Design and Analysis of Algorithms

Approximation Algorithms – Part 6

Pan Peng

School of Computer Science and Technology
University of Science and Technology of China

Outline

- Recap of approximation algorithms
- Hardness of approximation
- Some examples
  - k-center
  - bin packing
  - minimum vertex cover

Recap of Approximation Algorithms

Recall: Why approximation algorithms?

For many decision or optimization problems, we do not know how to solve them exactly in polynomial time

Recall: Why approximation algorithms?

For many decision or optimization problems, we do not know how to solve them exactly in polynomial time

Researchers introduced the computational complexity classes **P**, **NP**, **NP**-complete, **NP**-hard, etc.

Recall: Why approximation algorithms?

For many decision or optimization problems, we do not know how to solve them exactly in polynomial time

Researchers introduced the computational complexity classes **P**, **NP**, **NP**-complete, **NP**-hard, etc.

We now know that many problems are **NP**-complete, or **NP**-hard.

Recall: Why approximation algorithms?

For many decision or optimization problems, we do not know how to solve them exactly in polynomial time

Researchers introduced the computational complexity classes **P**, **NP**, **NP**-complete, **NP**-hard, etc.

We now know that many problems are **NP**-complete, or **NP**-hard.

A (trivial) Theorem

Assuming $\mathbf{P} \neq \mathbf{NP}$, there is no polynomial time algorithm for finding an optimal solution of any **NP**-hard problem Π.

Such problems Π include:

- minimum vertex cover, minimum dominate set, sparsest cut, etc.
- maximum independent set, maximum clique, maximum cut, etc.

Recall: Why approximation algorithms?

For many decision or optimization problems, we do not know how to solve them exactly in polynomial time

Researchers introduced the computational complexity classes **P**, **NP**, **NP**-complete, **NP**-hard, etc.

We now know that many problems are **NP**-complete, or **NP**-hard.

A (trivial) Theorem

Assuming $\mathbf{P} \neq \mathbf{NP}$, there is no polynomial time algorithm for finding an optimal solution of any **NP**-hard problem Π.

Such problems Π include:

- minimum vertex cover, minimum dominate set, sparsest cut, etc.
- maximum independent set, maximum clique, maximum cut, etc.

Approximation algorithms: trade accuracy for time!

Basic definitions

Minimization problem: For $\rho(n) \geq 1$, an algorithm $\mathcal{A}$ is called a $\rho(n)$-approximation algorithm if, for any instance I,

- $\mathcal{A}$ runs in polynomial-time in the input size n, and
- $\mathcal{A}$ computes a solution with objective function value

$$\mathrm{OPT}(I) \leq \mathcal{A}(I) \leq \rho(n) \cdot \mathrm{OPT}(I).$$

- $\rho(n)$ is the approximation ratio or performance guarantee,
- $\rho(n)$ can depend on the input size n or be constant,
- 1-approximation algorithms are exact.

Basic definitions II

Maximization problem: For $\rho(n) \leq 1$, an algorithm $\mathcal{A}$ is called a $\rho(n)$-approximation algorithm if, for any instance I,

- $\mathcal{A}$ runs in polynomial-time in the input size n, and
- $\mathcal{A}$ computes a solution with objective function value

$$\mathrm{OPT}(I) \geq \mathcal{A}(I) \geq \rho(n) \cdot \mathrm{OPT}(I).$$

- $\rho(n)$ is the approximation ratio or performance guarantee,
- $\rho(n)$ can depend on the input size n or be constant,
- 1-approximation algorithms are exact.

- Remark: Sometimes, people use $\frac{1}{\rho(n)}$ (≥ 1) as the approximation ratio for a maximization problem.

Some Facts

Approaches for designing approximation algorithms

- greedy
  - knapsack, vertex cover, set cover, k-center, metric TSP, correlation clustering (minimization)
- rounding data and dynamic programming
  - knapsack
- LP+rounding
  - weighted vertex cover
- random sampling
  - Max 2-SAT
- local search
  - max cut, k-median, k-means
- LP+primal dual
  - uncapacitated facility location (UFL)
- SDP + rounding
  - max cut, correlation clustering (maximization)
- LP + metric embedding
  - sparsest cut

One notion: polynomial time approximation scheme (PTAS)

Given an optimization instance and a value $\varepsilon > 0$, produce an answer in polynomial time that is within a factor $(1 \pm \varepsilon)$ of optimal

- We have just seen a PTAS for the knapsack problem

What complexities are allowed?

