Beating Simplex for Packing and Covering

Christos Koufogiannakis and Neal E. Young

May 13, 2007

packing and covering

packing

given: matrix A, vectors b, c

find vector $x \geq 0$ maximizing linear function $c^{\mathsf{T}}x$
subject to linear constraints $Ax \leq b$.

covering

given: matrix A, vectors b, c

find vector $x \geq 0$ minimizing linear function $c^{\mathsf{T}}x$
subject to linear constraints $Ax \geq b$.

For this talk, assume $A \in \{0,1\}^{n \times n}$, $b_j = c_i = 1$.

(Results extend to arbitrary nonnegative A, b, c.)

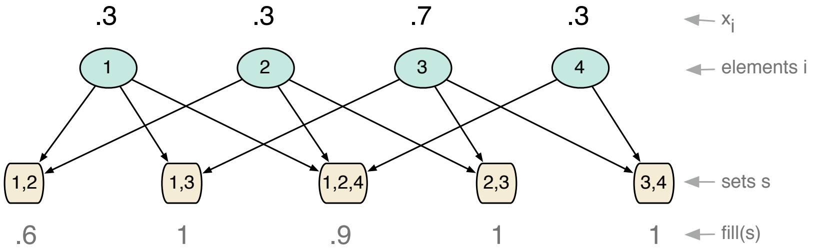
working example — packing

given: collection of sets

variables: x_i for each element i (call vector x a *packing*)

objective: maximize total weight $\sum_i x_i$

constraints: $\mathrm{fill}(s) \le 1$ for each set s,
where $\mathrm{fill}(s) = \sum_{i \in s} x_i$



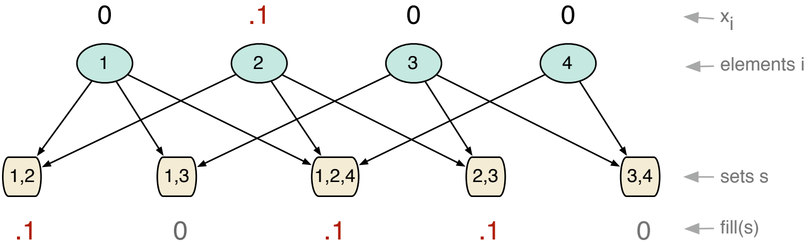
greedy algorithm?

1. $x \leftarrow \mathbf{0}$ —*maximize* $\sum_i x_i \,:\, \max_s \mathrm{fill}(s) \le 1$
2. repeat:
3. increase single x_i by ε,
   choosing i so increase in $\max_s \mathrm{fill}(s)$ is minimized.
4. return $x / \max_s \mathrm{fill}(s)$ — *note: scaling ensures fill* ≤ 1

greedy algorithm?

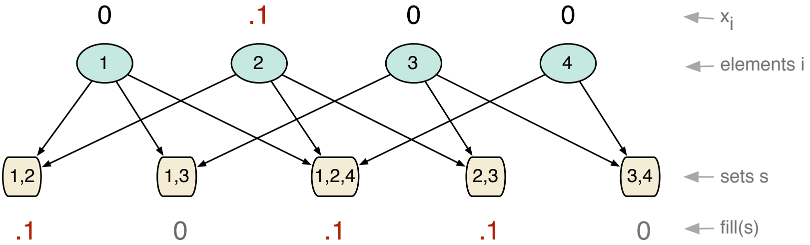
—*maximize* $\sum_i x_i$: $\max_s \text{fill}(s) \leq 1$

1. $x \leftarrow \mathbf{0}$
2. repeat:
3. increase single x_i by ε,
   choosing i so increase in $\max_s \text{fill}(s)$ is minimized.
4. return $x/\max_s \text{fill}(s)$ — *note: scaling ensures fill* ≤ 1



greedy algorithm?

1. $x \leftarrow \mathbf{0}$ —*maximize* $\sum_i x_i$: $\max_s \text{fill}(s) \leq 1$
2. repeat:
3. increase single x_i by ε,
   choosing i so increase in $\max_s \text{fill}(s)$ is minimized.
4. return $x / \max_s \text{fill}(s)$ — *note: scaling ensures fill* ≤ 1



what about: (1) number of sets x_i occurs in?
(2) non-max-fill sets?

