Introduction to Approximation Algorithms

Marcin Sydow

PhD STUDIES

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Project co-financed by European Union within the framework of European Social Fund

Selected Bibliography

Easy introduction:

- T.H.Cormen et al. "Introduction to Algorithms" 3rd edition, MIT Press 2009, chapters 34,35

Specialised textbooks:

- V.Vazirani "Approximation Algorithms", Springer 2003, chapters 1,2,3,12
- D.Williamson, D.Shmoys "The Design of Approximation Algorithms", Cambridge University Press, 2011
- (Ed. by T.Gonzalez) "Handbook of Approximation Algorithms and Metaheurisitcs", Chapman & Hall/CRC, 2007
- (Ed. by D.Hochbaum) "Approximation Algorithms for NP-hard Problems", PWS 1997

NP-hard

Multiple ways to deal with NP-hard problems:

- considering special cases
- fast heuristics (local search, genetic algorithms, etc.)
- fast, exponential algorithms
- randomised algorithms
- **approximation algorithms**

NP-optimisation problem (a bit more formally)

NP-optimisation problem Π consists of:

- set of *valid instances*, D_Π, recognisable in polynomial time (assume: all the numbers are rational, and encoded in binary, $|I|$ denotes the size of encoded instance I, in bits).
- each instance $I \in D_\Pi$ has a set of *feasible solutions*, $S_\Pi(I) \neq \emptyset$. Each feasible solution $s \in S_\Pi(I)$ is of length bounded by polynomial of $|I|$. Moreover, there is a polynomial algorithm that given a pair (I, s) decides whether $s \in S_\Pi(I)$
- there is a polynomially computable *objective function* obj_Π which assigns a nonnegative rational number to each pair (I, s) (an instance and its feasible solution).
- Π is specified to be either *minimisation* or *maximisation* problem

NP-optimisation problem, cont.

Optimal solution of an instance of a minimisation (maximisation) problem is a feasible solution which achieves the minimum (maximum) possible value of the objective function (called also "cost" for minimisation or "profit" for maximisation).

$OPT_{\Pi}(I)$ denotes optimum objective function value for an instance I

Decision version of an NP-optimisation problem I: a pair (I, B), where $B \in Q$ and the decision problem is stated as "does there exist a feasible solution to I of cost $\leq B$, for minimisation problem I" (or, analogously "of profit $\geq B$", for a maximisation problem)

Extending the definition of NP-hardness for optimisation problems

Decision version can be “reduced” to optimisation version. (i.e. polynomial algorithm for optimisation version can obviously solve the decision version)

NP-optimisation problem can be called NP-hard if its decision version is NP-hard.

Example: Vertex Cover

Given a graph $G = (V, E)$, find a subset of its vertices $V' \subseteq V$ that:

- "covers" all edges, i.e. each edge $e \in E$ is incident with at least one vertex from V' (*feasibility* constraint)
- $|V'|$ is minimum possible (*cost function* to be minimised)

VC is NP-complete (e.g. reduction from 3-SAT via Independent Set)

Solving NP-hard problems on special cases may be easy.
E.g. VC on cycles.

Fast Exponential Algorithms

Finding small VC for fixed k

If **k is fixed** (e.g. k=3 or 10) then VC is has an algorithm that is **polynomial** of n:
try all k-subsets of V (there are $\frac{n^k}{(n-k)!}$ such) and check each (in time $O(kn)$) if it forms a VC. (total: $O(kn^{k+1})$ - a polynomial of n)
However, such a polynomial-time algorithm is infeasible even for moderate values of k and n (e.g. n= 1000, and k=10).
Interestingly, for small k there is a **exponential-time** algoritm for VC that is more efficient

Two observations

If G has a vertex cover of size at most k, then G has at most $k(n-1)$ edges.

Lemma

Let $G=(V,E)$ be a graph and $(u,v)\in E$. G has a vertex cover of size at most k if at least one of the graphs $G\setminus\{u\}$ or $G\setminus\{v\}$ has a vertex cover of size at most k-1.
The above two observations lead directly to a recursive algorithm for VC that is feasible for small values of k.

Recursive algorithm for VC

If G contains no edges, then the empty set is a VC
If G contains more than *kn* edges than it has no k-node VC
Else let $e = (u, v)$ be an edge of G
Recursively check if either $G \setminus \{u\}$ or $G \setminus \{v\}$ has a VC of size at most k-1.
If neither of them does, then G ha no k-node VC
Else, one of the (say, the first one) has a $(k-1)$-node VC, call it T
In such case, $T \cup \{u\}$ is a k-node VC for G

Time complexity of the algorithm is $O(2^k \cdot kn)$.
The algorithm has **exponential** time but only in k[1].
Thus, for small k, it is **more efficient than polynomial-time** algorithm.
(e.g. k=10 and n=1000).
However, for higher values of k this approach is infeasible.

[1] Thus the problem is called *fixed-parameter tractable* (FPT)

Local Search Heuristics

Local Search Heuristics

For feasible solution set S, define *neighbourhood relation*. Walk from neighbour to neighbour (by small modifications) to find *local* optimum.

Examples:

- Hill Climbing
- Simulated Annealing

Positive:

- fast
- extrememly flexible
- easy to implement

Negative:

- no guarantee for finding global optimum
- no guarantee of how far the found solution is from the global optimum.

Neighbourhood Relation

Binary relation on solutions that satisfies:

- connected
- neighbours are similar
  - easy to compute/small modification
  - objective function has similar value
- diameter is small (polynomial of task size)
- neighbourhood is small (polynomial of task size)

Simulated Annealing (SA) Algorithm

```
simulatedAnnealing(MaxIter, Gap)
s ← randomInitialSolution()
best ← s; step ← 0; change ← step; temperature ← 1
while (step < MaxIter ∧ step − change < Gap)
do step = step + 1
    s′ = randomNeighbour(s)
    if random(0, 1) < P(s, s′, temperature)
       then s = s′
    if f(s) < f(best)
       then best = s; change = step
    temperature = 1/√step

return best
```

transition probability function:

$$P(s, s', temperature) = \begin{cases} 1 & s' \text{ better than } s \\ e^{\frac{-|f(s)-f(s')|}{temperature}} & else \end{cases}$$

$MaxIter = 100,000$, $Gap = 2000$ (tuned experimentally)

Combinatorial Approximation Algorithms: Examples

Approximation Algorithm

Let Π be a minimisation (maximisation) problem, $\delta : Z^+ \rightarrow Q^+$ a function that has values ≥ 1 (≤ 1).

Definition

An algorithm A is a *factor δ approximation algorithm* for Π if, for each instance I, A finds a feasible solution s for I such that: $obj_{\Pi}(I, s) \leq \delta(|I|) \cdot OPT(I)$
(for maximisation: $obj_{\Pi}(I, s) \geq \delta(|I|) \cdot OPT(I)$)

Observation: The closer δ to the value of 1, the better approximation.
Remark: δ can be also a function of some other parameter than length of input instance (|I|).

Example: Approximation of Vertex Cover

The vertex cover problem is an NP-optimisation problem and it is NP-hard because even its decision version is NP-hard

Obviously, no polynomial-time algorithm that finds optimum is known for this problem.

There will be presented a factor-2 (polynomial time) approximation algorithm for Vertex Cover.