- $O(n)$, $O(n^{2.45/\varepsilon})$, $O(n^{e^{e^{1/\varepsilon}}})$
- polynomial in n for each fixed ε
- the order is allowed to be different for different choices of ε

Some representative results in approximation algorithms

Problems with PTAS (i.e., $(1+\varepsilon)$-approx. for minimization or $(1-\varepsilon)$-approx. for maximization problem):

- knapsack problem, correlation clustering (max.), Euclidean TSP, ...

Some representative results in approximation algorithms

Problems with PTAS (i.e., $(1+\varepsilon)$-approx. for minimization or $(1-\varepsilon)$-approx. for maximization problem):

- knapsack problem, correlation clustering (max.), Euclidean TSP, ...

Problems with $O(1)$-approx. (min.) or $\Omega(1)$-approx. (max.):

- min. vertex cover, metric TSP, max cut, Max 2SAT, k-center, k-median, k-means, correlation clustering (minimization), UFL...

Some representative results in approximation algorithms

Problems with PTAS (i.e., $(1+\varepsilon)$-approx. for minimization or $(1-\varepsilon)$-approx. for maximization problem):

- knapsack problem, correlation clustering (max.), Euclidean TSP, ...

Problems with $O(1)$-approx. (min.) or $\Omega(1)$-approx. (max.):

- min. vertex cover, metric TSP, max cut, Max 2SAT, k-center, k-median, k-means, correlation clustering (minimization), UFL...

Problems with $O(\log n)$-approx. (min.) or $\Omega(1/\log n)$-approx. (max.):

- minimum set cover, minimum dominating set, sparsest cut...

Some representative results in approximation algorithms

Problems with PTAS (i.e., $(1+\varepsilon)$-approx. for minimization or $(1-\varepsilon)$-approx. for maximization problem):

- knapsack problem, correlation clustering (max.), Euclidean TSP, ...

Problems with $O(1)$-approx. (min.) or $\Omega(1)$-approx. (max.):

- min. vertex cover, metric TSP, max cut, Max 2SAT, k-center, k-median, k-means, correlation clustering (minimization), UFL...

Problems with $O(\log n)$-approx. (min.) or $\Omega(1/\log n)$-approx. (max.):

- minimum set cover, minimum dominating set, sparsest cut...

Problems for which even $O(n^{0.99})$-approx. (minimization) or $\Omega(n^{-0.99})$-approx. (maximization) were unknown:

- minimum coloring, maximum independent set, max clique, ...

Some representative results

Problems with PTAS (i.e., $(1+\varepsilon)$-approx. for minimization or $(1-\varepsilon)$-approx. for maximization problem):

- knapsack problem, correlation clustering (max.), Euclidean TSP, ...

Problems with $O(1)$-approx. (min.) or $\Omega(1)$-approx. (max.):

- min. vertex cover, metric TSP, max cut, Max 2SAT, k-center, k-median, k-means, correlation clustering (minimization), UFL...

Problems with $O(\log n)$-approx. (min.) or $\Omega(1/\log n)$-approx. (max.):

- minimum set cover, minimum dominating set, sparsest cut...

Problems for which even $O(n^{0.99})$-approx. (minimization) or $\Omega(n^{-0.99})$-approx. (maximization) were unknown:

- minimum coloring, maximum independent set, max clique, ...

Given a minimization problem Π, maybe it is impossible to obtain an approximation algorithm for Π with approximation ratio $O(1)$, $O(\log n)$ or even $n^{0.99}$?
(The case of maximization problems is similar.)

Hardness of Approximation

Hardness of Approximation

Idea: Even if we only aim for near-optimal (i.e., approximation) solutions, it is still hard!

Hardness of Approximation

Idea: Even if we only aim for near-optimal (i.e., approximation) solutions, it is still hard!