Lagrangian relaxation

Replace hard constraints by smooth penalties — like in life.

for "vector of concerns" y, define:

$$\mathrm{lmax}(y) \;=\; \ln \sum_{i=1}^{n} e^{y_i}$$

1. lmax approximates max:

$$\mathrm{lmax}(y) \;\approx\; \max_i y_i + \ln n$$

2. but lmax is smooth (1st-order approximation is good):

$$\mathrm{lmax}(y+d) \;\approx\; \mathrm{lmax}(y) \;+\; d \cdot \nabla\, \mathrm{lmax}(y)$$

– provided $\max_i d_i \le \varepsilon$

relaxed algorithm uses lmax instead of max

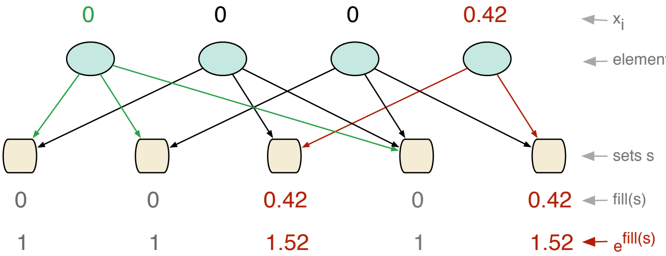
1. $x \leftarrow \mathbf{0}$ — *maximize* $\sum_i x_i \,:\, \max_s \mathrm{fill}(s) \leq 1$
2. repeat:
3. increase single x_i by ε,
   choosing i so increase in $\mathrm{lmax}_s \,\mathrm{fill}(s)$ is minimized.
4. stop when $\max_s \mathrm{fill}(s) \approx \ln(n)/\varepsilon$
5. return $x/\max_s \mathrm{fill}(s)$

relaxed algorithm uses lmax instead of max

1. $x \leftarrow \mathbf{0}$ — *maximize* $\sum_i x_i$: $\max_s \text{fill}(s) \le 1$
2. repeat:
3. increase single x_i by ε,
   choosing i so increase in $\text{lmax}_s \text{fill}(s)$ is minimized.
   i.e., choose i to minimize $\sum_{s \ni i} e^{\text{fill}(s)}$
4. stop when $\max_s \text{fill}(s) \approx \ln(n)/\varepsilon$
5. return $x / \max_s \text{fill}(s)$

relaxed algorithm uses lmax instead of max

1. $x \leftarrow \mathbf{0}$ — *maximize* $\sum_i x_i \; : \; \max_s \mathrm{fill}(s) \leq 1$
2. repeat:
3. increase single x_i by ε,
   choosing i so increase in $\mathrm{lmax}_s \, \mathrm{fill}(s)$ is minimized.
   i.e., choose i to minimize $\sum_{s \ni i} e^{\mathrm{fill}(s)}$
4. stop when $\max_s \mathrm{fill}(s) \approx \ln(n)/\varepsilon$
5. return $x / \max_s \mathrm{fill}(s)$



relaxed algorithm

1. $x \leftarrow \mathbf{0}$ — *maximize* $\sum_i x_i$: $\max_s \text{fill}(s) \leq 1$
2. repeat:
3. increase single x_i by ε,
   choosing i so increase in $\text{lmax}_s \text{fill}(s)$ is minimized.
   i.e., choose i to minimize $\sum_{s \ni i} e^{\text{fill}(s)}$
4. stop when $\max_s \text{fill}(s) \approx \ln(n)/\varepsilon$
5. return $x / \max_s \text{fill}(s)$

Theorem (Garg and Könemann 1998)

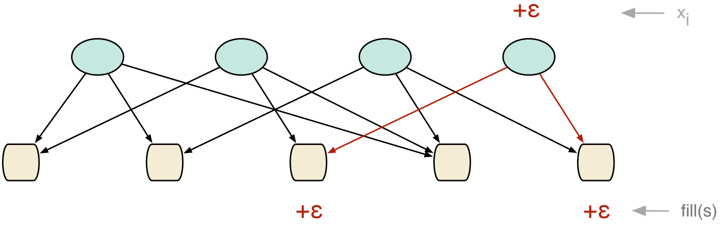
Returns $(1 - O(\varepsilon))$-approximate solution.

running time $O(n^3 \log(n)/\varepsilon^2)$

1. $x \leftarrow \mathbf{0}$ —*maximize* $\sum_i x_i$: $\max_s \text{fill}(s) \leq 1$
2. repeat:
3. increase single x_i by ε, choosing i so increase in $\text{lmax}_s\ \text{fill}(s)$ is minimized.
4. stop when $\max_s \text{fill}(s) \approx \ln(n)/\varepsilon$
5. return $x/\max_s \text{fill}(s)$

Theorem

Can maintain fill(s) *for all sets s in $O(n\log(n)/\varepsilon^2)$ total time.*



Each update to fill(s) takes $O(1)$ work; increases fill(s) by ε.
At most $\ln(n)/\varepsilon^2$ updates to fill(s) before fill(s) $= \ln(n)/\varepsilon$.

duality

dual of packing is covering:

given:	collection of sets
variables:	y_s for each set s (call vector y a *cover*)
objective:	minimize total weight $\sum_s y_s$
constraints:	$\mathrm{cov}(i) \geq 1$ for each element i, where $\mathrm{cov}(i) = \sum_{s \ni i} y_s$

strong duality:

For optimal packing x and cover y, $\sum_i x_i = \sum_s y_s$.

algorithm for covering

...just like algorithm for packing...

