Hardness and Approximation Algorithms

Ana Paula Tomás

LEIC - Desenho de Algoritmos
Universidade do Porto

May 2022

Efficient Algorithms

From textbook “Algorithms”, by Jeff Erickson, chapter 12.
https://jeffe.cs.illinois.edu/teaching/algorithms/

- A minimal requirement for an **algorithm** to be considered “efficient” is that its running time is bounded by a polynomial function of the input size: $O(n^c)$ for some constant c, where n is the size of the input.
- Researchers recognized early on that not all problems can be solved this quickly, but had a hard time figuring out exactly which ones could and which ones couldn’t.
- There are several so-called **NP-hard problems**, which most people believe cannot be solved in polynomial time, even though nobody can prove a super-polynomial lower bound.

P versus NP

From textbook “Algorithms”, by Jeff Erickson, chapter 12.
https://jeffe.cs.illinois.edu/teaching/algorithms/

A **decision problem** is a problem whose output is a single boolean value: YES or NO.

Three classes of decision problems:

- **P** is the set of decision problems that can be solved in polynomial time. Intuitively, P is the set of problems that can be solved quickly.

- **NP** is the set of decision problems with the following property: If the answer is YES, then there is a proof of this fact that can be checked in polynomial time. Intuitively, NP is the set of decision problems where we can verify a YES answer quickly **if we have the solution** in front of us.

- **co-NP** is essentially the opposite of NP. If the answer to a problem in co-NP is NO, then there is a proof of this fact that can be checked in polynomial time.

Examples of Decision Problems in NP

- **SAT**: Given a CNF formula $\Phi(x_1, \ldots, x_n) = C_1 \wedge \ldots \wedge C_m$, is Φ satisfiable, i.e., is there a truth assignment that satisfies all clauses?
- **CIRCUIT-SAT**: Given a combinational circuit built from AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?
- **HAMILTONEAN CYCLE**: Given an undirected graph $G = (V, E)$, does G contain an Hamiltonean cycle (a cycle that visits all nodes exactly once)?
- **EULERIAN CYCLE**: Given an undirected graph $G = (V, E)$, does G contain an Eulerian cycle (a cycle that visits all edges exactly once)?
- **PARTITION**: Given an set $S = \{a_1, a_2, \ldots, a_n\}$ of n positive integers, is there a set $A \subset S$ such that $\sum_{x \in A} x = \sum_{y \in S \setminus A} x$?
- **PLANAR 3-COLORING**: Given a planar undirected graph $G = (V, E)$, is there a way of coloring the vertices of G such that no two adjacent vertices are of the same color and using at most three colors?

Open Problems: P = NP? NP = co-NP?

Polynomial reductions

A problem B can be *polynomially reduced to* A, or $B \leq_p A$, if, given access to a solution for A, we can solve B in polynomial time using polynomially many calls to this solution for A.

NP-completeness

A problem L is **NP-hard** if, for all problems L' in the class NP, we have $L' \leq_p L$. If, in addition, L belongs to NP, then we say that L is NP-complete.

Cook-Levin Theorem

CIRCUIT-SAT is NP-complete. SAT is NP-complete.

If SAT could be solved in polynomial time by a **deterministic Turing machine**, then all problems in NP could be solved in polynomial time, and P = NP.

Open Problems: P = NP? NP = co-NP?

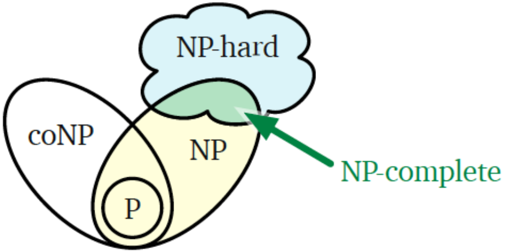


Figure 12.4. More of what we *think* the world looks like.