Hardness of Approximation

the computational complexity of finding near-optimal solutions to optimization problems.

Hardness of Approximation

Idea: Even if we only aim for near-optimal (i.e., approximation) solutions, it is still hard!

Hardness of Approximation

the computational complexity of finding near-optimal solutions to optimization problems.

Typical theorem for an NP-hard problem Π

Assuming **P** $\neq$ **NP**, there is no polynomial time algorithm for finding an $\alpha^*(\Pi)$-approximation solution of the problem Π.

Approaches for proving hardness of approximation of **NP**-hard optimization problems

- direct gap reductions from NP-complete (decision) problems
- indirect gap reductions (e.g., by PCP theorem, by approximation-preserving reductions)

More on direct gap reductions

Let L be an **NP**-complete decision problem/language and Π be an NP-hard optimization problem.

Assume that Π is a minimization problem. (The case of maximization can be defined analogously.)

Then a (f, c_1, c_2) gap reduction from L to Π is the following:

1. f maps instances (or strings) of L to instances of Π
2. c_1, c_2 are functions from $\mathbb{Z}^+$ to $\mathbb{Q}^+$
3. f, c_1, c_2 are polynomial time computable functions
4. if $x \in L$ then $\mathrm{OPT}(f(x)) \leq c_1(|f(x)|)$
5. if $x \notin L$ then $\mathrm{OPT}(f(x)) > c_2(|f(x)|)$

More on direct gap reductions

What is the use of (f, c_1, c_2) reduction?

We can say the following: If there is such a reduction from L to Π then unless **P**$=$**NP** there is no $c_2(n)/c_1(n)$ approximation for Π, where n is the size of input for Π

Proof: Exercise

Examples

We will see such gap reductions

- for k-center problem, we give a reduction from the dominating set problem to the k-center problem with $c_1(n) = 1$ and $c_2(n) = 2 - \varepsilon$ for any fixed $\varepsilon > 0$
- for bin-packing problem, we give a reduction from the partition problem with $c_1(n) = 2$ and $c_2(n) = 3 - \varepsilon$ for any fixed $\varepsilon > 0$

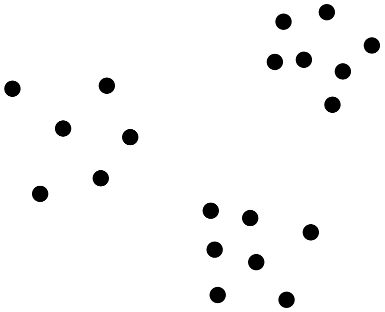
Example: k-center problem

k-center problem

Input: an undirected complete graph $G = (V, E)$, with distances $d_{ij} \geq 0 \ \forall i, j$ such that $d_{ii} = 0$, $d_{ij} = d_{ji}$, and $d_{ij} \leq d_{i\ell} + d_{j\ell}$, an integer k

Objective: find a set S with $|S| = k$ such that the cost of S, i.e., $\mathrm{cost}(S) := \max_{i \in V} \min_{s \in S} d_{is}$, is minimized.

(**) every vertex (or point) in a cluster is in distance at most $\mathrm{cost}(S)$ from its respective center.

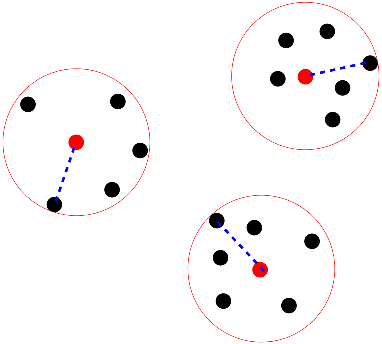


k-center problem

Input: an undirected complete graph $G = (V, E)$, with distances $d_{ij} \geq 0 \ \forall i, j$ such that $d_{ii} = 0$, $d_{ij} = d_{ji}$, and $d_{ij} \leq d_{i\ell} + d_{j\ell}$, an integer k

Objective: find a set S with $|S| = k$ such that the cost of S, i.e., $\text{cost}(S) := \max_{i \in V} \min_{s \in S} d_{is}$, is minimized.