1. $y \leftarrow \mathbf{0}$ — *minimize* $\sum_s y_s$: $\min_i \operatorname{cov}(i) \geq 1$
2. repeat:
3. increase single y_s by ε,
   choosing s so increase in $\min_i \operatorname{cov}(i)$ is maximized.
4. delete elements i such that $\operatorname{cov}(i) \geq \ln(n)/\varepsilon$
5. stop when all elements deleted
6. return $y / \min_i \operatorname{cov}(i)$

Theorem (Garg and Könemann 1998)

Returns $(1 + O(\varepsilon))$-approximate solution.

coupling
from Grigoriadis and Khachiyan, 1995

vector x, function $f(x)$

$$f(x + \Delta x) - f(x) \approx \Delta x \cdot \nabla f(x)$$

vector y, function $g(y)$

$$g(y + \Delta y) - g(y) \approx \Delta y \cdot \nabla g(y)$$

coupling

from Grigoriadis and Khachiyan, 1995

vector x, function $f(x)$

$$f(x + \Delta x) - f(x) \approx \Delta x \cdot \nabla f(x)$$

vector y, function $g(y)$

$$g(y + \Delta y) - g(y) \approx \Delta y \cdot \nabla g(y)$$

Take $\Delta y = \nabla f(x)$ and $\Delta x = \nabla g(y)$...

coupling

from Grigoriadis and Khachiyan, 1995

vector x, function $f(x)$

$$f(x+\Delta x) - f(x) \approx \Delta x \cdot \nabla f(x) \approx \nabla g(y) \cdot \nabla f(x)$$

vector y, function $g(y)$

$$g(y+\Delta y) - g(y) \approx \Delta y \cdot \nabla g(y) \approx \nabla f(x) \cdot \nabla g(y)$$

Take $\Delta y = \nabla f(x)$ and $\Delta x = \nabla g(y)$...
then increase in f equals increase in g.

coupled algorithm

packing — maximize $\sum_i x_i$: $\max_s \mathrm{fill}(s) \leq 1$

1. $x \leftarrow \mathbf{0}$
2. repeat:
3. increase single x_i by ε to minimize increase in $\mathrm{lmax}_s\,\mathrm{fill}(s)$
4. stop when $\max_s \mathrm{fill}(s) \approx \ln(n)/\varepsilon$
5. return $x / \max_s \mathrm{fill}(s)$

covering — minimize $\sum_s y_s$: $\min_i \mathrm{cov}(i) \geq 1$

1. $y \leftarrow \mathbf{0}$
2. repeat:
3. increase single y_s by ε to maximize increase in $\mathrm{lmin}_i\,\mathrm{cov}(i)$
4. delete elements i such that $\mathrm{cov}(i) \geq \ln(n)/\varepsilon$
5. stop when all elements deleted
6. return $y / \min_i \mathrm{cov}(i)$

coupled algorithm

packing — *maximize* $\sum_i x_i \;:\; \max_s \mathrm{fill}(s) \le 1$

1. $x \leftarrow \mathbf{0}$
2. repeat:
3. ~~increase single x_i by ε to minimize increase in $\mathrm{lmax}_s\, \mathrm{fill}(s)$~~
3. $\quad x \leftarrow x + \varepsilon \nabla\, \mathrm{lmin}_i\, \mathrm{cov}(i)$
4. $\quad$ stop when $\max_s \mathrm{fill}(s) \approx \ln(n)/\varepsilon$
5. return $x / \max_s \mathrm{fill}(s)$

covering — *minimize* $\sum_s y_s \;:\; \min_i \mathrm{cov}(i) \ge 1$

1. $y \leftarrow \mathbf{0}$
2. repeat:
3. ~~increase single y_s by ε to maximize increase in $\mathrm{lmin}_i\, \mathrm{cov}(i)$~~
3. $\quad y \leftarrow y + \varepsilon \nabla\, \mathrm{lmax}_s\, \mathrm{fill}(s)$
4. $\quad$ delete elements i such that $\mathrm{cov}(i) \ge \ln(n)/\varepsilon$
5. $\quad$ stop when all elements deleted
6. return $y / \min_i \mathrm{cov}(i)$

coupled algorithm

1. $x \leftarrow \mathbf{0}$; $y \leftarrow \mathbf{0}$
2. repeat:
3. $\quad x \leftarrow x + \varepsilon \nabla \operatorname{lmin}_i \operatorname{cov}(i)$; $y \leftarrow y + \varepsilon \nabla \operatorname{lmax}_s \operatorname{fill}(s)$
4. $\quad$ delete elements i such that $\operatorname{cov}(i) \geq \ln(n)/\varepsilon$
5. $\quad$ stop when all elts deleted or $\max_s \operatorname{fill}(s) \approx \ln(n)/\varepsilon$
6. return $x/\max_s \operatorname{fill}(s)$ and $y/\min_i \operatorname{cov}(i)$

Theorem

Algorithm returns $(1 \pm \varepsilon)$*-approximate solutions.*

Proof.