https://jeffe.cs.illinois.edu/teaching/algorithms/book/12-nphard.pdf

Examples of Decision Problems in NP

Examples of optimization problems whose decision version is in NP

- **SHORTEST PATH**: Given a weighted graph $G = (V, E, d)$, with $d(e) \in \mathbb{Z}^+$, for all $e \in E$, two nodes $s, t \in V$, and $k \in \mathbb{Z}^+$, is there a path γ from s to t with $d(\gamma) \leq k$? The optimization problem asks for shortest path.
- **LONGEST PATH**: Given a weighted graph $G = (V, E, d)$, with $d(e) \in \mathbb{Z}^+$, for all $e \in E$, two nodes $s, t \in V$, and $k \in \mathbb{Z}^+$, is there a path γ from s to t with $d(\gamma) \geq k$? The optimization problem asks for longest path.
- **TSP** (travelling salesperson problem): Given a **complete** weighted graph $G = (V, E, d)$, with $d(e) \in \mathbb{Z}^+$, for all $e \in E$, and $k \in \mathbb{Z}^+$, is there a hamiltonean cycle γ with $d(\gamma) \leq k$? Optimization version asks for shortest hamiltonean cycle.
- **MAXIMUM INDEPENDENT SET**: Given an undirected graph $G = (V, E)$ and $k \in \mathbb{Z}^+$, does G contain an independent set I with $|I| \geq k$, i.e., a set $I \subseteq V$ such that no two nodes in I are linked by an edge in G?

Examples of Decision Problems in NP

Examples of optimization problems whose decision version is in NP

- **BIN PACKING**: Given a set $S = \{a_1, \ldots, a_n\}$ of items, with $0 < a_i < 1$, for all i, and $k \in \mathbb{Z}^+$, is it possible to pack the items in at most k bins with capacity 1?
- **SET COVER**: Given $S = \{a_1, \ldots, a_n\}$ and a family $\mathcal{F}$ of subsets of S, is there $C \subseteq \mathcal{F}$ such that $|C| \leq k$ and $S = \bigcup_{X \in C} X$?
- **VERTEX COVER**: Given an undirected graph $G = (V, E)$ and $k \in \mathbb{Z}^+$, is there a subset C of V such that $|C| \leq k$ and each edge in V is incident to a vertex in C?
- **KNAPSACK**: Given a set S of n items, each with a weight w_i and a value v_i, the capacity W of the knapsack, and a value k, is there a subset I of S such that $\sum_{i \in I} w_i \leq W$ and $\sum_{i \in I} v_i \geq k$?

P versus NP

If a given boolean circuit is satisfiable, then *any set of n input values that produces output* TRUE is a **proof** that the circuit is satisfiable.

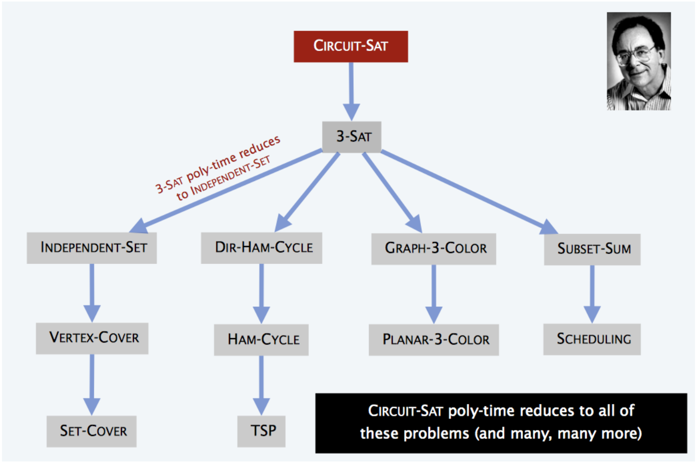
We **can check the proof** by evaluating the circuit in polynomial (actually, linear) time using depth-first-search.

But nobody knows how to **solve** it than trying all 2^n possible inputs to the circuit by **brute force**, which requires **exponential time**.

It is widely believed that circuit satisfiability is not in P or in co-NP, but nobody actually knows.

Every decision problem in P is also in NP. If a problem is in P, we can verify YES answers in polynomial time recomputing the answer from scratch! Similarly, every problem in P is also in co-NP.

Karp

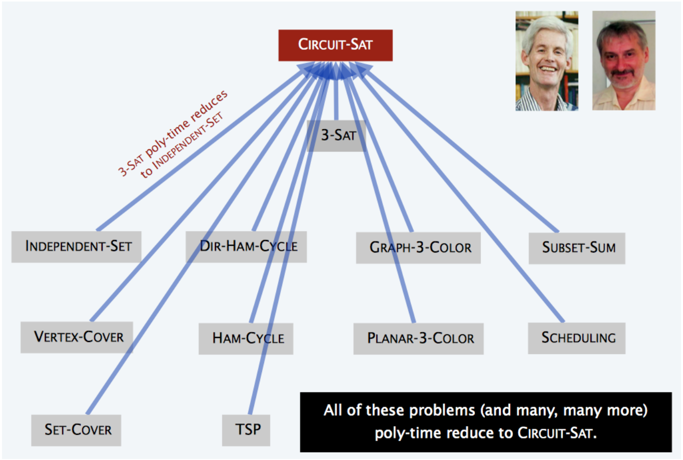


CIRCUIT-SAT poly-time reduces to all of these problems (and many, many more)