(**) every vertex (or point) in a cluster is in distance at most $\text{cost}(S)$ from its respective center.



A hardness result for approximating k-center

We know that:

Theorem: There is a 2-approximation algorithm for the k-center problem.

A hardness result for approximating k-center

We know that:

Theorem: There is a 2-approximation algorithm for the k-center problem.

We can also show that:

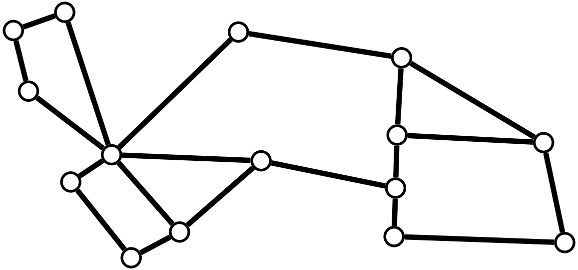
Theorem 1: Assuming that **P** $\neq$ **NP**, there is no $(2-\varepsilon)$-approximation algorithm for the k-center problem for any $\varepsilon > 0$.

Proof of Theorem 1

Idea: Reduction from dominating set problem (DSP)

DSP

Input: an undirected graph $G = (V, E)$, an integer k
Objective: decide if there exists a set $S \subseteq V$ with $|S| = k$ such that each vertex is either in S or adjacent to a vertex in S.

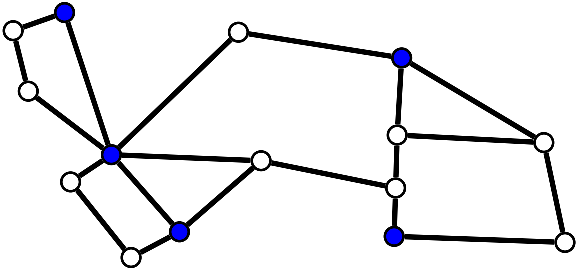


Proof of Theorem 1

Idea: Reduction from dominating set problem (DSP)

DSP	▷**NP**-hard

Input: an undirected graph $G = (V, E)$, an integer k
Objective: decide if there exists a set $S \subseteq V$ with $|S| = k$ such that each vertex is either in S or adjacent to a vertex in S.



Proof of Theorem 1

Idea: Reduction from dominating set problem (DSP)

DSP	$\triangleright$**NP**-hard
Input: an undirected graph $G = (V, E)$, an integer k Objective: decide if there exists a set $S \subseteq V$ with $|S| = k$ such that each vertex is either in S or adjacent to a vertex in S.	

The reduction: Given an instance $\langle G = (V, E), k \rangle$ of DSP, we create an instance $\langle H = (V, E'), d_{ij}, k \rangle$ of k-center problem:

- if $(i, j) \in E$, then $d_{ij} = 1$; otherwise, $d_{ij} = 2$

Note that $d_{ii} = 0$, $d_{ij} = d_{ji}$, and $d_{ij} \leq d_{i\ell} + d_{j\ell}$.

Proof of Theorem 1

Idea: Reduction from dominating set problem (DSP)

DSP	▷**NP**-hard
Input: an undirected graph $G = (V, E)$, an integer k	
Objective: decide if there exists a set $S \subseteq V$ with $\lvert S \rvert = k$ such that each vertex is either in S or adjacent to a vertex in S.	

The reduction: Given an instance $\langle G = (V, E), k \rangle$ of DSP, we create an instance $\langle H = (V, E'), d_{ij}, k \rangle$ of k-center problem:

- if $(i, j) \in E$, then $d_{ij} = 1$; otherwise, $d_{ij} = 2$

Note that $d_{ii} = 0$, $d_{ij} = d_{ji}$, and $d_{ij} \leq d_{i\ell} + d_{j\ell}$.