Lower Bounding

Note: for NP-hard optimisation problems specifying just the value of OPT is also computationally hard.

Problem: how to provide the approximation guarantee when the value of OPT is not known?

One of the methods for minimisation problems is the *Lower Bounding* technique[2]:

Find a *polynomially computable lower bound* for OPT that is *at the same time naturally related to a feasible solution* to the considered optimisation problem.

[2] for maximisation problems the analogous *upper bounding* can be applied

Illustration: Lower Bounding for Vertex Cover

The Matching Problem: Given a graph $G = (V, E)$, a subset S of its edges is called a *matching* iff no two edges in S are incident with a common vertex in G.

A matching in a given particular graph $G = (V, E)$ is called:

- *maximal*, if it cannot be extended by any edge in G
- *maximum*, if it achieves maximum cardinality over all matchings in G

Key observation: **size of any maximal matching M is a lower bound for optimal vertex cover ($|M| \leq OPT$)** (Why?).

Illustration: Lower Bounding for Vertex Cover

The Matching Problem: Given a graph $G = (V, E)$, a subset S of its edges is called a *matching* iff no two edges in S are incident with a common vertex in G.

A matching in a given particular graph $G = (V, E)$ is called:

- *maximal*, if it cannot be extended by any edge in G
- *maximum*, if it achieves maximum cardinality over all matchings in G

Key observation: **size of any maximal matching M is a lower bound for optimal vertex cover ($|M| \leq OPT$)** (Why?).

(because at least one endpoint of *each* edge in any matching S has to be selected to make vertex cover, otherwise some edge in S would be left uncovered)

Illustration: Lower Bounding for Vertex Cover

The Matching Problem: Given a graph $G = (V, E)$, a subset S of its edges is called a *matching* iff no two edges in S are incident with a common vertex in G.

A matching in a given particular graph $G = (V, E)$ is called:

- *maximal*, if it cannot be extended by any edge in G
- *maximum*, if it achieves maximum cardinality over all matchings in G

Key observation: **size of any maximal matching M is a lower bound for optimal vertex cover ($|M| \leq OPT$)** (Why?). (because at least one endpoint of *each* edge in any matching S has to be selected to make vertex cover, otherwise some edge in S would be left uncovered)

On the other hand, **selecting *both* endpoints of each edge in a *maximal* matching makes a vertex cover!**

2-Approximation Algorithm for Vertex Cover

Thus, we obtain the following *2-approximation algorithm* for vertex cover:

`Find any maximal matching` M `in the input graph and output the set` W `of all matched vertices`

Since a maximal matching[3] can be found in polynomial time by a simple greedy algorithm we obtain polynomial-time 2-approximation for vertex cover.

Proof: All edges in the graph are covered by the set W of picked vertices (any uncovered edge could be added to the matching M, contradicting its maximality). $|M| \leq OPT$ (previous slide) and the found vertex cover W has cardinality $|W| = 2 \cdot |M|$ so that $|W| \leq 2 \cdot OPT$

---

[3] maximum matching can be found in polynomial time, too

What can be improved?

1. Is it possible to improve the approximation guarantee of the algorithm by a better analysis?
2. Is it possible to use the same lower bounding scheme (i.e. the size of a maximal matching) to design another approximation algorithm with a better guarantee?
3. Is there a different lower bounding scheme that can result in a better approximation guarantee for vertex cover?

Tight example

Is it possible to improve the approximation guarantee of the algorithm by a better analysis?

For example, for any *complete bipartite graph* $K_{n,n}$, OPT is n but the vertex cover found by the algorithm has cardinality $2 \cdot n$. So that the approximation guarantee limit is reached exactly.

Any infinite family of instances that achieve the approximation guarantee (also asymptotically) is called a **tight example**.

Thus, $K_{n,n}$ is a tight example for the presented 2-approximation algorithm and the answer to the question is “no”.

Tight examples play a crucial role in designing approximation algorithms.

Can factor of 2 be improved with this lower bounding scheme?

Is it possible to use the same lower bounding scheme (i.e. the size of a maximal matching) to design another approximation algorithm with a better guarantee?

The answer to this question is also "no", since there exists an infinite family of instances for which the size of maximal matching is indeed *2 times smaller* than the cardinality of the minimum vertex cover.

To see this, consider a full graph K_n for odd n. (what is the size of maximal matching here? Of minimum vertex cover?

Can factor of 2 be improved with this lower bounding scheme?

Is it possible to use the same lower bounding scheme (i.e. the size of a maximal matching) to design another approximation algorithm with a better guarantee?

The answer to this question is also “no”, since there exists an infinite family of instances for which the size of maximal matching is indeed *2 times smaller* than the cardinality of the minimum vertex cover.

To see this, consider a full graph K_n for odd n. (what is the size of maximal matching here? Of minimum vertex cover?
(any maximal matching in it has size $(n-1)/2$ and minimum vertex cover has cardinality $n-1$)

Can factor of 2 be improved with another approximation algorithm?

Is there a different lower bounding scheme that can result in a better approximation guarantee for vertex cover?

There is no approximation algorithm with *constant* factor better than 2 for vertex cover. [Dinur, Safra, 2005]

However, in 2004, an algorithm was found with a bit better approximation factor of $2 - \Theta(1/\sqrt{log|V|})$ [Karakostas 2004]

Set Cover Problem

Given a universe U of n elements, a family $\mathcal{S} = S_1, ..., S_k$ of subsets of U and a cost function $c : \mathcal{S} \rightarrow \mathcal{Q}^+$, find a minimum cost subfamily of $\mathcal{S}$ that covers all elements of U.

The set cover problem is very important in approximation algorithms. It is possible to illustrate multiple concepts and techniques on this problem.

The *frequency* of an element of U is the number of sets in $\mathcal{S}$ that contain it. Let f denote the maximum frequency of an element in a given set cover instance.

Various approximation algorithms for set cover achieve either f or $O(log(n))$ approximation factor guarantee. Notice that neither dominates the other for all possible instances.

Vertex Cover as Set Cover

Vertex cover can be viewed as a special case of set cover. (Why?)

Vertex Cover as Set Cover

Vertex cover can be viewed as a special case of set cover. (Why?)

Let $G = (V, E)$ be the input graph in vertex cover. Define $U = E$ and for $1 < i < |V|$ let S_i be the set of all edges in E incident with the vertex $v_i \in V$. Thus, any solution to such defined set cover instance is a valid solution to the vertex cover (notice: this is actually an example of *reduction* of vertex cover to set cover).

Vertex Cover as Set Cover

Vertex cover can be viewed as a special case of set cover. (Why?)

Let $G = (V, E)$ be the input graph in vertex cover. Define $U = E$ and for $1 < i < |V|$ let S_i be the set of all edges in E incident with the vertex $v_i \in V$. Thus, any solution to such defined set cover instance is a valid solution to the vertex cover (notice: this is actually an example of *reduction* of vertex cover to set cover).

What is the value of f in vertex cover viewed as set cover in the way described above?

Vertex Cover as Set Cover

Vertex cover can be viewed as a special case of set cover. (Why?)

Let $G = (V, E)$ be the input graph in vertex cover. Define $U = E$ and for $1 < i < |V|$ let S_i be the set of all edges in E incident with the vertex $v_i \in V$. Thus, any solution to such defined set cover instance is a valid solution to the vertex cover (notice: this is actually an example of *reduction* of vertex cover to set cover).