Slide by K.Wayne,

https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/08IntractabilityII.pdf

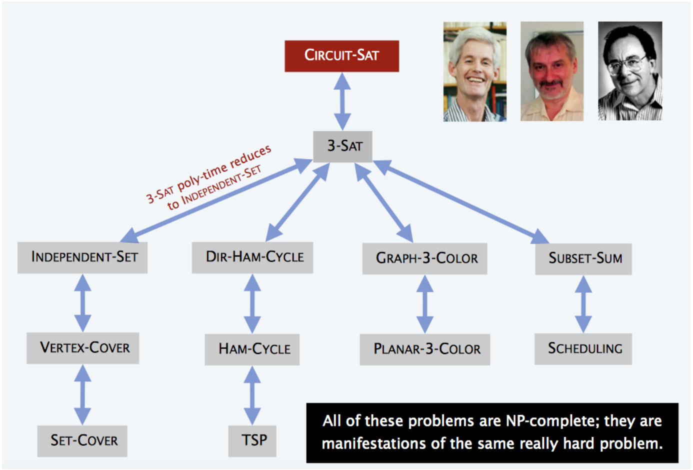
Cook-Levin



Slide by K.Wayne,

`https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/08IntractabilityII.pdf`

Karp & Cook-Levin


CIRCUIT-SAT
3-SAT
3-SAT poly-time reduces to INDEPENDENT-SET
INDEPENDENT-SET
DIR-HAM-CYCLE
GRAPH-3-COLOR
SUBSET-SUM
VERTEX-COVER
HAM-CYCLE
PLANAR-3-COLOR
SCHEDULING
SET-COVER
TSP
All of these problems are NP-complete; they are manifestations of the same really hard problem.

Slide by K.Wayne,

https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/08IntractabilityII.pdf

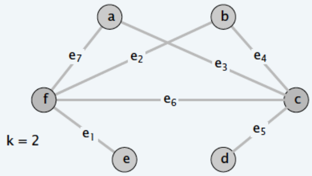
Reduction from Vertex-Cover to Set-Cover

Theorem. VERTEX-COVER $\leq_P$ SET-COVER.

Pf. Given a VERTEX-COVER instance $G = (V, E)$ and k, we construct a SET-COVER instance (U, S, k) that has a set cover of size k iff G has a vertex cover of size k.

Construction.

- Universe $U = E$.
- Include one subset for each node $v \in V$: $S_v = \{e \in E : e \text{ incident to } v\}$.



vertex cover instance
(k = 2)

$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$

$S_a = \{ 3, 7 \}$	$S_b = \{ 2, 4 \}$
$S_c = \{ 3, 4, 5, 6 \}$	$S_d = \{ 5 \}$
$S_e = \{ 1 \}$	$S_f = \{ 1, 2, 6, 7 \}$

set cover instance
(k = 2)

Slide by K.Wayne,

https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/08IntractabilityI.pdf

Reduction from 3-SAT to MAX INDEPENDENT SET

Idea:

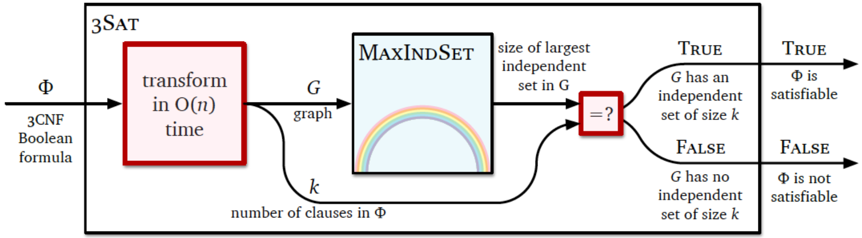
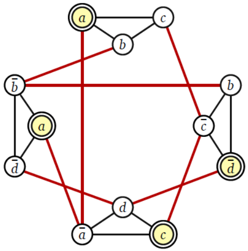