Key observations:

- there exists a DS of size k in G if and only if the optimal radius (or cost) of k-center problem in H is 1
- note any $(2 - \varepsilon)$-approx. for k-center problem can distinguish if the optimal solution of $\langle H, d_{ij}, k \rangle$ is 1 or 2

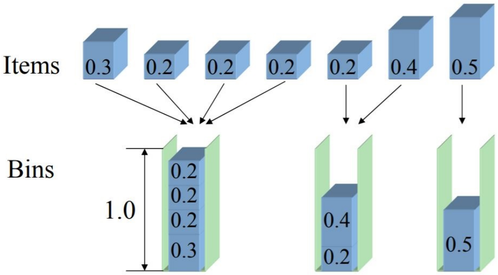
Example: bin packing problem

The bin packing problem

bin packing problem

Input: a set I of n items, each item o_i has size $s_i \in (0,1]$; a set B of n bins, each bin b_i has capacity 1

Objective: find an assignment $a : I \to B$ such that the number of non-empty bins is minimal.

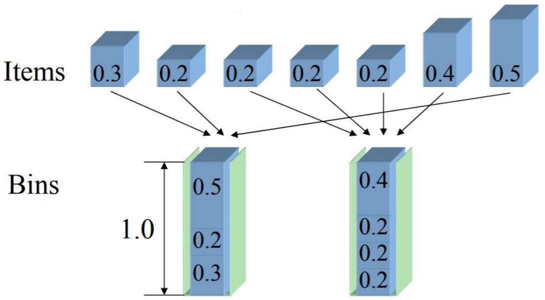


The bin packing problem

bin packing problem

Input: a set I of n items, each item o_i has size $s_i \in (0,1]$; a set B of n bins, each bin b_i has capacity 1

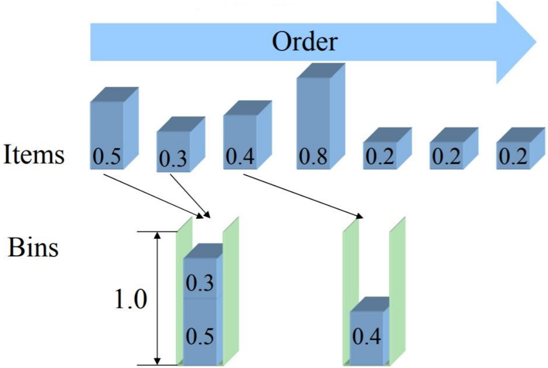
Objective: find an assignment $a : I \to B$ such that the number of non-empty bins is minimal.



A $2-$approximation algorithm

FirstFit

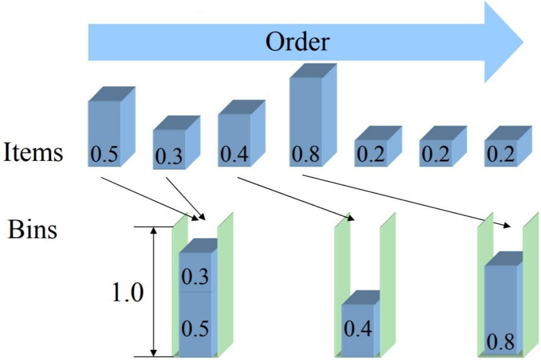
- Put each item in one of partially packed bins
  - If the item does not fit into any of these bins, open a new bin and put the item into it.



A 2−approximation algorithm

FirstFit

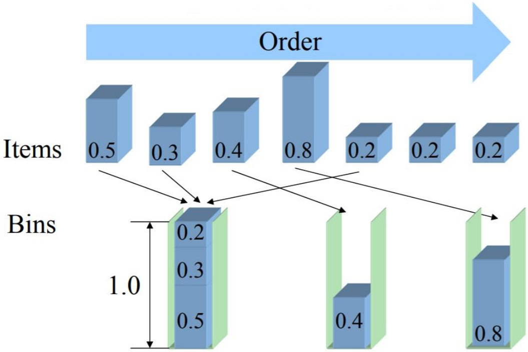
- Put each item in one of partially packed bins
  - If the item does not fit into any of these bins, open a new bin and put the item into it.



