CS 577 - Approximation

Marc Renault

Department of Computer Sciences
University of Wisconsin – Madison

Spring 2023

TopHat Section 001 Join Code: 020205
TopHat Section 002 Join Code: 394523

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Approximation Algorithms

Approximation Algorithms

Dealing with hard problems

Efficient Approximation Algorithms

- Polynomial run-time
- Guaranteed (worst-case) performance.
- Bi-criteria Goal: Be as efficient as possible and as close to optimal as possible.

Approximation Algorithms

Dealing with hard problems

Efficient Approximation Algorithms

- Polynomial run-time
- Guaranteed (worst-case) performance.
- Bi-criteria Goal: Be as efficient as possible and as close to optimal as possible.

Some Common Techniques

- Heuristics (esp. Greedy ones)

Approximation Algorithms

Dealing with hard problems

Efficient Approximation Algorithms

- Polynomial run-time
- Guaranteed (worst-case) performance.
- Bi-criteria Goal: Be as efficient as possible and as close to optimal as possible.

Some Common Techniques

- Heuristics (esp. Greedy ones)
- Linear Programming Relaxations (and SDP)

Approximation Algorithms

Dealing with hard problems

Efficient Approximation Algorithms

- Polynomial run-time
- Guaranteed (worst-case) performance.
- Bi-criteria Goal: Be as efficient as possible and as close to optimal as possible.

Some Common Techniques

- Heuristics (esp. Greedy ones)
- Linear Programming Relaxations (and SDP)
- Primal-Dual (related to LPs)

Approximation Algorithms

Dealing with hard problems

Efficient Approximation Algorithms

- Polynomial run-time
- Guaranteed (worst-case) performance.
- Bi-criteria Goal: Be as efficient as possible and as close to optimal as possible.

Some Common Techniques

- Heuristics (esp. Greedy ones)
- Linear Programming Relaxations (and SDP)
- Primal-Dual (related to LPs)
- Local Search

Approximation Algorithms

Dealing with hard problems

Efficient Approximation Algorithms

- Polynomial run-time
- Guaranteed (worst-case) performance.
- Bi-criteria Goal: Be as efficient as possible and as close to optimal as possible.

Some Common Techniques

- Heuristics (esp. Greedy ones)
- Linear Programming Relaxations (and SDP)
- Primal-Dual (related to LPs)
- Local Search
- Metric Embeddings

Approximation Algorithms

Dealing with hard problems

Efficient Approximation Algorithms

- Polynomial run-time
- Guaranteed (worst-case) performance.
- Bi-criteria Goal: Be as efficient as possible and as close to optimal as possible.

Some Common Techniques

- Heuristics (esp. Greedy ones)
- Linear Programming Relaxations (and SDP)
- Primal-Dual (related to LPs)
- Local Search
- Metric Embeddings
- Randomization

Approximation Ratio

Worst-Case Approximation Analysis

- Let $\textsc{alg}(I)$ be the value of an algorithm $\textsc{alg}$ for an instance I of problem P.
- r-approximation ratio:

$$\forall I, \textsc{alg}(I) \leq r \cdot \textsc{opt}(I) + \eta$$

- η is some constant (not dependant on the input)

Approximation Ratio

Worst-Case Approximation Analysis

- Let $\textsc{alg}(I)$ be the value of an algorithm $\textsc{alg}$ for an instance I of problem P.
- r-approximation ratio:

$$\forall I, \textsc{alg}(I) \leq r \cdot \textsc{opt}(I) + \eta$$

- η is some constant (not dependant on the input)
- $\eta = 0$ is a strict approximation ratio.
- $\eta > 0$ is a asymptotic approximation ratio.

Approximation Ratio

Worst-Case Approximation Analysis

- Let $\textsc{alg}(I)$ be the value of an algorithm $\textsc{alg}$ for an instance I of problem P.
- r-approximation ratio:

$$\forall I, \textsc{alg}(I) \leq r \cdot \textsc{opt}(I) + \eta$$

- η is some constant (not dependant on the input)

Minimization vs Maximization

- Min: Approximation ratio ≥ 1.
- Max: Approximation ratio ≤ 1, or we use $c \geq 1$ where $1/c = r \leq 1$.

Heuristics

Bin Packing Problem

Definition

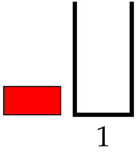
- Consists of:
  - an initial empty set of bins of capacity 1.
  - Finite request sequence: σ of length n.
- Each $r_i \in \sigma$ is an item with a size $s(r_i) \in (0, 1]$.
- Action: r_i must be packed in a bin without violating the capacity constraint.
- Goal: Minimize the number of bins used.