Figure 12.7. A polynomial-time reduction from 3SAT to MAXINDSET.

https://jeffe.cs.illinois.edu/teaching/algorithms/book/12-nphard.pdf

Reduction from 3-SAT to MAX INDEPENDENT SET

For each clause $l_1 \vee l_2 \vee l_3$, the graph G has 3 new nodes, labeled l_1, l_2 and l_3, linked by edges. Any two nodes that contain a variable x and its negation $\neg x$ will be linked by an edge also. Check `https://jeffe.cs.illinois.edu/teaching/algorithms/book/12-nphard.pdf`



$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$

Figure 12.8. A graph derived from the satisfiable 3CNF formula with 4 clauses, and an independent set of size 4.

The size of any independent set of G is at most k, being k the number of clauses of Φ.

***G* has a maximum independent set of size *k* iff Φ is satisfiable.**

Reduction from HAMILTONEAN CYCLE to TSP

Given an undirected graph $G = (V, E)$, we construct a **complete weighted** undirected graph $G' = (V, E', d)$ with $E' = E \cup \{\langle u, v\rangle \mid u \neq v, \langle u, v\rangle \notin E$, and $d(e) = 1$, for all $e \in E$ and $d(e) = |V| + 1$, for all $e \in E' \setminus E$.

- G' is a complete graph.
- G' can be obtained from G in polynomial time. In fact, in quadractic time in the number of vertices $O(|V|^2$.
- By construction, G contains an Hamiltonean cycle if and only if G' contains an Hamiltonean cycle γ such that $d(\gamma) = |V|$.
- If we have a polynomial time algorithm for TSP, we can ask whether G' contains an Hamiltonean cycle γ with $d(\gamma) \leq |V|$. If the answer is YES, G has an Hamiltonean cycle. Otherwise, it does not. So, we can use that algorithm to decide HAMILTONEAN CYCLE.
- Therefore, TSP is NP-hard since it is known that HAMILTONEAN CYCLE is NP-complete.

Fast. Cheap. Reliable. Choose two.

An old engineering slogan says, "Fast. Cheap. Reliable. Choose two." Similarly, if $P \neq NP$, we can't simultaneously have algorithms that

- find optimal solutions
- in polynomial time
- for any instance

At least one of these requirements must be relaxed in any approach to dealing with an NP-hard optimisation problem.

- David P. Williamson & David B. Shmoys. **The Design of Approximation Algorithms**. Cambridge University Press, 2010.
- Vijay Vazirani. **Approximation Algorithms**. Springer, 2001.

Polynomial-time Approximation Algorithms

The goal. Finding near-optimal (or good enough) solutions *efficiently* (in polynomial-time).

- This is not about "heuristic algorithms" and empirical analysis of their performance.
- The **theory of approximation algorithms:**
  - provable performance in the worst-case;
  - inapproximability.
- How close to optimum can we get in polynomial time? We will see some inapproximability results.

Recall: Hard Variants of Stable Marriage

D. F. Manlove, R. Irving, K. Iwama, S. Miyazaki, Y. Morita, Hard variants of stable marriage, Theoretical Computer Science*, 276: 261-279, 2002.*

. . . *Here, we present the first comprehensive study of variants of the problem in which the preference lists of the participants are not necessarily complete and not necessarily totally ordered. We show that, under surprisingly restrictive assumptions, a number of these variants are hard, and hard to approximate. The key observation is that, in contrast to the case where preference lists are complete or strictly ordered (or both), a given problem instance may admit stable matchings of different sizes. (. . .) Examples of problems that are hard:*

- *Finding a stable matching of maximum or minimum size; determining whether a pair is stable, even if indifference takes the form of ties on one side only, the ties are at the tails of the lists, there is at most one tie per list, and each tie has length two, and*
- *Finding or approximating, both "an egalitarian" and a "minimum regret" stable matching.*

Approximation Algorithms

- **NPO problem**: an optimisation problem such that the associated decision problem is in NP.

- The **approximation ratio** of an algorithm $\mathcal{A}$ for a **minimization problem**

$$\alpha_{\mathcal{A}} = \max_{I} \frac{\mathcal{A}(I)}{OPT(I)}$$

  So, $\mathcal{A}(I) \leq \alpha_{\mathcal{A}} OPT(I)$, where $\mathcal{A}(I)$ is the value of the solution $\mathcal{A}$ returns for instance I.