A $2-$approximation algorithm

FirstFit

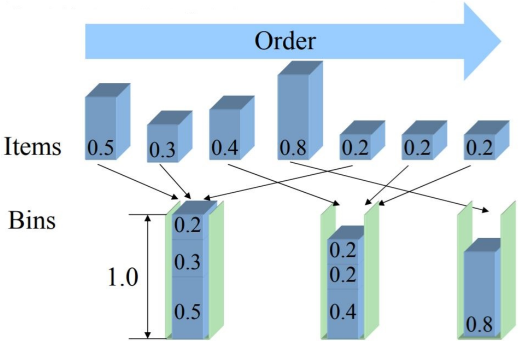
- Put each item in one of partially packed bins
  - If the item does not fit into any of these bins, open a new bin and put the item into it.



A $2-$approximation algorithm

FirstFit

- Put each item in one of partially packed bins
  - If the item does not fit into any of these bins, open a new bin and put the item into it.

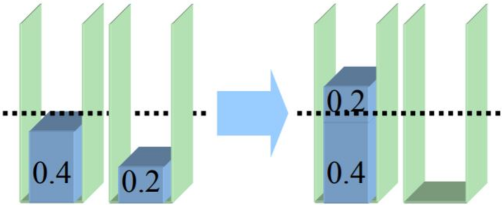


A 2-approximation algorithm: analysis

Theorem The algorithm FirstFit is a 2-approximation algorithm for the bin-packing problem.

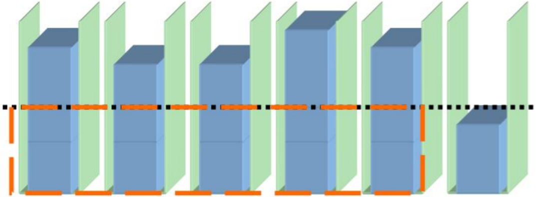
Proof: OPT: $\#$ bins used in the optimal solution
Suppose that FirstFit uses m bins
Then, at least $(m-1)$ bins are more than half full.

- We never have two bins less than half full.
  - If there are two bins less than half full, items in the second bin can be substituted into the first bin by FirstFit.



A 2-approximation algorithm: analysis

- Suppose that FIRSTFIT uses m bins
  Then, at least $(m-1)$ bins are more than half full; call such bins good



- Note that $\mathrm{OPT} \geq \sum_{i=1}^{n} s_i$, the sum of sizes of the items
- it also holds that $\sum_{i=1}^{n} s_i \geq \sum_{i \in \text{ good bins}} s_i = \sum_{j: \text{ good bin}}$ total size in bin $j > \frac{m-1}{2}$,
- thus, $2\mathrm{OPT} > m - 1$
- thus, $2\mathrm{OPT} \geq m$, as OPT and m are integers

Hardness of the bin packing problem

Theorem: Assuming $\mathbf{P} \neq \mathbf{NP}$, there is no $(\frac{3}{2} - \varepsilon)$-approximation algorithm for the bin packing problem, for any $\varepsilon > 0$.

Proof idea: A known and useful fact

partition problem

Input: a set of n numbers, each number c_i is an integer
Objective: decide if there exists a set $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$.

Fact: The partition problem is **NP**-hard

We can further assume that for each i, $c_i \leq \frac{\sum_{i=1}^{n} c_i}{2}$, as otherwise, we can simply answer "no".

Proof of hardness of the bin packing problem

We preform the following reduction from *Partition* to *Bin Packing*:

Reduction: Let I be an instance of *Partition* with numbers $c_1, \ldots, c_n$, then the instance I' of *Bin Packing* has:

- one item o_i with size $s_i = 2c_i/C$, where $C = \sum_{i=1}^{n} c_i$
  – note that each $s_i \leq 1$, as for each i, we assumed that $c_i \leq \frac{C}{2}$.

Proof of hardness of the bin packing problem – cont.

- The given reduction clearly runs in polynomial-time.
- Let I be an instance of *Partition* and let I' be the instance of *Bin Packing* obtained from I using the given reduction, then:
  - (P1) Let S be a solution for I, then we can easily pack the items of I' into 2 bins, i.e., one bin containing all items in S and one bin containing all items not in S.
  - (P2) If I has no solution then one needs at least 3 bins to fit all the items of I'. (Reason: if we divide the items into two groups, at least one group has total size > 1, which exceeds the capacity of one bin.)