Bin Packing Problem

Definition

- Consists of:
  - an initial empty set of bins of capacity 1.
  - Finite request sequence: σ of length n.
- Each $r_i \in \sigma$ is an item with a size $s(r_i) \in (0, 1]$.
- Action: r_i must be packed in a bin without violating the capacity constraint.
- Goal: Minimize the number of bins used.

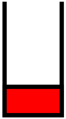
Bin Packing Problem



Definition

- Consists of:
  - an initial empty set of bins of capacity 1.
  - Finite request sequence: σ of length n.
- Each $r_i \in \sigma$ is an item with a size $s(r_i) \in (0, 1]$.
- Action: r_i must be packed in a bin without violating the capacity constraint.
- Goal: Minimize the number of bins used.

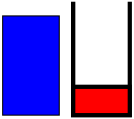
Bin Packing Problem



Definition

- Consists of:
    - an initial empty set of bins of capacity 1.
    - Finite request sequence: σ of length n.
- Each $r_i \in \sigma$ is an item with a size $s(r_i) \in (0, 1]$.
- Action: r_i must be packed in a bin without violating the capacity constraint.
- Goal: Minimize the number of bins used.

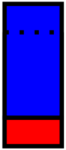
Bin Packing Problem



Definition

- Consists of:
  - an initial empty set of bins of capacity 1.
  - Finite request sequence: σ of length n.
- Each $r_i \in \sigma$ is an item with a size $s(r_i) \in (0, 1]$.
- Action: r_i must be packed in a bin without violating the capacity constraint.
- Goal: Minimize the number of bins used.

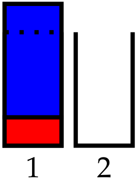
Bin Packing Problem



Definition

- Consists of:
    - an initial empty set of bins of capacity 1.
    - Finite request sequence: σ of length n.
- Each $r_i \in \sigma$ is an item with a size $s(r_i) \in (0, 1]$.
- Action: r_i must be packed in a bin without violating the capacity constraint.
- Goal: Minimize the number of bins used.

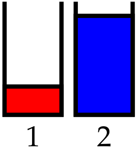
Bin Packing Problem



Definition

- Consists of:
  - an initial empty set of bins of capacity 1.
  - Finite request sequence: σ of length n.
- Each $r_i \in \sigma$ is an item with a size $s(r_i) \in (0, 1]$.
- Action: r_i must be packed in a bin without violating the capacity constraint.
- Goal: Minimize the number of bins used.

Bin Packing Problem



Definition

- Consists of:
  - an initial empty set of bins of capacity 1.
  - Finite request sequence: σ of length n.
- Each $r_i \in \sigma$ is an item with a size $s(r_i) \in (0, 1]$.
- Action: r_i must be packed in a bin without violating the capacity constraint.
- Goal: Minimize the number of bins used.

First Fit Heuristic

First Fit (FF)

- A greedy heuristic
- Bins are opened (and ordered) sequentially.
- Heuristic: Place the next item in the open bin of lowest index in which it fits, opening a new bin if needed.

First Fit Heuristic

First Fit (FF)

- A greedy heuristic
- Bins are opened (and ordered) sequentially.
- Heuristic: Place the next item in the open bin of lowest index in which it fits, opening a new bin if needed.

Tight Bounds

- Typically, approximation ratios are exact (no big-Oh).

First Fit Heuristic

First Fit (FF)

- A greedy heuristic
- Bins are opened (and ordered) sequentially.
- Heuristic: Place the next item in the open bin of lowest index in which it fits, opening a new bin if needed.

Tight Bounds

- Typically, approximation ratios are exact (no big-Oh).
- For an algorithm with an approx ratio of $r \geq 1$:
  - r_ℓ is a *lower bound* if $\exists$ an instance I where $\textsc{alg}(I) \geq r_\ell \cdot \textsc{opt}(I)$.
  - r_u is an *upper bound* if, $\forall$ instances I, $\textsc{alg}(I) \leq r_u \cdot \textsc{opt}(I)$.

First Fit Heuristic

First Fit (FF)

- A greedy heuristic
- Bins are opened (and ordered) sequentially.
- Heuristic: Place the next item in the open bin of lowest index in which it fits, opening a new bin if needed.