- For a **maximization problem**, $\alpha_{\mathcal{A}} = \max_I \frac{OPT(I)}{\mathcal{A}(I)}$, so $\mathcal{A}(I) \geq \frac{1}{\alpha_{\mathcal{A}}} OPT(I)$.
- By this definition, $\alpha_{\mathcal{A}} \geq 1$ even for maximization problems.
- On an input of size n, the ratio $\alpha_{\mathcal{A}}$ can be a function $\alpha_{\mathcal{A}}(n)$.
- If the function is constant, i.e., does not depend on n, then $\mathcal{A}$ is a constant factor approximation algorithm.

The class APX and APX-hard problems

- **APX** is the class of NPO problems for which there are constant factor *polynomial time* approximation algorithms.
  - MINIMUM CARDINALITY VERTEX COVER is in APX (2-approximable)
  - METRIC TSP is in APX. (2-approximable, 3/2-approximable)
  - GENERIC TSP is not in APX (unless P=NP).
  - BIN PACKING is in APX (2-approximable, 3/2-approximable)

- An NPO problem is **APX-hard** if there is a constant $\epsilon > 0$ such that an approximation ratio of $1 + \epsilon$ cannot be guaranteed by any polynomial-time algorithm, **unless P = NP**.
  - MINIMUM CARDINALITY VERTEX COVER is APX-hard.
  - BIN PACKING is APX-hard.

Approximation Schemes

An **approximation scheme** for an optimization problem Π is a family of $(1+\varepsilon)$-approximation algorithms A_ε for problem Π, over all $0 < \varepsilon < 1$.

- **PTAS Polynomial Time Approximation Scheme**
  A polynomial time approximation scheme for Π is an approximation scheme such that the time complexity of A_ε is polynomial in the input size, for all ε. (the time complexity can be exponential in $1/\varepsilon$)
  (e.g. EUCLIDEAN TSP)

- **FPTAS Fully Polynomial Time Approximation Scheme**
  A polynomial time approximation scheme for Π is an approximation scheme such that the time complexity of A_ε is polynomial in the input size and also polynomial in $1/\varepsilon$, for all ε.
  (e.g. KNAPSACK PROBLEM)

[Arora et al., FOCS'92] **If there is a PTAS for some APX-hard problem, P=NP.**

MAX 3-SAT: Maximum Satisfiability Problem

MAX 3-SAT Given a 3-SAT formula $\Phi(x_1, \ldots, x_n) = C_1 \wedge \ldots \wedge C_m$, find a truth assignment that satisfies as many clauses as possible.

MAX 3-SAT is NP-hard because 3-SAT is NP-complete.

Johnson's Randomized Algorithm for MAX 3-SAT

A 7/8-approximation algorithm that runs in expected polynomial time: **Repeatedly generate random truth assignments until one of them satisfies at least 7m/8 clauses.**

Inapproximabilty of MAX E3SAT (Håstad, 1997)

MAX E3SAT, the version of MAX SAT in which each clause is of length **exactly three**, cannot be approximated in polynomial time to within a ratio greater than 7/8, unless P=NP.

J. Håstad. Some optimal inapproximability results. In Proc. 28th Annual ACM Symp. on Theory of Computing, pp 1–10, 1997.

2-approximation algorithm for VERTEX COVER

VERTEXCOVER asks for a *minimum cardinality vertex-cover* of a given undirected graph $G = (V, E)$. A **vertex-cover** of G is a subset $S \subseteq V$ such that for each edge $(u, v) \in E$, either $u \in S$ or $v \in S$, or both.

- Given G and $k \in \mathbb{N}$ as input, deciding if G has a vertex-cover S of size $|S| \leq k$ is a well-known **NP-complete** problem.
- VERTEXCOVER is **APX-complete** (i.e., APX-hard and belongs to APX).