Proof of hardness of the bin packing problem - cont.

Suppose there exists $(\frac{3}{2} - \varepsilon)$-approximation algorithm for *Bin Packing*, then we could solve *Partition* in polynomial-time as follows (Recall that I is a given instance of *Partition*):

- apply the previous polynomial-time reduction to I to obtain the instance of *Bin Packing* I',
- run the $(\frac{3}{2} - \varepsilon)$-approximation algorithm for *Bin Packing* on I' and:
  - if the solution for I' uses fewer than 3 bins, then answer that I is a yes-instance,
  - otherwise answer that I is a no-instance

Proof of hardness of the bin packing problem – cont.

The resulting algorithm (that combines the above two processes) clearly runs in polynomial-time. Furthermore:

- $\rightarrow$ If I is a yes-instance, then by (P1) I' has a solution using at most 2 bins. Hence the $(\frac{3}{2} - \varepsilon)$-approximation algorithm will return a solution using at most $(\frac{3}{2} - \varepsilon) \cdot 2 < 3$ bins, as required.

- $\leftarrow$ If the $(\frac{3}{2} - \varepsilon)$-approximation algorithm on I' finds a solution using less than 3 bins, then by (P2) I does have a solution.

This contradicts the assumption that **P** $\neq$ **NP**, since *Partition* is **NP**-hard).

Thus, assuming **P** $\neq$ **NP**, there is no $(\frac{3}{2} - \varepsilon)$-approximation algorithm for the bin packing problem, for any $\varepsilon > 0$.

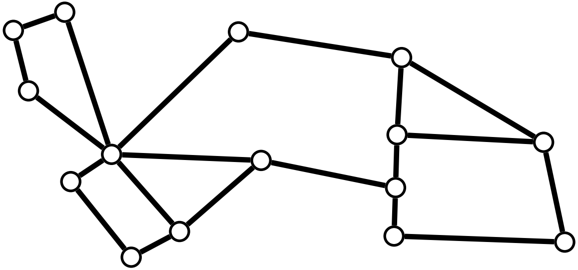
Example: minimum vertex cover problem

Minimum vertex cover problem

Vertex cover of a graph: a subset C of its vertices such that for each edge $\{u, v\}$ at least one endpoint u or v is in C

minimum vertex cover problem

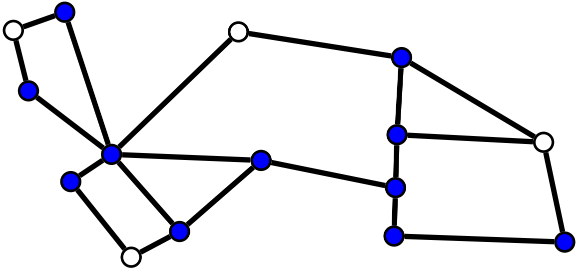
Input: an undirected graph $G = (V, E)$
Objective: find a vertex cover of G of smallest possible size.



Minimum vertex cover problem

Vertex cover of a graph: a subset C of its vertices such that for each edge $\{u, v\}$ at least one endpoint u or v is in C

> minimum vertex cover problem
>
> Input: an undirected graph $G = (V, E)$
> Objective: find a vertex cover of G of smallest possible size.



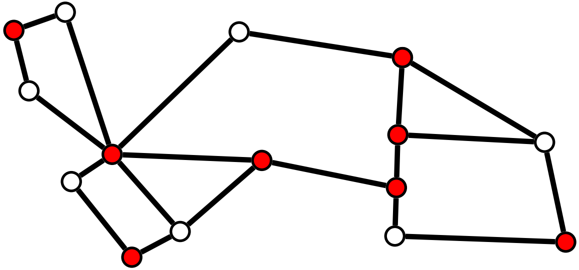
A vertex cover of size 11

Minimum vertex cover problem

Vertex cover of a graph: a subset C of its vertices such that for each edge $\{u, v\}$ at least one endpoint u or v is in C

minimum vertex cover problem

Input: an undirected graph $G = (V, E)$
Objective: find a vertex cover of G of smallest possible size.