What is the value of f in vertex cover viewed as set cover in the way described above?

$f = 2$, since each edge is present in exactly two sets from family $\mathcal{S}$ (represented by its endpoints).

Vertex Cover as Set Cover

Vertex cover can be viewed as a special case of set cover. (Why?)

Let $G = (V, E)$ be the input graph in vertex cover. Define $U = E$ and for $1 < i < |V|$ let S_i be the set of all edges in E incident with the vertex $v_i \in V$. Thus, any solution to such defined set cover instance is a valid solution to the vertex cover (notice: this is actually an example of *reduction* of vertex cover to set cover).

What is the value of f in vertex cover viewed as set cover in the way described above?

$f = 2$, since each edge is present in exactly two sets from family $\mathcal{S}$ (represented by its endpoints).

Notice, that the 2-approximation algorithm for vertex cover achieves exactly the value of f as approximation factor guarantee.

Greedy Algorithm for Set Cover

The algorithm iteratively picks the most *cost-effective* set $S \in \mathcal{S}$ until U is covered. Let $C \subseteq U$ denote the set of elements of U already covered in current iteration.

Define the *cost-effectiveness* of a set $S \in \mathcal{S}$ as the averaged cost of covering uncovered elements: $c(S)/|S - C|$. Define the *price* of an element of U as the average cost at which it is covered.

1. $C = \emptyset$
2. while $C \neq U$ find the lowest cost-effectiveness set S, pick it, and for each $e \in S \setminus C$ set its $price(e) = c(S)/|S \setminus C|$, update $C = C \cup S$
3. output the picked sets

Approximation Guarantee Analysis

Let the order of elements of U covered by the algorithm be $e_1, ..., e_n$ (resolving any ties arbitrarily) Lemma:

For any $1 \leq k \leq n$, $price(e_k) \leq OPT/(n-k+1)$.

Proof: At any moment, all the yet uncovered elements can be covered by unused sets from the optimal solution at total cost not higher than OPT. Thus, there has to be at least one such set with cost-effectiveness not higher than $OPT/|U \setminus C|$. When the element e_k was covered by the most cost-effective set at the moment, the set of uncovered elements $U \setminus C$ had at least $n-k+1$ elements, so that:
$price(e_k) \leq OPT/|U \setminus C| \leq OPT/(n-k+1)$

O(log(n)) approximation guarantee

The greedy algorithm is $O(log(n))$-approximation algorithm for the set cover problem.

Proof: Because the cost of each picked set is distributed over all newly covered element, the total cost of the picked set is $\sum_{k=1}^{n} price(e_k)$. Due to the lemma, it is upper bounded by $(1/n + ... + 1/2 + 1) \cdot OPT = H_n \cdot OPT$. But the harmonic number H_n grows assymptotically as the logarithm of n: $H_n = O(log(n))$.

Tight example for Greedy Set Cover Algorithm

Consider the following instance. $U = \{1, ..., n\}$, $\mathcal{S}$ consists of $n + 1$ sets, where $S_i = \{i\}$ and $c(S_i) = 1/i$, for $1 \leq i \leq n$ and $S_{n+1} = U$ with $c(S_{n+1}) = 1 + \epsilon$ for arbitrarily small $\epsilon > 0$.

Obviously, the greedy algorithm will iteratively pick n sets: $S_n, S_{n-1}, ..., S_1$ achieving the total cost of $1/n + 1/(n-1) + ... + 1 = H_n$ while the optimal choice is to pick only S_{n+1} with the total cost of $1 + \epsilon$.

Steiner Tree Problem

Given an undirected graph $G = (V, E)$ with non-negative edge costs and with distinguished subset R of *required* vertices the goal is to find a tree of minimum cost in G containing all required vertices.

In *metric* Steiner Tree variant of the problem the graph is *complete* and the weights satisfy the *triangle inequality*: for any vertices u, v, w, $cost(u, v) \leq cost(u, w) + cost(w, v)$.

Theorem: Steiner Tree can be *reduced* to metric Steiner Tree with preserving the approximation factor.

This means that metric Steiner Tree is “not easier” than Steiner Tree problem. In addition, any factor-α approximation algorithm for metric Steiner Tree problem carries over the general Steiner Tree problem.

Proof of the Theorem

Let I be an instance of the ST problem on undirected graph $G = (V, E)$ with costs on edges. The polynomial transformation of I to I' representing metric ST is as follows. Let $G' = (V, E')$ be the complete undirected graph obtained from G by completing all missing edges and setting costs on all edges (u, v) as the shortest-path cost in G from u to v (such a graph is called the "metric closure" of G). Obviously, it represents metric ST problem. Any cost in G' is not higher than in G, so $OPT(I') \leq OPT(I)$. Now, any Steiner tree T' found on G' can be transformed to a Steiner tree T on G without increasing its cost in polynomial time as follows. Replace each edge in T' with the shortest path that it represents, to obtain a connected graph containing all the required vertices. As it can contain cycles, finally, remove some edges to obtain a Steiner Tree.

From now on, it will be enough to assume that the input graph in ST is complete and metric (since the obtained result can be easily translated "back" to any arbitrary input graph as above).

MST-based Approximation Algorithm

A MST on R (required vertices) is a feasible solution to ST problem. However, since MST is in P and ST is NP-hard, it is not always the optimal solution. (Example?)

There exists the following lower bound, though:

Theorem: *The cost of MST on R is not higher than $2 \cdot OPT$.*

Proof of the Lower Bound on ST

Let T be a Steiner tree of cost OPT. Let substitute each edge in T with two opposite *directed* edges (keep the weights fixed), to obtain an Eulerian graph containing all required vertices. Find in this graph an Euler cycle; it has cost $2 \cdot OPT$. Next, "reduce" it to a Hamiltonian cycle by "short-cutting" (in the whole input graph) all previously visited or non-required vertices. Due to triangle inequality, the Hamiltonian cycle has not higher cost than the Euler cycle. Finally, delete one edge from it to obtain a tree containing all required vertices and of cost not higher than $2 \cdot OPT$.

The above procedure naturally describes a **2-approximation algorithm** for (metric) Steiner Tree problem.

Tight example

Consider a complete graph with the set R of n required vertices and a single non-required vertex. The cost of any edge between two required vertices is 2 and 1 otherwise. Any MST on R has cost $2(n-1)$ but $OPT = n$. Thus, it is a tight example (assymptotically[4])

A question:
Why cannot we set the weights to 3 (for example) instead of 2 to obtain factor of 3?

[4] i.e. we can obtain any value arbitrarily close to 2 for sufficiently high n

Tight example

Consider a complete graph with the set R of n required vertices and a single non-required vertex. The cost of any edge between two required vertices is 2 and 1 otherwise. Any MST on R has cost $2(n-1)$ but $OPT = n$. Thus, it is a tight example (assymptotically[4])

A question:
Why cannot we set the weights to 3 (for example) instead of 2 to obtain factor of 3?
Because, for any value higher than 2 we violate the triangle inequality.

[4] i.e. we can obtain any value arbitrarily close to 2 for sufficiently high n

TSP and its inapproximability

Given a complete graph $G(V, E)$ with non-negative costs on edges find a hamiltonian cycle of minimum possible total cost.