Each iteration, both $\sum_i x_i$ and $\sum_s y_s$ increase by ε.
By coupling both $\operatorname{lmax}_s \operatorname{fill}(s)$ and $\operatorname{lmin}_i \operatorname{cov}(i)$ increase $\approx$ equally.
So at end,

$$\frac{\sum_i x_i}{\max_s \operatorname{fill}(s)} \approx \frac{\sum_s y_s}{\min_i \operatorname{cov}(i)}.$$

randomized algorithm

1. $x_i \leftarrow y_s \leftarrow 0$ for each element i and set s
2. repeat:
3. For one random i from distribution $\nabla \operatorname{lmin}_i \operatorname{cov}(i)$
4. and one random s from distribution $\nabla \operatorname{lmax}_s \operatorname{fill}(s)$:
5. $x_i \leftarrow x_i + \varepsilon$; $y_s \leftarrow y_s + \varepsilon$.
6. Delete elements i such that $\operatorname{cov}(i) \geq \ln(n)/\varepsilon$.
6. Stop when all elts deleted or $\max_s \operatorname{fill}(s) \approx \ln(n)/\varepsilon$.
7. return $x/\max_s \operatorname{fill}(s)$ and $y/\min_i \operatorname{cov}(i)$

Theorem (Koufogiannakis, Young 2007)

Algorithm returns $(1 \pm \varepsilon)$*-approximate solutions (in expectation).*

Proof.

$\operatorname{lmax}_s \operatorname{fill}(s)$ and $\operatorname{lmin}_i \operatorname{cov}(i)$ increase equally in expectation... □

randomized algorithm

1. $x_i \leftarrow y_s \leftarrow 0$ for each element i and set s
2. repeat:
3. For one random i from distribution $\nabla \operatorname{lmin}_i \operatorname{cov}(i) \propto e^{-\operatorname{cov}(i)}$
4. and one random s from distribution $\nabla \operatorname{lmax}_s \operatorname{fill}(s)$: $\propto e^{\operatorname{fill}(s)}$
5. $x_i \leftarrow x_i + \varepsilon$; $y_s \leftarrow y_s + \varepsilon$.
6. Delete elements i such that $\operatorname{cov}(i) \geq \ln(n)/\varepsilon$.
6. Stop when all elts deleted or $\max_s \operatorname{fill}(s) \approx \ln(n)/\varepsilon$.
7. return $x/\max_s \operatorname{fill}(s)$ and $y/\min_i \operatorname{cov}(i)$

Theorem (Koufogiannakis, Young 2007)

Algorithm returns $(1 \pm \varepsilon)$*-approximate solutions (in expectation).*

Proof.

$\operatorname{lmax}_s \operatorname{fill}(s)$ and $\operatorname{lmin}_i \operatorname{cov}(i)$ increase equally in expectation... □

Theorem

Randomized algorithm takes $O(n^2 + n\log(n)/\varepsilon^2)$ *time.*

Note: probability of choosing i is proportional to $\exp(-\operatorname{cov}(i))$;
probability of choosing s is proportional to $\exp(\operatorname{fill}(s))$.

simplex algorithm for linear programming
the competition

- Simplex invented by George Dantzig in 1947.
- Exponential time in worst case but "works well in practice".
- Takes typically *at least* $5n^3$ (n pivots, $5n^2$ each) basic operations, even for $\varepsilon = .05$.
- Worse on "ill-conditioned" matrices.

simplex algorithm for linear programming

the competition

- Simplex invented by George Dantzig in 1947.
- Exponential time in worst case but “works well in practice”.
- Takes typically *at least* $5n^3$ (n pivots, $5n^2$ each) basic operations, even for $\varepsilon = .05$.
- Worse on “ill-conditioned” matrices.

In comparison, our algorithm (first draft) finds $1 \pm \varepsilon$-approximate solutions guaranteed in about $5n^2 + 75n \ln n/\varepsilon^2$ basic operations.

Time for our algorithm / time for simplex is at most

$$\frac{1}{n} + \frac{\ln n}{(n\varepsilon/4)^2}.$$

E.g. when $\varepsilon = .01$, $\quad \dfrac{1}{n} + \dfrac{\ln n}{(n/400)^2}.$