Tight Bounds

- Typically, approximation ratios are exact (no big-Oh).
- For an algorithm with an approx ratio of $r \geq 1$:
  - r_ℓ is a *lower bound* if $\exists$ an instance I where $\textsc{alg}(I) \geq r_\ell \cdot \textsc{opt}(I)$.
  - r_u is an *upper bound* if, $\forall$ instances I, $\textsc{alg}(I) \leq r_u \cdot \textsc{opt}(I)$.
- A tight bound is $r_\ell = r_u$.

FF Lower Bound

Theorem 1

FF *has an approximation ratio* $\geq 4/3$.

FF Lower Bound

Theorem 1

FF has an approximation ratio $\geq 4/3$.

Proof.

Consider a sequence of n items of size $\frac{1}{3} - 2\varepsilon$, followed by $2n$ items of size $\frac{1}{3} + \varepsilon$ (assume n divisible by 3):

FF Lower Bound

Theorem 1

FF has an approximation ratio $\geq 4/3$.

Proof.

Consider a sequence of n items of size $\frac{1}{3} - 2\varepsilon$, followed by $2n$ items of size $\frac{1}{3} + \varepsilon$ (assume n divisible by 3):

- FF packing: $\frac{n}{3}$ bins for the first n items, and n bins for the last $2n$ items. Overall $\frac{4}{3}n$ bins.

FF Lower Bound

Theorem 1

FF *has an approximation ratio* $\geq 4/3$.

Proof.

Consider a sequence of n items of size $\frac{1}{3} - 2\varepsilon$, followed by $2n$ items of size $\frac{1}{3} + \varepsilon$ (assume n divisible by 3):

- FF packing: $\frac{n}{3}$ bins for the first n items, and n bins for the last $2n$ items. Overall $\frac{4}{3}n$ bins.
- OPT packing: n bins each with 1 item of size $\frac{1}{3} - 2\varepsilon$, and 2 items of size $\frac{1}{3} + \varepsilon$.

□

FF Upper Bound

Observation 1

OPT $\geq N$, where N is the sum of the items.

FF Upper Bound

Observation 1

OPT $\geq N$, where N is the sum of the items.

Observation 2

In a packing S, if all bins are at least $0 < \alpha \leq 1$ full (except for possibly a constant number of bins), then the approximation ratio is α.

FF Upper Bound

Observation 1

OPT $\geq N$, where N is the sum of the items.

Observation 2

In a packing S, if all bins are at least $0 < \alpha \leq 1$ full (except for possibly a constant number of bins), then the approximation ratio is α.

Theorem 2

FF has an approximation ratio ≤ 2.

FF Upper Bound

Observation 1

OPT $\geq N$, where N is the sum of the items.

Observation 2

In a packing S, if all bins are at least $0 < \alpha \leq 1$ full (except for possibly a constant number of bins), then the approximation ratio is α.

Theorem 2

FF has an approximation ratio ≤ 2.

Proof.

All bins in FF are $\geq 1/2$ full (except possibly one bin). $\square$

FF Upper Bound

Theorem 2

FF has an approximation ratio ≤ 2.

Theorem 3

First fit decreasing (FFD) has an approximation ratio $\leq \frac{3}{2}$.

FF Upper Bound

Theorem 2

FF has an approximation ratio ≤ 2.

Theorem 3

First fit decreasing (FFD) *has an approximation ratio* $\leq \frac{3}{2}$.

Proof.

- Let k be the number of bins in a FFD packing.
- Let b_i be the last bin with an item of size $> \frac{1}{3}$.

FF Upper Bound

Theorem 2

FF has an approximation ratio ≤ 2.

Theorem 3

First fit decreasing (FFD) has an approximation ratio $\leq \frac{3}{2}$.

Proof.

- Let k be the number of bins in a FFD packing.
- Let b_i be the last bin with an item of size $> \frac{1}{3}$.
- If $i < k$, all bins are filled to at least $\frac{2}{3}$ (except possibly last bin).

□

FF Upper Bound

Theorem 2

FF has an approximation ratio ≤ 2.

Theorem 3

First fit decreasing (FFD) has an approximation ratio $\leq \frac{3}{2}$.

Proof.

- Let k be the number of bins in a FFD packing.
- Let b_i be the last bin with an item of size $> \frac{1}{3}$.
- If $i < k$, all bins are filled to at least $\frac{2}{3}$ (except possibly last bin).
- If $i = k$, there are at least k items of size $> \frac{1}{3}$. Since these k items can fit at most 2 to a bin, OPT $\geq \frac{2}{3}k$.

□

FF Upper Bound

Theorem 2

FF has an approximation ratio ≤ 2.

Theorem 3

First fit decreasing (FFD) *has an approximation ratio* $\leq \frac{3}{2}$.