Interestingly, TSP **cannot be approximated** unless $P = NP$.

This can be stated more formally as follows:

Theorem: *For any polynomially computable function $\alpha : Z^+ \to Q^+$, there is no $\alpha(|V|)$-approximation algorithm for TSP assuming $P \neq NP$.*

Proof of inapproximability of TSP

Proof: Assume the opposite. This would lead to a polynomial-time algorithm for deciding Hamiltonian Cycle (HC) problem that is known to be NP-complete. To this end, let's transform HC to TSP as follows. Let $G = (V, E)$ represent a given instance I of HC. To obtain the corresponding instance I' of TSP, set the cost on each edge from E to 1 and add all the missing edges with costs of $\alpha(n) \cdot n$, where $n = |V|$.
Now, run the polynomial-time $\alpha(n)$-approximation algorithm on I'. If the found TSP solution has cost of n, then I has Hamiltonian cycle. Otherwise, the TSP solution has to use at least one "heavy" edge, so that its cost is higher than $\alpha(n) \cdot n$. Thus, we could use the approximation algorithm to effectively *decide* the HC problem in polynomial time. But, since HC is NP-complete, this would mean that $P = NP$.

Metric TSP

In the proof of inapproximability, the edge costs violate the triangle inequality. Otherwise, we obtain the *metric* variant of TSP and the following theorem:

Theorem:
For *metric* TSP, there exists a 2-approximation algorithm.

The algorithm uses the lower bounding technique. More precisely, it is based on the observation that cost of MST is a lower bound for TSP (because deleting any edge from a TSP results in a spanning tree).

2-approximation algorithm for metric TSP

The algorithm is related with the presented proof for lower bound on Steiner Tree problem.

Let G be the input graph to the metric TSP problem.

1. find a MST for G (call it T)
2. substitute each edge in T with a pair of opposite directed edges
3. find an Eulerian tour T' on this graph
4. return the tour that visits vertices of G in the order of their first appearance in T', call it C.

Remark: the last step is very similar to the "short-cutting" idea presented before.

factor-2 approximation guarantee

The presented algorithm has factor-2 approximation guarantee for metric TSP.

Proof:
As was stated before, $cost(T) \leq OPT$. Due to the construction of T', $cost(T') = 2 \cdot cost(T)$. Since we consider the metric variant of TSP, $cost(C) \leq cost(T)$. Thus, we obtain that $cost(C) \leq 2 \cdot OPT$.

Tight Example

Consider a complete n-vertex graph $G = (V, E)$. Let $v \in V$ and $S = V \setminus \{v\}$. Select a $(n-1)$-element cycle in G and set cost of each edge in it to 1. Also, set the cost of each edge incident to v as 1. The remaining edges have costs of 2.
Now, the optimal TSP tour has cost of n (traverse the cycle except one edge and visit v).
On the other hand, if the found MST is "star-shaped" with v in the center (cost of n), after "short-cutting" it has cost of 2n-2. Thus, we assyptotically obtain the factor of 2.

How to improve the algorithm?

There is a cheaper way of obtaining an Euler tour than doubling the edges of a MST.

Namely, a graph is Eulerian iff all the vertices have even degrees. Thus, it suffices to focus only on the set, call it V', of odd-degree vertices of MST. |V'| is even (since the sum of all degrees in any graph is even). Now, it is enough to add to the MST a *minimum cost perfect matching* (a perfect matching exists due to evenness) on V'. The result is the demanded cheaper Eulerian graph.

3/2-approximation algorithm for metric TSP

Christofides algorithm:

1. Find MST, T, on G
2. Compute a perfect matching M of minimum cost on the set of vertices of T that have odd degree.
3. Add M to T to obtain an Eulerian graph
4. Find an Euler tour T' on it
5. The result, C, is the tour that visits all vertices of G in the order of their first appearance in T'.

Interestingly, another lower bound is used to prove the guarantee of the algorithm.

Lemma

Let M be a minimum cost perfect matching on an even-cardinality set of vertices $V' \subseteq V$. Then, $cost(M) \leq OPT/2$ (for TSP problem).

Proof: Let τ be an optimal TSP tour in G and τ' be he tour on V' obtained by short-cutting τ. $cost(\tau') \leq cost(\tau)$ (triangle equality). But τ' is a union of two perfect matchings on V', consisting of alternate edges of τ' each. The cheaper of these matchings has $cost \leq cost(\tau') \leq OPT/2$. Thus, the optimal matching does not exceed $OPT/2$.

3/2 approximation guarantee for metric TSP

Theorem: the presented algorithm has 3/2 approximation guarantee for metric TSP.

Proof: The cost of the Euler tour is at most the sum of costs of the MST T (at most OPT) and matching M (at most OPT/2) found by the algorithm. Due to metric inequality, the "short-cutting" step does not increase the cost, so that it does not exceed $3/2 \cdot OPT$

Tight example

(on picture)

The MST found by the algorithm has only 2 odd-degree vertices. After joining them by the perfect matching that consists of a single edge with cost $\lfloor n/2 \rfloor$ we obtain a solution of cost $(n-1) + \lfloor n/2 \rfloor$ that assymptotically is 1.5 times more costly than optimum (cost of n).

Linear Programming

Linear Programming

Numerous important optimisation problems can be represented as *linear programs* (LP).

A linear program in *standard form* consits of:

- a *linear objective function* to be mimised (or maximised)
- and a set of *linear inequality constraints*.

The *decision variables* are non-negative.

For example:

minimise

$$6x_1 + 2x_2 + 4x_3$$

subject to

$$2x_1 - x_2 + x_3 \geq 7$$

$$3x_1 + x_2 + 2x_3 \geq 5$$

$$x_1, x_2, x_3 \geq 0$$

LP standard form

Demanding that constraints are in the form of *inequalities* of the same type (minimisation: $\geq$, maximisation: $\leq$) and that variables are non-negative does not limit the applications of LP since otherwise it can be transformed to standard form by simple operations (multiplication by -1 or two inequalities for equality)

Integer Program (IP)

LP is in P. However, if the variables are additionally required to be integers, the problem is known as *integer programming* (IP) and is $NP-hard$ (e.g. Set Cover can be expressed as IP).

The most-known algorithm for LP is Simplex. Simplex is not polynomial in worst case, but it is very efficient in practice. On the other hand, the ellipsoid algorithm is polynomial but it is less efficient on many cases than Simplex.

LP-based Tools

LP theory provides useful tools for designing and analysing exact and approximation algorithms.

In particular, the following techniques are very useful:

- rounding
- primal-dual schema
- dual-fitting

Example: Set Cover Problem (Reminder)

Given a universe U of n elements, a family $\mathcal{S} = S_1, ..., S_k$ of subsets of U and a cost function $c : \mathcal{S} \rightarrow \mathcal{Q}^+$, find a minimum cost subfamily of $\mathcal{S}$ that covers all elements of U.

The set cover problem is very important in approximation algorithms. It is possible to illustrate multiple concepts and techniques on this problem.

The *frequency* of an element of U is the number of sets in $\mathcal{S}$ that contain it. Let f denote the maximum frequency of an element in a given set cover instance.