2-approximation for VERTEXCOVER

```
$S := \emptyset$
while ($E \neq \emptyset$) do
    remove an edge $e = (u, v)$ from $E$
    remove all edges incident to $u$ or $v$
    $S := S \cup \{u, v\}$
return $S$
```

Proof: The algorithm yields a vertex-cover S in poly-time. The selected edges e do not share endpoints (they form a matching M of G). If $S^\star$ is an optimal vertex-cover, $|S^\star| \geq |M| = |S|/2$, since each edge in M is covered by a distinct vertex of $S^\star$. Thus, $|S| \leq 2|S^\star|$. □

2-approximation for VERTEX COVER (by LP rounding)

Consider VERTEXCOVER as a *boolean linear programming problem*.

$$\text{minimize } \sum_{v \in V} x_v$$
$$\begin{cases} x_u + x_v \geq 1, \text{ for all } (u,v) \in E \\ x_v \in \{0,1\}, \text{ for all } v \in V \end{cases}$$

- It is known that its **linear relaxation**, i.e., the problem we obtain if we replace the domain constraint $x_v \in \{0,1\}$ by $x_v \in [0,1]$, for all v, can be solved in polynomial time. Let $x^\star$ be its optimal solution.
- The boolean solution given by $x_v = 1$ if $x_v^\star \geq 1/2$ and $x_v = 0$ if $x_v^\star < 1/2$, for all $v \in V$, is a **feasible solution to VERTEXCOVER**. In fact, for each edge (u,v), either $x_u^\star \geq 1/2$ or $x_v^\star \geq 1/2$ (otherwise, $x_u^\star + x_v^\star \geq 1$ would be violated). So, $S = \{v \in V \mid x_v = 1\}$ is a vertex-cover.
- If $S^\star$ is a minimum vertex-cover, then $|S| \leq 2|S^\star|$. In fact, by construction $|S| \leq 2\sum_{v \in V} x_v^\star$, and $\sum_{v \in V} x_v^\star \leq |S^\star|$ because the optimal solution of the relaxation cannot be worse than the value of any other of its solutions (and, therefore, of the boolean solution induced by $S^\star$).

Minimum vertex-cover is APX-hard

Some known inapproximability bounds for minimum vertex cover on graphs:

- It is hard to approximate to within $2 - \varepsilon$, for any constant $\varepsilon > 0$, if the *unique games conjecture* is true (S.Khot & O.Regev, 2008).
- Håstad (J.ACM, 2001) showed that it is NP-hard to approximate within constant factors less than 7/6. This factor was improved by Dinur and Safra (STOC'2002) to $10\sqrt{5} - 21 \approx 1.36$.
- If the graph has degree bounded by 3, it cannot be approximated within $100/99 - \epsilon$, for $\epsilon > 0$, unless P=NP; $53/52 - \epsilon$ if the degree is bounded by 4. [Chlebík & Chlebíková, FCT 2003]. Improved to $1.0101215 - \epsilon$ and $1.0194553 - \epsilon$. (Chlebík & Chlebíková, TCS 354, 320–338, 2006);

Unique games conjecture: `https://en.wikipedia.org/wiki/Unique_games_conjecture`

2-approximation for the METRIC TSP

Traveling Salesman Problem (TSP): given a **complete** undirected weighted graph $G = (V, E, d)$ such that $d : E \to \mathbb{R}_0^+$, find an Hamiltonian cycle $C^\star$ in G such that $d(C^\star) = \sum_{e \in C^\star} d(e)$ is minimum.

The **Metric TSP** is TSP with triangle inequality, i.e., the cost function d satisfies $d(x, y) \leq d(x, z) + d(z, y)$, for all $x, y, z \in V$.

The approximation algorithms we will introduce for the metric TSP make use of the following property.

Property

Given any walk $\gamma = (x_1, x_2, x_3, \ldots, x_{p-1}, x_p)$, with $p \geq 3$, we can replace (x_{i-1}, x_i, x_{i+1}) by (x_{i-1}, x_{i+1}), to obtain a walk γ' from x_1 to x_p such that $d(\gamma) \leq d(\gamma')$, with $1 < i < p$.

Proof: (x_{i-1}, x_{i+1}) is an edge of G, because G is a complete graph. Thus, γ' is a walk in G. By the triangle inequality,

$d(\gamma') = d(\gamma) + d(x_{i-1}, x_{i+1}) - (d(x_{i-1}, x_i) + d(x_i, x_{i+1})) \leq d(\gamma)$. □

2-approximation for the METRIC TSP (cont.)

A 2-approximation algorithm for Metric TSP

Construct a minimum spanning tree (MST) $T^\star$ of G; Double every edge of T to get an **eulerian graph**; Find an Eulerian tour W on this graph (e.g., induced by a traversal of T in depth-first order); Let C be the list of vertices obtained by deleting all duplicates in W (keep the last vertex); Return C.