Known Bounds

- FF has an approximation ratio of 1.7.
- FFD has an approximation ratio of 11/9.

PTAS

Arbitrarily Good Approximations

Polynomial Time Approximation Scheme (PTAS)

For some ε:

- An arbitrarily good approximation guarantee that is a function of ε.
- A run-time that is polynomial in the input for a fixed ε.

Arbitrarily Good Approximations

Polynomial Time Approximation Scheme (PTAS)

For some ε:

- An arbitrarily good approximation guarantee that is a function of ε.
- A run-time that is polynomial in the input for a fixed ε.
- Run-time vs approximation ratio trade-off.

Arbitrarily Good Approximations

Polynomial Time Approximation Scheme (PTAS)

For some ε:

- An arbitrarily good approximation guarantee that is a function of ε.
- A run-time that is polynomial in the input for a fixed ε.
- Run-time vs approximation ratio trade-off.

Inapproximability Results

- Not all problems admit a PTAS.

Arbitrarily Good Approximations

Polynomial Time Approximation Scheme (PTAS)

For some ε:

- An arbitrarily good approximation guarantee that is a function of ε.
- A run-time that is polynomial in the input for a fixed ε.
- Run-time vs approximation ratio trade-off.

Inapproximability Results

- Not all problems admit a PTAS.
- Some problems have inapproximability results:
  - Unless $\mathrm{NP} = \mathrm{P}$, Vertex Cover cannot be approximated within a factor of 1.3606.

Arbitrarily Good Approximations

Polynomial Time Approximation Scheme (PTAS)

For some ε:

- An arbitrarily good approximation guarantee that is a function of ε.
- A run-time that is polynomial in the input for a fixed ε.
- Run-time vs approximation ratio trade-off.

Inapproximability Results

- Not all problems admit a PTAS.
- Some problems have inapproximability results:
  - Unless $\mathrm{NP} = \mathrm{P}$, Vertex Cover cannot be approximated within a factor of 1.3606.
  - Unless $\mathrm{NP} = \mathrm{P}$, Bin Packing cannot be strictly approximated within a factor of $3/2$.

Bin Packing Brute Force

Bin Types

- Let k be the number of different item sizes.

Bin Packing Brute Force

Bin Types

- Let k be the number of different item sizes.
- Let ε be the smallest item size:
  - at most $M = \lfloor 1/\varepsilon \rfloor$ items in a bin.

Bin Packing Brute Force

Bin Types

- Let k be the number of different item sizes.
- Let ε be the smallest item size:
  - at most $M = \lfloor 1/\varepsilon \rfloor$ items in a bin.
- $R = \left(\!\binom{M}{k+1}\!\right) = \binom{M+K}{M}$ bin types:
  - Choose M items to pack in a bin from $k+1$ different items sizes (includes 0).

Bin Packing Brute Force

Bin Types

- Let k be the number of different item sizes.
- Let ε be the smallest item size:
  - at most $M = \lfloor 1/\varepsilon \rfloor$ items in a bin.
- $R = \left(\!\binom{M}{k+1}\!\right) = \binom{M+K}{M}$ bin types:
  - Choose M items to pack in a bin from $k+1$ different items sizes (includes 0).

Number of Packings

- At most n bins are need, and we have R types of bins, so there are:

$$P = \left(\!\!\binom{n}{R+1}\!\!\right) = \binom{n+R}{R}$$

feasible packings to check.

Bin Packing Brute Force

Bin Types

- Let k be the number of different item sizes.
- Let ε be the smallest item size:
  - at most $M = \lfloor 1/\varepsilon \rfloor$ items in a bin.
- $R = \left(\!\binom{M}{k+1}\!\right) = \binom{M+K}{M}$ bin types:
  - Choose M items to pack in a bin from $k+1$ different items sizes (includes 0).

Number of Packings

- At most n bins are need, and we have R types of bins, so there are:

$$P = \left(\!\!\binom{n}{R+1}\!\!\right) = \binom{n+R}{R} \leq (n+R)^R = O(n^R)$$

feasible packings to check.

Bin Packing (asymptotic) PTAS

Theorem 4

For any $\varepsilon > 0$*, there is an* $\textsc{alg}_\varepsilon$ *that runs in* $\mathrm{poly}(n)$ *time for which* $\textsc{alg}_\varepsilon(I) \leq (1 + 2\varepsilon)\textsc{opt}(I) + 1$ *for all* I.