Various approximation algorithms for set cover achieve either f or $O(log(n))$ approximation factor guarantee. Notice that neither dominates the other for all possible instances.

Set cover as linear (integer) program

How to express Set Cover in the form of integer program?

Set cover as linear (integer) program

How to express Set Cover in the form of integer program?

minimise

$$\sum_{S \in \mathcal{S}} c(S) x_S$$

subject to

$$\sum_{S \ni e} x_S \geq 1, \;\; e \in U$$

$$x_S \in \{0, 1\}, \;\; S \in \mathcal{S}$$

Interpretation: decision variable $x_S == 1$ iff S is picked. Each constraint guarantees coverage of each element $e \in U$.

LP Relaxation

Consider the above Set Cover IP *without the integrality constraint* on variables:

minimise

$$\sum_{S \in \mathcal{S}} c(S) x_S$$

subject to

$$\sum_{S \ni e} x_S \geq 1, \quad e \in U$$

$$x_S \geq 0, \quad S \in \mathcal{S}$$

(notice that $x_s \leq 1$ is inherently implied by minimisation of the objective)

The LP problem obtained in this way is called an **LP-relaxation** of the original IP problem.

Rounding

The idea of Rounding

In general, an optimal solution to LP-relaxation can be *better* than optimum for the original integer problem.

Since LP is in P, any LP-relaxation can be efficiently solved.

On the other hand, a solution to LP-relaxation may be non-integer (fractional) which does not, in general, represent a valid solution to the original IP formulation. For example, in Set Cover, it would mean taking "fractions" of sets.

Thus, to obtain a valid solution to original IP problem, the optimal solution to LP-relaxation can be *rounded* to the closest integer numbers.

Such obtained rounded solution to IP problem is usually only an *approximation* of integral optimum.

Example of solution of LP-relaxation

A solution to LP-relaxation can be better than a solution to integer program.

For, example, consider the following instance of the Set Cover Problem:

The universe consists of 3 elements: $U = \{e, f, g\}$, and the sets are as follows: $S_1 = \{e, f\}, S_2 = \{f, g\}, S_3 = \{g, e\}$, each has cost of 1. Any integer solution must contain at least 2 of the sets (with total cost of 2). However, the fractional solution of taking "half" of each set is a feasible solution to LP-relaxation (i.e. each elements is fully covered) and has better total cost of 3/2.

LP-rounding Algorithm for Set Cover

(Let f denote the max frequency of an element) Algorithm:
Find the optimum for the LP-relaxation of the primal
Take all the sets for which its decision variable is at least $1/f$

Theorem

The above integer solution is feasible (i.e. covers all elements) and is a f-factor approximation for Set Cover.

(Proof easy)

LP-rounding Algorithm for Set Cover

(Let f denote the max frequency of an element) Algorithm:
Find the optimum for the LP-relaxation of the primal
Take all the sets for which its decision variable is at least $1/f$

Theorem

The above integer solution is feasible (i.e. covers all elements) and is a f-factor approximation for Set Cover.

(Proof easy)(feasibility: uncovered element would contradict the value of f; factor guarantee: fractional solution is a lower bound for OPT and it is "multiplied" by f giving a feasible solution that implies factor guarantee)

Tight example

Let's generalise the view of Vertex Cover as Set Cover as follows. Consider viewing a set cover instance as a hypergraph such that each set corresponds to a vertex and each element corresponds to an incident hyperedge.

Let $V_1, \ldots, V_k$ are k sets of cardinality n (each) that are disjoint. The vertex set of the hypergraph is $V = V_1 \cup \cdots \cup V_k$ (nk vertices in total) and it has n^k hyperedges (each hyperedge picks one vertex from each V_i) (a complete k-partite hypergraph). Notice that f = k in such constructed instances. Each set has unit cost.

Optimal fractional solution takes 1/k of each of the nk vertices (sets of incident hyperedges) and has a total cost of n. Thus the rounding algorithm will pick all the nk sets so that the cost is k times larger.

(In addition, taking all sets of hyperedges corresponding with e.g. V_1 also gives the cost of n)

Randomised Rounding of Set Cover

Find optimal fractional solution to a given instance.
Then pick each of the sets with probability equal to the (fractional) value of the corresponding decision variable.
Repeating the process $O(logn)$ times and selecting a set if it is choosen at least once gives a Set Cover with high probability and can be verified in polynomial time.
The expected cost of the cover is
$O(logn) \cdot OPT_f \leq O(logn) \cdot OPT$ (expected value of sum is sum of expected values, etc.)

Randomised Rounding of Set Cover, cont.

Let x = p be an optimal solution to the linear program. Each set $S \in \mathcal{S}$ is picked with probability p_S. Denote the picked family of sets by C.

$$E[c(C)] = \sum_{S \in \mathcal{S}} Pr[S \text{ is picked}] \cdot c(S) = \sum_{S \in \mathcal{S}} p_s \cdot c(S) = OPT_f$$

The probability that an element $a \in U$ is covered is lower bounded by $1 - 1/e$.

Explanation: Assume that a occurs in k sets of $\mathcal{S}$. Denote the corresponding probabilities by $p_1, \ldots, p_k$. Because a is covered by fractional solution, $p_1 + \cdots + p_k \geq 1$. Under such condition, it is easy to check that the probability of covering an element a is minimised where all p_i are equal to $1/k$.

$$Pr[a \text{ is covered by } C] \geq 1 - (1 - \frac{1}{k})^k \geq 1 - \frac{1}{e}$$

Randomised Rounding of Set Cover, cont.

To complete the set cover, repeat the process $d log(n)$ times and compute the union of sets, denote C', where d is selected so that:

$$(\frac{1}{e})^{d\ log(n)} \leq \frac{1}{4n}$$

Hence,

$$Pr[a \text{ is not covered by } C'] \leq (\frac{1}{e})^{d\ log(n)} \leq \frac{1}{4n}$$

By summing over all elements in U:

$$Pr[C' \text{ is not a valid set cover}] \leq n \cdot \frac{1}{4n} \leq \frac{1}{4}$$

Randomised Rounding of Set Cover, cont.

By $E[c(C')] \leq OPT_f \cdot dlog(n)$ and Markov's inequality ($Pr[X \geq t] \leq E[X]/t$), we get that probability of "too expensive" cover is also "small":

$$Pr[c(C') \geq OPT_f \cdot 4dlog(d)] \leq \frac{1}{4}$$

Thus, the union of the two unwanted events is not higher than $1/2$ ($\leq 1/4 + 1/4$)
Hence, the probability of obtaining a "good" cover (feasible and cheap enough) is:

$$Pr[C' \, is \, a \, valid \, set \, cover \, and \, hascost \leq OPT_f \cdot 4d \cdot log(n)] \geq \frac{1}{2}$$

The fact whether C' is satisfies both conditions is polynomially verifiable, thus by sufficient repetition (expected number of repetitions is 2) we obtain the solution.

LP-duality

Well Characterised Problems

Consider the *decision* versions of cardinality vertex cover and maximum matching:

- is the size of the minimum vertex cover less or equal to k?
- is the size of the maximum matching greater or equal to l?

Both decision problems are in NP so there exist *"Yes" certificates*. Do there also exist "No" certificates for these problems (i.e.: do the problems are in co-NP class) ?