Proof: C is an Hamiltonean cycle in G and, by the triangle inequality, $d(C) \leq 2d(T^\star)$ (to remove a duplicate, we replaced two edges in W by a single one in C). If $C^\star$ is the optimal cycle, $d(C) \leq 2d(C^\star)$ because if we delete an edge e from $C^\star$ we get a spanning tree T with $d(T^\star) \leq d(T) = d(C^\star) - d(e) \leq d(C^\star)$. Therefore, $d(C) \leq 2d(T^\star) \leq 2d(C^\star)$. □

1.5-approximation for the metric TSP

Christofides algorithm for the <u>Metric TSP</u>:

- Find a minimum spanning tree $T^\star$ of G.
- Instead of duplicating all edges of $T^\star$ (to form an Eulerian circuit), take the set of nodes $\mathcal{O}$ that have odd degree. (Recall that a graph has an Eulerian circuit iff every node has even degree). For the set $\mathcal{O}$, find a matching $M^\star$ of minimum weight in G. (Note that $M^\star$ exists because $|O|$ is always even and G is complete).
- Add M to $T^\star$ to obtain a subgraph G' of G, with V as vertex set and that has an Eulerian circuit. Find an Eulerian circuit $\mathcal{C}_e$ in G'.
- Visit $\mathcal{C}_e$, eliminating duplicates to produce an Hamiltonean cycle $\mathcal{C}$.

Theorem: Every step can be carried out in a polynomial time and $d(\mathcal{C}) \leq 1.5d(\mathcal{C}^\star)$.

A sketch of the proof: *Given an optimal solution $C^\star$ to the TSP, we can start from a vertex in $\mathcal{O}$ and remove from $C^\star$ all the vertices in $V \setminus \mathcal{O}$. This gives a cycle $\mathcal{C}_\mathcal{O}$ such that $d(\mathcal{C}^\star) \geq d(\mathcal{C}_\mathcal{O})$, by the triangle inequality. $\mathcal{C}_\mathcal{O}$ consists of two disjoint matchings, say M_1 and M_2, for the nodes in O. Since $M^\star$ is minimum, $d(\mathcal{C}_\mathcal{O}) = d(M_1) + d(M_2) \geq 2d(M^\star)$. Therefore, $d(M^\star) \leq 0.5d(\mathcal{C}_\mathcal{O}) \leq 0.5d(\mathcal{C}^\star)$. Moreover, $d(T^\star) \leq d(\mathcal{C}^\star)$ (when we remove an edge from $\mathcal{C}^\star$, we get a supporting tree). Thus, $d(\mathcal{C}) \leq d(T^\star) + d(M^\star) \leq 1.5d(\mathcal{C}^\star)$.* □

Inapproximability of the general TSP

Traveling Salesman Problem (TSP): given a complete undirected weighted graph $G = (V, E, d)$ such that $d : E \to \mathbb{R}_0^+$, find an Hamiltonian cycle $C^\star$ in G such that $d(C^\star) = \sum_{e \in C^\star} d(e)$ is minimum.

Inapproximability of the general TSP

If $P \neq NP$, there is no polynomial time $\alpha(n)$-approximation algorithm for TSP, for any polynomial time computable function $\alpha(n)$, where $n = |V|$.

Proof: By reduction from the HAMILTONIAN CYCLE PROBLEM. Let $G = (V, E)$ be an undirected graph. Construct a complete graph $G' = (V, E')$ from V, and define $d(e) = 1$ if $e \in E$ and $d(e) = \alpha(n)n + 1$ if $e \notin E$, for each $e \in E'$.

Suppose $\mathcal{A}$ is a polynomial time $\alpha(n)$-approximation algorithm for TSP. Run $\mathcal{A}$ on G'. If G has an Hamiltonean cycle $C^\star$, then $\mathcal{A}$ must return a cycle C in G' such that $d(C) \leq \alpha(n)d(C^\star) = \alpha(n)n$. If G has no Hamiltonean cycle, then $\mathcal{A}$ must return a cycle C in G' such that $d(C) \geq (\alpha(n)n + 1) + (n - 1) > \alpha(n)n$, because C must have at least one edge $e \notin E$ (i.e., with $d(e) = \alpha(n)n + 1$). Thus, we can use $\mathcal{A}$ to decide the existence of an hamiltonian cycle in G. Therefore, $\mathcal{A}$ cannot exist if $P \neq NP$. □

BIN PACKING: Approximation and Inapproximability

BIN PACKING: Given a set $S = \{a_1, \ldots, a_n\}$ of items, with $0 < a_i < 1$, for all i, and bins with capacity 1, pack the items using the minimum number of bins.