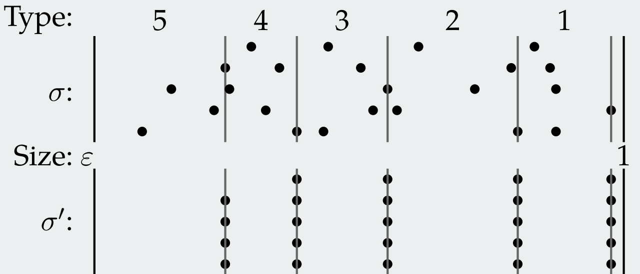
Bin Packing (asymptotic) PTAS

Theorem 4

For any $\varepsilon > 0$, there is an $\textsc{alg}_\varepsilon$ *that runs in* $\mathrm{poly}(n)$ *time for which* $\textsc{alg}_\varepsilon(I) \le (1 + 2\varepsilon)\textsc{opt}(I) + 1$ *for all I.*

Proof (PTAS).

- Split the items of size $\geq \varepsilon$ (denoted by L) into groups containing $\lfloor n\varepsilon^2 \rfloor$ items $\implies$ $\lceil 1/\varepsilon^2 \rceil$ groups.
- Round items up to size of largest item:



Bin Packing (asymptotic) PTAS

Theorem 4

For any $\varepsilon > 0$, there is an ALG_ε *that runs in* $\text{poly}(n)$ *time for which* $\text{ALG}_\varepsilon(I) \leq (1 + 2\varepsilon)\text{OPT}(I) + 1$ *for all I.*

Proof (PTAS).

- Split the items of size $\geq \varepsilon$ (denoted by L) into groups containing $\lfloor n\varepsilon^2 \rfloor$ items $\implies \lceil 1/\varepsilon^2 \rceil$ groups.
- Round items up to size of largest item.
- Pack rounded items: $O(n^{f(\frac{1}{\varepsilon})}) \implies O(n^{\frac{2}{\varepsilon^2}})$.

Bin Packing (asymptotic) PTAS

Theorem 4

For any $\varepsilon > 0$, there is an $\textsc{alg}_\varepsilon$ *that runs in* $\mathrm{poly}(n)$ *time for which* $\textsc{alg}_\varepsilon(I) \leq (1+2\varepsilon)\textsc{opt}(I) + 1$ *for all* I.

Proof (PTAS).

- Split the items of size $\geq \varepsilon$ (denoted by L) into groups containing $\lfloor n\varepsilon^2 \rfloor$ items $\implies \lceil 1/\varepsilon^2 \rceil$ groups.
- Round items up to size of largest item.
- Pack rounded items: $O(n^{f(\frac{1}{\varepsilon})}) \implies O(n^{\frac{2}{\varepsilon^2}})$.
- Pack items $< \varepsilon$ using FF.

Bin Packing (asymptotic) PTAS

Theorem 4

For any $\varepsilon > 0$, there is an $\textsc{alg}_\varepsilon$ *that runs in* $\mathrm{poly}(n)$ *time for which* $\textsc{alg}_\varepsilon(I) \leq (1+2\varepsilon)\textsc{opt}(I) + 1$ *for all I.*

Proof (PTAS).

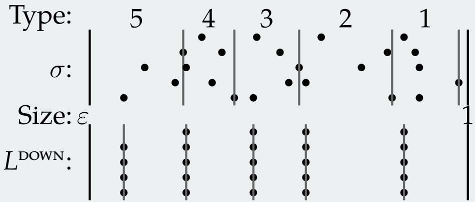
- Approximation ratio:
  - $\textsc{opt}(L) \geq \varepsilon|L| \geq n\varepsilon$.

Bin Packing (asymptotic) PTAS

Proof (PTAS).

- Approximation ratio:
  - if items $< \varepsilon$ fit in $\textsc{alg}_\varepsilon(L)$ bins:

$$\textsc{alg}_\varepsilon(I) = \textsc{alg}_\varepsilon(L) \leq \textsc{alg}_\varepsilon(L^{\textsc{down}}) + n\varepsilon^2$$



  - Let S'' be optimal packing of $L^{\textsc{down}}$ ($\textsc{alg}_\varepsilon(L^{\textsc{down}}) \leq \textsc{opt}(L)$).
  - We can pack σ using S'':
    (1) Group 1 items in bins by themselves ($\leq n\varepsilon^2$ bins at most).
    (2) For $\lceil 1/\varepsilon^2 \rceil \geq i > 1$, pack the items of group i in the spots for the items of group $i-1$ in S''.

Bin Packing (asymptotic) PTAS

Theorem 4

For any $\varepsilon > 0$, there is an $\textsc{alg}_\varepsilon$ *that runs in* $\mathrm{poly}(n)$ *time for which* $\textsc{alg}_\varepsilon(I) \le (1+2\varepsilon)\textsc{opt}(I) + 1$ *for all* I.