The problems that have at the same time "Yes" and "No" certificates are called *well characterised* and form a class $NP \cap co - NP$ that contains P class. It is an open problem whether the containment is strict (what is widely believed, though).

Min-Max Relations

As observed before, size of a maximum matching is *a lower bound* for the minimum vertex cover size.

Interestingly, for bipartite graphs, the *equality* holds:

$max_{matching\ M}|M| = min_{vertex\ cover\ C}|C|$
(Koenig-Egervary Theorem)

i.e. the "no" answer to the first question is equivalent to the existence of a $(k + 1)$-element matching. Analogousy, the negative answer to the second question is equivalent to the existence of a $(l - 1)$-element vertex cover.

Min-max relations and LP-duality

Min-max relations play a crucial role in designing approximation algorithms.

Often, they are implied by so-called LP-duality theorem.

In arbitrary graphs maximum matching can be strictly smaller than a minimum vertex cover. Examples:

- In a cycle of odd length $2l+1$, maximum matching size is l, but min vertex cover is l+1
- Petersen graph has a maximum *perfect matching* (of size 5) but still needs at least 6-element vertex cover.

Thus, in arbitrary graphs, minimum vertex cover (as NP-hard) does not have "No" certificate assuming $NP \neq co-NP$.

Approximate min-max relation for vertex cover

In any graph, the following approximate min-max relation holds:

$$max_{matching\ M}|M| \leq min_{vertex\ cover\ U}|U| \leq 2 \cdot (max_{matching\ M}|M|)$$

what is implied by the presented 2-approximation algorithm for vertex cover.

Factor-α approximate No certificate

No certificate for instances (I,k) of minimisation problems where $k < OPT(I)/\alpha$

Vertex cover has factor-2 approximate No certificate. (previous slide)

Exact Min-max relation for Odd Set Cover

The maximum matching problem, being in P, has “No” certificates in the form of *odd set cover*.

Theorem:
In any graph, the following min-max equality holds:

$$max_{matching\ M}|M| = min_{odd\ set\ cover\ C}\, w(C)$$

where, *odd set cover* C in a graph $G = (V, E)$ is a family of odd-cardinality subsets of V: $S_1, ..., S_k$ and a set of vertices $v_1, ..., v_l$ so that each edge in E has both ends in S_i for some i or is incident with v_j for some j. The cost of covering C is defined as: $w(C) = l + \sum_{i=1}^{k}(|S_i| - 1)/2$.

LP is well characterised

Consider the decision variant of the minimisation LP problem in standard form:
"Does there exist a feasible solution of objective value at most k?"
Obviously, any feasible solution is a "yes" certificate for some k in this case.
Interestingly, it is also possible to naturally construct "no" certificates.

Example of "yes" certificate

minimise

$$6x_1 + 2x_2 + 4x_3$$

subject to

$$2x_1 - x_2 + x_3 \geq 7$$

$$3x_1 + x_2 + 2x_3 \geq 5$$

$$x_1, x_2, x_3 \geq 0$$

for example: (4,1,1) is a "yes" certificate for $k = 30$

Negative certificates

minimise

$$6x_1 + 2x_2 + 4x_3$$

subject to

$$2x_1 - x_2 + x_3 \geq 7$$

$$3x_1 + x_2 + 2x_3 \geq 5$$

$$x_1, x_2, x_3 \geq 0$$

For example, notice that, for this particular program, objective value is higher than left side of the second constraint. Thus objective value must be at least 5. It is a *lower bound* on optimal solution, thus it is a "no" certificate. But 7 is even better lower bound in this case (first constraint).

But how to construct best negative certificate more systematically in general?

Finding the best lower bound

minimise

$$6x_1 + 2x_2 + 4x_3$$

subject to

$$2x_1 - x_2 + x_3 \geq 7$$

$$3x_1 + x_2 + 2x_3 \geq 5$$

$$x_1, x_2, x_3 \geq 0$$

In general, to obtain the best lower bound on objective, try to maximise the *linear combination* of constraints without exceeding the coefficients in the objective.

A Dual

minimise

$$6x_1 + 2x_2 + 4x_3$$

subject to

$$2x_1 - x_2 + x_3 \geq 7$$
$$3x_1 + x_2 + 2x_3 \geq 5$$
$$x_1, x_2, x_3 \geq 0$$

maximise

$$7y_1 + 5y_2$$

subject to

$$2y_1 + 3y_2 \leq 6$$
$$-y_1 + y_2 \leq 2$$
$$y_1 + 2y_2 \leq 4$$
$$y_1, y_2 \geq 0$$

The second problem is called a *dual* to the first problem. In this context the first one is called *primal*.

LP-Duality in general form

primal:
minimise

$$\sum_{j=1}^{n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \ \ i = 1, \ldots, m$$

$$x_j \geq 0, \ \ j = 1, \ldots, n$$

dual:
maximise

$$\sum_{i=1}^{m} b_i y_i$$

subject to

$$\sum_{i=1}^{m} a_{ij} y_i \leq c_j, \ \ j = 1, \ldots, n$$

$$y_i \geq 0, \ \ i = 1, \ldots, m$$

Basic Properties of Duality

- minimisation $\leftrightarrow$ maximisation
- number of constraints $\leftrightarrow$ number of variables
- $\geq \leftrightarrow \leq$

Furthermore, a dual of dual is primal.

Weak Duality

Any feasible solution to the dual is a lower bound of any feasible solution of the primal.

Theorem

(Weak Duality) If x is a feasible primal solution and y is a feasible solution to the dual, then

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{i=1}^{m} b_i y_i$$

Proof:

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{j=1}^{n} (\sum_{i=1}^{m} a_{ij} y_i) x_j = \sum_{i=1}^{m} (\sum_{j=1}^{n} a_{ij} x_j) y_i \geq \sum_{i=1}^{m} b_i y_j$$

(by feasibility of x,y and their non-negativity)

(Strong) Duality Theorem

Theorem

Primal program has finite optimum iff dual program has finite optimum. Furthermore, if primal and dual are feasible, their optimal values are ***the same****. More precisely, for any optimal solutions x^*, y^* to primal and dual, respectively the following holds:*

$$\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$$

Corollary

The LP-duality theorem is a min-max relation, thus it is well-characterised. Feasible solutions to primal (dual) provide "yes" ("no") certificates to the question "Is the optimum at most α?". Thus, LP is in $NP \cap co-NP$.

