each element to be covered at least μ times.)
Assign each set s weight x(s) = # times s chosen / μ.


andom


T=12
sets
elements
μ=6
0/6
5/6
4/6
3/6
5/6
5/6
8/6
9/6
7/6
8/6
7/6

thm: *With pos. prob. x is a (1-ϵ)-cover.*

Consider an element e.

E[# times e covered] ≥ μ.

Pr[# times e covered < (1-ϵ)μ] < 1/n *(Chernoff, '52)*

Expected # insufficiently covered elements < 1.

With pos. prob., x covers every e at least 1-ϵ.

Note: size(x) = T/μ = size(x). (guaranteed)*

derandomized algorithm

1. Let $\mu = 2\ln(n)/\epsilon^2$.
2. Repeat until every element is covered μ times:
3. Choose a set S that maximizes

$$\sum_{e \in S} (1-\epsilon)^{(\#\text{times e covered so far})}.$$

4. Let x(S) = (#times S chosen)/μ.
5. Return x.

from Chernoff

Choose each set to minimize $\sum_e (1-\epsilon)^{(\#\text{times e covered so far})}$.

Assign each set s weight x(s) = (# times chosen) / μ.

Yields (1-ϵ)-cover x, size(x) ≤ size(x*).

At most $T = O(\log(n)\ \text{size}(x^*)/\epsilon^2)$ iterations.

multi-commodity flow

input: set of paths P in directed graph

maximize $\Sigma_{p \in P}$ f(p) subject to

for each edge e, $\Sigma_{p \ni e}$ f(p) ≤ μ.

rounding scheme for multicommodity flow

1. Compute optimal multicommodity flow f*.
2. Compute probability distribution q on paths.
3. Let $\mu = 3\ln(n)/\epsilon^2$.
4. Repeat $T = \text{size}(f^*)\mu$ times:
5. Choose a random path p according to q.
6. Let f(p) = (#times p chosen)/μ.
7. Return p.

Randomly sample T paths from distribution defined by f.*
(Expect each edge to be covered at most μ times.)
Assign each path p flow f(p) = # times p chosen / μ.

derandomized algorithm

1. Let $\mu = 3\ln(n)/\epsilon^2$.
2. Repeat until flow on some edge would exceed μ:
3. Choose a path p that minimizes

$$\sum_{e \in p} (1+\epsilon)^{(\#\text{times e covered so far})}.$$

4. Let f(p) = (#times p chosen)/μ.
5. Return f.

Yields (1+ϵ)-feasible flow f, size(f) ≥ size(f*).

At most T = O(m log(n) /ϵ^2) iterations.

More...

- Random stopping times
- Probabilistic tools (Chernoff-like bounds)

- packing and covering - linear programs and integer linear programs with non-negative coefficients (k-medians, facility location)
- parallel algorithms

- beyond packing and covering (frac. Steiner tree)
- convex (non-linear) programs
- dynamic problems
- relations to on-line algorithms, learning theory?