Proof (PTAS).

- Approximation ratio:
  - $\textsc{opt}(L) \ge \varepsilon|L| \ge n\varepsilon$.
  - if items $< \varepsilon$ fit in $\textsc{alg}_\varepsilon(L)$ bins:

$$\begin{aligned}\textsc{alg}_\varepsilon(I) = \textsc{alg}_\varepsilon(L) &\le \textsc{alg}_\varepsilon(L^{\textsc{down}}) + n\varepsilon^2 \\ &\le \textsc{opt}(L) + \varepsilon\textsc{opt}(L) \\ &= (1+\varepsilon)\textsc{opt}(L)\end{aligned}$$

Bin Packing (asymptotic) PTAS

Theorem 4

For any $\varepsilon > 0$, there is an $\textsc{alg}_\varepsilon$ *that runs in* $\mathrm{poly}(n)$ *time for which* $\textsc{alg}_\varepsilon(I) \leq (1 + 2\varepsilon)\textsc{opt}(I) + 1$ *for all* I.

Proof (PTAS).

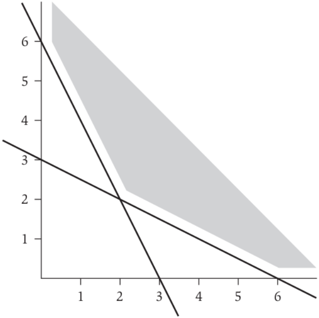
- Approximation ratio:
  - if items $< \varepsilon$ open new bins:
    - All bins (except possibly last one) are $(1 - \varepsilon)$ full:

$$\begin{aligned} \textsc{opt}(I) &\geq (\textsc{alg}_\varepsilon(I) - 1)(1 - \varepsilon) \\ \iff \textsc{alg}_\varepsilon(I) &\leq (1 + 2\varepsilon)\textsc{opt}(I) + 1 \end{aligned}$$

$\square$

Linear Programming

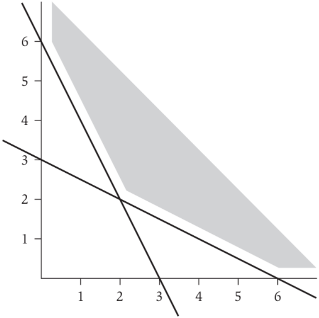
Linear Programming



Basic Linear Program

$$
\begin{array}{rlcl}
\min & 1.5x_1 + x_2 & & \\
\text{s.t.} & x_1 + 2x_2 & \geq & 6 \\
& 2x_1 + x_2 & \geq & 6 \\
& x_1 \geq 0, x_2 \geq 0 & &
\end{array}
$$

Linear Programming

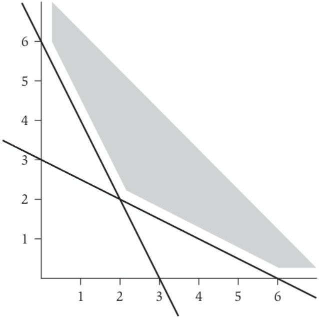


Basic Linear Program

$$
\begin{array}{rlcl}
\min & 1.5x_1 + x_2 & & \\
\text{s.t.} & x_1 + 2x_2 & \geq & 6 \\
 & 2x_1 + x_2 & \geq & 6 \\
 & x_1 \geq 0, x_2 \geq 0 & &
\end{array}
$$

TH1: What is the solution vector (x_1, x_2)?

Linear Programming



Basic Linear Program

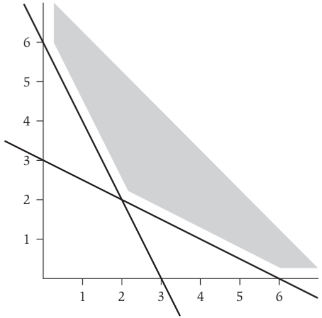
$$\begin{array}{rlll} \min & 1.5x_1 + x_2 & & \\ \text{s.t.} & x_1 + 2x_2 & \geq & 6 \\ & 2x_1 + x_2 & \geq & 6 \\ & x_1 \geq 0, x_2 \geq 0 & & \end{array}$$

General Linear Program

- Optimize a linear object function subject to linear constraints.

$$\begin{array}{rllll} \text{optimize}_x & c^T x & & & \\ \text{subject to} & Ax & \leq & b & \\ & x_j & \geq & 0 & \forall j \end{array}$$

Linear Programming



Basic Linear Program

$$c^T = \begin{bmatrix} 1.5 & 1 \end{bmatrix} \quad A = \begin{bmatrix} -1 & -2 \\ -2 & -1 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad b = \begin{bmatrix} -6 \\ -6 \end{bmatrix}$$

General Linear Program

- Optimize a linear object function subject to linear constraints.

$$\begin{array}{rlll} \text{optimize}_x & c^T x & & \\ \text{subject to} & Ax & \leq & b \\ & x_j & \geq & 0 \qquad \forall j \end{array}$$

Linear Programming (LP)

Some Notes

- In 1947, Dantzig developed general linear programming.