Complementary Slackness Conditions

A corollary from the strong duality is as follows:

Theorem

If x and y are feasible solutions to primal and dual, respectively, then x and y are both optimal iff all the following conditions hold:

(Primal complementary slackness):

For each $1 \leq j \leq n$ *either* $x_j = 0$ *or* $\sum_{i=1}^{m} a_{ij} y_i = c_j$

(Dual complementary slackness):

For each $1 \leq i \leq m$ *either* $y_i = 0$ *or* $\sum_{j=1}^{n} a_{ij} x_j = b_i$

Relation of Relaxed and Integer Solutions

$0 \leq$ dual fractional solutions $\leq OPT_f \leq$ primal fractional solutions $\leq OPT \leq$ primal integer solutions $\leq \infty$

Integrality Gap

For a minimisation IP problem Π let $OPT(I)$ denote its optimal objective value and $OPT_f(I)$ the optimal objective value for its LP-relaxation. The following value:

$$\sup_{I \in \Pi} \frac{OPT(I)}{OPT_f(I)}$$

is called *the integrality gap* of Π

(for a maximisation problem, we take infimum instead of supremum)

If optimum of LP-relaxation is integer, we call it an *exact relaxation* (integrality gap is equal to 1 in such case)

Boundedness

Theorem

For primal (P) and dual (D) programs one of the following cases must hold:

- *both P and D are feasible*
- *P is infeasible and D is unbounded*
- *P is unbounded and D is infeasible*
- *both P and D are infeasible*

Constructing Dual Program for Set Cover

Relaxation of Set Cover (Primal):

minimise

$$\sum_{S \in \mathcal{S}} c(S) x_S$$

subject to

$$\sum_{S \ni e} x_S \geq 1, \quad e \in U$$

$$x_S \geq 0, \quad S \in \mathcal{S}$$

Dual Program for Set Cover?

Dual Program for Set Cover?

maximise

$$\sum_{e \in U} y_e$$

subject to

$$\sum_{e: e \in S} y_e \leq c(S), \;\; S \in \mathcal{S}$$

$$y_e \geq 0, \;\; e \in U$$

Interpretting the Dual for Set Cover

maximise

$$\sum_{e \in U} y_e$$

subject to

$$\sum_{e: e \in S} y_e \leq c(S), S \in \mathcal{S}$$

$$y_e \geq 0, e \in U$$

It may be viewed as "packing" the elements of the universe U in order to maximise the "total amount packed" subject to the constraint that no set is "overpacked" (i.e. the amount packed into elements does not exceed the cost of the set).

Covering-packing Pair

Any pair of linear programs where the coefficients in the constraint matrix, objective function and right-hand side of inequalities are non-negative is called *covering-packing pair* of LP programs

Primal-Dual Schema

Primal-Dual Schema for Exact Algorithms

Primal-dual is a general technique for designing combinatorial approximation algorithms based on LP-duality theory.

This approach was first used in the context of exact algorithms (matching, max flow, shortest paths, etc.).

It is based on the fact that optimal solutions to linear programs satisfy complementary slackness conditions.

In case of exact algorithms it starts with an initial feasible primal and dual solutions and iteratively satisfies complementary slackness conditions. When all the conditions are satisfied, both solutions are optimal.

The invariant of iterations is that primal is always integral.

Primal-Dual Schema for NP-hard Problems

In LP-relaxation of an NP-hard problem the optimum is not integral, in general.

The primal-dual schema for approximation algorithms is usually adapted so that some complementary slackness conditions are satisfied and other are *relaxed*.

The "relaxed" conditions are controlled by rational factors $\alpha, \beta \geq 1$; i.e. if primals (duals) are ensured then $\alpha = 1$ ($\beta = 1$).

Primal and Dual Complementary Slackness Conditions

Primal complementary slackness conditions:
Let $\alpha \geq 1$
for each $1 \leq j \leq n$: either $x_j = 0$ or $c_j/\alpha \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$

Dual complementary slackness conditions:
Let $\beta \geq 1$
For each $1 \leq i \leq m$: either $y_i = 0$ or $b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta \cdot b_i$

Theorem

If x and y are primal and dual feasible solutions satisfying the above conditions then:

$$\sum_{j=1}^{n} c_j x_j \leq \alpha \cdot \beta \cdot \sum_{i=1}^{m} b_i y_i$$

Primal-Dual Algorithm

It starts with a primal infeasible solution x and dual feasible y solution (usually trivial: x=y=0).

Iteratively, feasibility of the primal and “optimality” of the dual are improved. At the end primal is feasible and all conditions (for some α, β, specified at the beginning) are satisfied.

Primal is always improved integrally. The cost of the dual is always used as a lower bound on OPT, and finally the $\alpha \cdot \beta$ approximation factor is guaranteed.

The alternating iterative “improvements” to primal and dual are connected: i.e. they suggest each other in an alternating manner.

Primal-dual Schema for Set Cover

$\alpha = 1, \ \beta = f$

Primal conditions:

$$\forall S \in \mathcal{S} : x_S \neq 0 \Rightarrow \sum_{e:\, e \in S} y_e = c(S)$$

(pick only tight sets; do not overpack any set)

Dual conditions:

$$\forall e : y_e \neq 0 \Rightarrow \sum_{S:\, e \in S} x_S \leq f$$

(since each element is in at most f sets, it is satisfied trivially)

Primal-Dual Algorithm for Set Cover

Initialize x=0, y=0
Until all elemenst are covered:
pick an uncovered element e, raise y_e until some set becomes tight
pick all thight sets in the cover and update s
declare all the elements in these sets as "covered"
Output the set cover for x

The above is a f-approximation algorithm for Set Cover.
Proof: all elements are covered, no overpacked sets. Primal and dual will be both feasible and satisfy the relaxed compl. slack. cond. with $\alpha = 1, \beta = f$ that makes f-approximation guarantee.

Tight Example

$\mathcal{S}$ consists of n-1 sets of unit cost: $\{e_i, e_n\}$ for $1 \leq i \leq n-1$ and one set $\{e_1, \ldots, e_{n+1}\}$ with cost of $1+\epsilon$ $(\epsilon > 0)$. f=n (e_n is contained in n sets).
Assume, the algorithm first raises y_{e_n}. When y_{e_n} is raised to 1 all "pair" sets become tight and are all picked to cover first n elements. Next, $y_{e_{n+1}}$ is raised to ϵ and the "big" set becomes tight. Thus, the found solution has the cost of $n+\epsilon$, while the optimal solution (single set) has cost of $1+\epsilon$.

Dual Fitting

Dual Fitting

Dual fitting is a LP-duality-based technique to support analysis of combinatorial algorithms.

Assuming the minimisation problem, it helps analysing a combinatorial algorithm using LP-relaxation of the problem and its dual.

It aims to show that primal integral solution has objective function not higher than dual, but the dual is infeasible. By finding an appropriate factor, the dual, though, is divided by the factor to become feasible.

Thus the scaled-down dual is a lower bound for OPT.

Example: Dual Fitting Analysis of Greedy Algorithm for Set Cover

Remind the O(log(n))-factor greedy approximation algorithm for Set Cover.

Dual fitting provides an alternative analysis of that algorithm.

The algorithm defines dual variables $price(e)$ for the elements. The integral cover found by the algorithm has objective value that is upper-bounded by that dual solution, and the dual is infeasible.

However, by scaling down the dual: $y_e = price(e)/H_n$ we obtain a feasible dual solution that is lower bound for OPT. Thus, the algorithm can be now viewed as based on the lower-bounding technique (proof similar as previously)

Greedy Algorithm for SC as lower-bounding-based

Remind the 3 questions mentioned previously. In case of the greedy algorithm, the answer to the question 1 was answered negatively (tight example was provided). (also, the third question has negative answer)

We will now show that approximation guarantee based on that lower-bounding technique cannot be significantly improved: i.e. the integrality gap has size O(log n)

O(log n) Lower Bound for Set Cover Integrality Gap

Consider the following instance of Set Cover. Let $n = 2^k - 1$, $k \in N^+$, $U = \{e_1, \ldots, e_n\}$. For any $1 \leq i \leq n$, consider that i is written as a binary k-bit number. It can be viewed as $k-$dimensional vector over Z_2. Denote it by $\mathbf{i}$. For $1 \leq i \leq n$ define set $S_i = \{e_j | \ \mathbf{i} \cdot \mathbf{j} = 1\}$ (dot product over Z_2). Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ and each set is of unit cost.