A vertex cover of size 8

Minimum vertex cover problem

Vertex cover of a graph: a subset C of its vertices such that for each edge $\{u, v\}$ at least one endpoint u or v is in C

> minimum vertex cover problem
>
> Input: an undirected graph $G = (V, E)$
> Objective: find a vertex cover of G of smallest possible size.

Background: there exists a 2-approximation algorithm for the minimum vertex cover problem

Minimum vertex cover problem

Vertex cover of a graph: a subset C of its vertices such that for each edge $\{u, v\}$ at least one endpoint u or v is in C

minimum vertex cover problem
Input: an undirected graph $G = (V, E)$ Objective: find a vertex cover of G of smallest possible size.

Background: there exists a 2-approximation algorithm for the minimum vertex cover problem

Theorem

Assuming **P** $\neq$ **NP**, *there is no $(1 + \eta)$-approximation algorithm for the minimum vertex cover problem, where η is some constant such that $1 < \eta < 2$.*

Proof idea

Reduction from 3-SAT (3-SATISFIABILITY) to the minimum vertex cover problem:

Given an instance ϕ of 3-SAT, we create a graph $G = (V, E)$ such that

- if ϕ is satisfiable, then G has a vertex cover of size $\leq \frac{2}{3}|V|$, and
- if ϕ is not satisfiable, then the smallest vertex cover of G has size $> (1 + \eta) \cdot \frac{2}{3}|V|$

Proof idea

Reduction from 3-SAT (3-SATISFIABILITY) to the minimum vertex cover problem:

Given an instance ϕ of 3-SAT, we create a graph $G = (V, E)$ such that

- if ϕ is satisfiable, then G has a vertex cover of size $\leq \frac{2}{3}|V|$, and
- if ϕ is not satisfiable, then the smallest vertex cover of G has size $> (1 + \eta) \cdot \frac{2}{3}|V|$

The above reduction is based on the following fundamental and quite difficult result which is equivalent to the PCP theorem

Theorem: There is a gap reduction f from 3-SAT to Max-3SAT with the following properties: there exists an absolute constant ε_0 such that

- if ϕ is satisfiable then $f(\phi)$ is also satisfiable
- if ϕ is not satisfiable then less than $1 - \varepsilon_0$ fraction of clauses in $f(\phi)$ are satisfiable by any assignment to $f(\phi)$

- cf. textbook “Approximation Algorithms” Chapter 29 for details

Some representative hardness results

Assuming **P** $\neq$ **NP**, there is no polynomial time algorithm for finding an $\alpha^*(\Pi)$-approximation solution of the problem Π.

Some representative hardness results

Assuming **P** $\neq$ **NP**, there is no polynomial time algorithm for finding an $\alpha^*(\Pi)$-approximation solution of the problem Π.

Problem Π	$\alpha^*(\Pi)$	the best known approx. ratio
min. vertex cover	$\sqrt{2} - \varepsilon$	2
max. cut	$\frac{16}{17} + \varepsilon$	0.878
k-center	$2 - \varepsilon$	2
max. 2-SAT	$0.954 + \varepsilon$	0.940
minimum set cover	$(1 - o(1)) \cdot \ln n$	$\ln n - \ln \ln n + \Theta(1)$
max. clique, max. independent set	$O(n^{\varepsilon - 1})$	

- The red numbers: upper bound and lower bound almost tight

Some representative hardness results

Another commonly used conjecture for proving hardness of approximation: unique games conjecture

- cf. “recommended textbooks”

Assuming that **the unique games conjecture is true**, there is no polynomial time algorithm for finding an $\alpha^*(\Pi)$-approximation solution of the problem Π.

Problem Π	$\alpha^*(\Pi)$
min. vertex cover	$2 - \varepsilon$
max. cut	$0.878 + \varepsilon$
max. 2-SAT	$0.9439 + \varepsilon$

References for hardness of approximation

- *The Design of Approximation Algorithms* Chapter 16
  David Williamson and David Shmoys
  Cambridge University Press, 2011.

- *Approximation Algorithms*. Chapter 29
  Vijay Vazirani
  Springer-Verlag, 2004.