Linear Programming (LP)

Some Notes

- In 1947, Dantzig developed general linear programming.
- Khachiyan 1979: Ellipsoid Method, polynomial algorithm to solve LPs.

Linear Programming (LP)

Some Notes

- In 1947, Dantzig developed general linear programming.
- Khachiyan 1979: Ellipsoid Method, polynomial algorithm to solve LPs.
- In practice:

Linear Programming (LP)

Some Notes

- In 1947, Dantzig developed general linear programming.
- Khachiyan 1979: Ellipsoid Method, polynomial algorithm to solve LPs.
- In practice:
  - Interior Point Method (poly-time 1984 Karmarkar)

Linear Programming (LP)

Some Notes

- In 1947, Dantzig developed general linear programming.
- Khachiyan 1979: Ellipsoid Method, polynomial algorithm to solve LPs.
- In practice:
  - Interior Point Method (poly-time 1984 Karmarkar)
  - Simplex (exp-time, usually poly-time 1947 Dantzig)

Linear Programming (LP)

Some Notes

- In 1947, Dantzig developed general linear programming.
- Khachiyan 1979: Ellipsoid Method, polynomial algorithm to solve LPs.
- In practice:
  - Interior Point Method (poly-time 1984 Karmarkar)
  - Simplex (exp-time, usually poly-time 1947 Dantzig)
- BUT: Integer Linear Programming is NP-complete (one of Karp's 21 NP-complete problems)

Linear Programming (LP)

Some Notes

- In 1947, Dantzig developed general linear programming.
- Khachiyan 1979: Ellipsoid Method, polynomial algorithm to solve LPs.
- In practice:
  - Interior Point Method (poly-time 1984 Karmarkar)
  - Simplex (exp-time, usually poly-time 1947 Dantzig)
- BUT: Integer Linear Programming is NP-complete (one of Karp's 21 NP-complete problems)
  - ILP requires all x to be integers.

Linear Programming (LP)

Some Notes

- In 1947, Dantzig developed general linear programming.
- Khachiyan 1979: Ellipsoid Method, polynomial algorithm to solve LPs.
- In practice:
  - Interior Point Method (poly-time 1984 Karmarkar)
  - Simplex (exp-time, usually poly-time 1947 Dantzig)
- BUT: Integer Linear Programming is NP-complete (one of Karp's 21 NP-complete problems)
  - ILP requires all x to be integers.
  - Mixed ILP is also NP-complete.

Vertex Cover

ILP

$$
\begin{array}{llll}
\min\limits_{x} & \sum_{i \in V} w_i x_i & & \\
\text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\
& x_i \in & \{0,1\} & \forall i \in V
\end{array}
$$

Weighted Vertex Cover

- A graph $G = (V, E)$.
- For each $v_i : w_i \geq 0$.
- Goal: Find the minimum weighted vertex cover.

Vertex Cover

ILP

$$\begin{array}{llll} \min\limits_{x} & \sum_{i \in V} w_i x_i & & \\ \text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\ & x_i \in & \{0,1\} & \forall i \in V \end{array}$$

$\rightarrow$

Relaxed LP

$$\begin{array}{llll} \min\limits_{x} & \sum_{i \in V} w_i x_i & & \\ \text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\ & x_i \in & [0,1] & \forall i \in V \end{array}$$

Weighted Vertex Cover

- A graph $G = (V, E)$.
- For each $v_i : w_i \geq 0$.
- Goal: Find the minimum weighted vertex cover.

Vertex Cover LP

<u>Relaxed LP</u>

$$\begin{array}{llll} \min\limits_{x} & \sum_{i \in V} w_i x_i & & \\ \text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\ & x_i \in & [0,1] & \forall i \in V \end{array}$$

What does a solution mean?

- Say $x_i = 0.42$ in a solution to the LP:
  - TH2: What is the minimum value for x_j if $(i,j) \in E$?