How many elements does each S_i contain?

O(log n) Lower Bound for Set Cover Integrality Gap

Consider the following instance of Set Cover. Let $n = 2^k - 1$, $k \in N^+$, $U = \{e_1, \ldots, e_n\}$. For any $1 \leq i \leq n$, consider that i is written as a binary k-bit number. It can be viewed as $k-$dimensional vector over Z_2. Denote it by **i**. For $1 \leq i \leq n$ define set $S_i = \{e_j | \mathbf{i} \cdot \mathbf{j} = 1\}$ (dot product over Z_2). Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ and each set is of unit cost.

How many elements does each S_i contain? $2^{k-1} = (n+1)/2$ (why?)

O(log n) Lower Bound for Set Cover Integrality Gap

Consider the following instance of Set Cover. Let $n = 2^k - 1$, $k \in N^+$, $U = \{e_1, \ldots, e_n\}$. For any $1 \leq i \leq n$, consider that i is written as a binary k-bit number. It can be viewed as $k-$dimensional vector over Z_2. Denote it by $\mathbf{i}$. For $1 \leq i \leq n$ define set $S_i = \{e_j | \ \mathbf{i} \cdot \mathbf{j} = 1\}$ (dot product over Z_2). Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ and each set is of unit cost.

How many elements does each S_i contain? $2^{k-1} = (n+1)/2$ (why?) (the number of odd subsets of k)

O(log n) Lower Bound for Set Cover Integrality Gap

Consider the following instance of Set Cover. Let $n = 2^k - 1$, $k \in N^+$, $U = \{e_1, \ldots, e_n\}$. For any $1 \leq i \leq n$, consider that i is written as a binary k-bit number. It can be viewed as $k-$dimensional vector over Z_2. Denote it by **i**. For $1 \leq i \leq n$ define set $S_i = \{e_j | \mathbf{i} \cdot \mathbf{j} = 1\}$ (dot product over Z_2). Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ and each set is of unit cost.

How many elements does each S_i contain? $2^{k-1} = (n+1)/2$ (why?) (the number of odd subsets of k)

Each element is contained in $(n+1)/2$ sets (why?)

O(log n) Lower Bound for Set Cover Integrality Gap

Consider the following instance of Set Cover. Let $n = 2^k - 1$, $k \in N^+$, $U = \{e_1, \ldots, e_n\}$. For any $1 \leq i \leq n$, consider that i is written as a binary k-bit number. It can be viewed as $k-$dimensional vector over Z_2. Denote it by $\mathbf{i}$. For $1 \leq i \leq n$ define set $S_i = \{e_j | \, \mathbf{i}{\cdot}\mathbf{j} = 1\}$ (dot product over Z_2). Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ and each set is of unit cost.

How many elements does each S_i contain? $2^{k-1} = (n+1)/2$ (why?) (the number of odd subsets of k)

Each element is contained in $(n+1)/2$ sets (why?) (symmetry i-j of the definition)

Thus, $x_i = 2/(n+1)$ (for each i) is a *fractional* solution (to LP-relaxation of Set Cover). The total cost of this solution is $2n/(n+1)$.

O(log n) Lower Bound on Set Cover Integrality Gap, cont.

(Total cost of that fractional solution is $2n/(n+1)$)
But any integer solution has to pick at least k sets. Assume a union of p sets where $p < k$. Let $i_1, \dots, i_p$ are indices of these p sets, and let A be a $p \times k$ matrix over Z_2 whose rows are the vectors $\mathbf{i}_1, \dots, \mathbf{i}_p$.
$rank(A) < k$ thus the dimension of its null space is positive, so it contains a non-zero vector. Call it $\mathbf{j}$. Because $A \cdot \mathbf{j} = 0$, the element e_j is not covered by any of the p selected sets. Contradiction. (i.e. $p \leq k$)
Thus, any integral cover has total cost of at least $k = log_2(n+1)$. Thus the lower bound for the integrality gap is $(n+1)/2n \cdot log_2(n+1) > 1/2 \cdot log_2 n = O(logn)$

It can be also shown that Integrality gap of LP-relaxation of Set Cover is upper bounded by H_n.
Thus the example is a tight bound.

Example of less obvious approximation factor value

Most of the examples presented before have simple algorithms (usually greedy) and factor that is a “simple” rational number (like 2 or 3/2, etc.)

Simple, greedy algorithms do not always lead to such situations.

Maximum Coverage

Maximum Coverage

Given a universal set U of elements with weights ($w : U \rightarrow Q^+$), a family F of subsets of U and $k \in N^+$ find a subset of F so that the sum of weights of covered elements of U is maximised.

It is NP-hard.
Greedy algorithm?

Maximum Coverage

Given a universal set U of elements with weights ($w : U \rightarrow Q^+$), a family F of subsets of U and $k \in N^+$ find a subset of F so that the sum of weights of covered elements of U is maximised.

It is NP-hard.
Greedy algorithm?

In iterations, take the set from F that maximises the total weight of newly covered elements.

Greedy algorithm has 0.683 factor

Greedy algorithm has 0.683 factor

Theorem

The greedy algorithm for Maximum Coverage has $1 - (1 - \frac{1}{k})^k > (1 - \frac{1}{e})$ *approximation factor.*

Lemma 1

Let's denote the i-th element picked by the greedy algorithm as G_i

$$w(\cup_{i=1}^{l} G_i) - w(\cup_{i=1}^{l-1} G_i) \geq \frac{1}{k}(OPT - w(\cup_{i=1}^{l-1} G_i))$$

for $0 < l \leq k$

Proof: The k sets from optimal solution can cover the elements not covered by the first l-1 "greedy" sets (their weight is $OPT - w(\cup_{i=1}^{l-1} G_i)$. Thus, by the pigeonhole principle, one of the k sets in the optimal solution must cover at least $\frac{1}{k}(OPT - w(\cup_{i=1}^{l-1} G_i))$ of uncovered elements. But G_l is picked as covering most heavy additional elements, so it also does.

Lemma 2

$$w(\cup_{i=1}^{l} G_i) \geq [(1-(1-\frac{1}{k})^l)] \cdot OPT$$

for $0 < l \leq k$

Proof (by induction): For l=1 the result holds: $w(G_l) \geq OPT/k$. For l+1:

$$w(\cup_{i=1}^{l+1} G_i) = w(\cup_{i=1}^{l} G_i) + w(\cup_{i=1}^{l+1} G_i) - w(\cup_{i=1}^{l} G_i)$$

$$\geq w(\cup_{i=1}^{l} G_i) + \frac{1}{k}(OPT - w(\cup_{i=1}^{l} G_i))$$

$$= (1-1/k)w(\cup_{i=1}^{l} G_i) + 1/k(OPT)$$

$$\geq (1-\frac{1}{k})(1-(1-\frac{1}{k})^l) \cdot OPT + \frac{OPT}{k}$$

$$= (1-(1-\frac{1}{k})^{l+1}) \cdot OPT$$

Thank you for attention