Vertex Cover LP

<u>Relaxed LP</u>

$$\begin{array}{rlcll} \min\limits_{x} & \sum_{i \in V} w_i x_i & & & \\ \text{s.t.} & x_i + x_j & \geq & 1 & \forall (i,j) \in E \\ & x_i & \in & [0,1] & \forall i \in V \end{array}$$

What does a solution mean?

- Say $x_i = 0.42$ in a solution to the LP:
  - $x_j \geq 0.58$ if $(i,j) \in E$

Vertex Cover LP

Relaxed LP

$$\begin{array}{llll} \min\limits_{x} & \sum_{i\in V} w_i x_i & & \\ \text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\ & x_i \in & [0,1] & \forall i \in V \end{array}$$

What does a solution mean?

- Say $x_i = 0.42$ in a solution to the LP:
  - $x_j \geq 0.58$ if $(i,j) \in E$
- It means we have 0.42 of i and ≥ 0.58 of j.

Vertex Cover LP

Relaxed LP

$$\begin{array}{llll} \min_x & \sum_{i \in V} w_i x_i & & \\ \text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\ & x_i \in & [0,1] & \forall i \in V \end{array}$$

What does a solution mean?

- Say $x_i = 0.42$ in a solution to the LP:
  - $x_j \geq 0.58$ if $(i,j) \in E$
- It means we have 0.42 of i and ≥ 0.58 of j.

Rounding Fractional Solution

Some methods:

- Fixed rounding rule
- Treat it as probability and do a probabilistic rounding.

Vertex Cover LP

Relaxed LP

$$\begin{array}{llll} \min\limits_{x} & \sum_{i \in V} w_i x_i & & \\ \text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\ & x_i \in & [0,1] & \forall i \in V \end{array}$$

What does a solution mean?

- Say $x_i = 0.42$ in a solution to the LP:
  - $x_j \geq 0.58$ if $(i,j) \in E$
- It means we have 0.42 of i and ≥ 0.58 of j.

Vertex Cover Rounding

- Round up to 1 if $x_i \geq 1/2$, else down to 0.

Vertex Cover LP

Relaxed LP

$$
\begin{array}{llll}
\min\limits_x & \sum_{i \in V} w_i x_i & & \\
\text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\
& x_i \in & [0,1] & \forall i \in V
\end{array}
$$

What does a solution mean?

- Say $x_i = 0.42$ in a solution to the LP:
    - $x_j \geq 0.58$ if $(i,j) \in E$
- It means we have 0.42 of i and ≥ 0.58 of j.

Vertex Cover Rounding

- Round up to 1 if $x_i \geq 1/2$, else down to 0.
- Let S be the rounded solution, and S^* be the optimal ILP solution.

Vertex Cover LP

Relaxed LP

$$
\begin{array}{llll}
\min\limits_{x} & \sum_{i \in V} w_i x_i & & \\
\text{s.t.} & x_i + x_j & \geq & 1 & \forall (i,j) \in E \\
& x_i & \in & [0,1] & \forall i \in V
\end{array}
$$

What does a solution mean?

- Say $x_i = 0.42$ in a solution to the LP:
  - $x_j \geq 0.58$ if $(i,j) \in E$
- It means we have 0.42 of i and ≥ 0.58 of j.

Vertex Cover Rounding

- Round up to 1 if $x_i \geq 1/2$, else down to 0.
- Let S be the rounded solution, and S^* be the optimal ILP solution.
- S is a VC since, $\forall (i,j) \in E$, x_i or $x_j > 1/2 \implies i$ or $j \in S$.

Vertex Cover LP

Relaxed LP

$$\begin{array}{llll} \min_x & \sum_{i \in V} w_i x_i & & \\ \text{s.t.} & x_i + x_j \geq & 1 & \forall (i,j) \in E \\ & x_i \in & [0,1] & \forall i \in V \end{array}$$

What does a solution mean?

- Say $x_i = 0.42$ in a solution to the LP:
  - $x_j \geq 0.58$ if $(i,j) \in E$
- It means we have 0.42 of i and ≥ 0.58 of j.

Vertex Cover Rounding

- Round up to 1 if $x_i \geq 1/2$, else down to 0.
- Let S be the rounded solution, and S^* be the optimal ILP solution.
- S is a VC since, $\forall (i,j) \in E$, x_i or $x_j > 1/2 \implies i$ or $j \in S$.

$$w(S) = \sum_{i \in S} w_i \leq 2 \sum_{i \in S} w_i x_i \leq 2 \sum_i w_i x_i \leq 2w(S^*)$$

Appendix

References

Image Sources I

https://brand.wisc.edu/web/logos/