- **NP-hardness** by reduction from PARTITION;
- Inapproximabiity to $3/2 - \varepsilon$, for $\varepsilon > 0$, if P=NP, by reduction from PARTITION;
- **Belongs to APX**: 2-approximation algorithms, e.g. ”First Fit” strategy; 3/2-approximation for “First Fit Decreasing”

For the proofs and to know more, please refer to:

- http://ac.informatik.uni-freiburg.de/lak_teaching/ws11_12/combopt/notes/bin_packing.pdf
- https://sites.cs.ucsb.edu/~suri/cs130b/BinPacking
- http://ac.informatik.uni-freiburg.de/lak_teaching/ws07_08/algotheo/Slides/13_bin_packing.pdf

$O(\log n)$-approximation for the SET COVER

MIN-SET-COVER Given a set $A = \{a_1, \ldots, a_n\}$ and a collection $\mathcal{F}$ of nonempty subsets of A, find a covering $C^\star \subseteq \mathcal{F}$ of A such that $|C^\star|$ is minimum (if we consider all possible coverings $C \subseteq \mathcal{F}$).

GREEDY APPROXIMATION ALGORITHM: while there are uncovered elements, selects the set that covers the maximum number of uncovered elements.

$O(\log n)$-approximation for the SET COVER (cont.)

Lemma

Suppose there are k sets covering everything. After t choices, the greedy algorithm has at most $(1 - 1/k)^t$ fraction uncovered.

Proof:
If $\mathcal{C}$ is a covering with $|\mathcal{C}| = k$, there is at least a set C_i in $\mathcal{C}$ such that $|C_i| \geq n/k$ (Pigeon's hole principle).

Since the first set selected by the **greedy** algorithm, say F_1, must have at least $|C_i|$ elements, there remain at most $n - n/k$ elements uncovered after the first iteration, i.e., $(1 - 1/k)n$ elements uncovered.

Let $\mathcal{F}' = \{F_j \setminus F_1 \mid F_j \in \mathcal{F}, F_j \setminus F_1 \neq \emptyset\}$ and $\mathcal{C}' = \{C_i \setminus F_1 \mid C_i \in \mathcal{C}, C_i \setminus F_1 \neq \emptyset\}$.

Clearly, $\mathcal{C}' \subseteq \mathcal{F}'$ and $|\mathcal{C}'| \leq k$ and we can use $\mathcal{C}'$ to cover $A \setminus F_1$. If we note that $(1 - 1/|\mathcal{C}'|) \leq (1 - 1/k)$, the result follows. $\square$

$O(\log n)$-approximation for the SET COVER (cont.)

Lemma

Suppose there are k sets covering everything. After t choices, the greedy algorithm has at most $(1 - 1/k)^t$ fraction uncovered.

Proposition

The greedy algorithm is a $(1 + \ln n)$ approximation algorithm.

Proof: Let $k^\star$ be the optimal value. Once we have $< 1/n$ fraction uncovered, we are done.

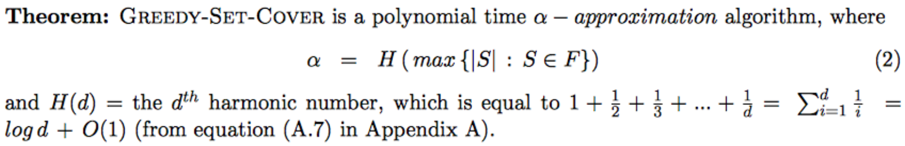
Since $e^x = \sum_{n \in \mathbb{N}} x^n/n!$, we have $1 - 1/k^\star < e^{-1/k^\star}$ and $(1 - 1/k^\star)^t < (e^{-1/k^\star})^t$.

Thus, $e^{-t/k^\star} < 1/n$ if $t > (\ln n)\, k^\star$.

So, the greedy algorithm performs at most $\lfloor (\ln n) k^\star \rfloor + 1$ iterations (and adds a set to the covering per iteration). Hence, $|\mathcal{C}_{greedy}| \leq (1 + \ln n) k^\star$. $\square$

$O(\log n)$-approximation for the SET COVER (cont.)

It can be proved in fact that:



Theorem: Greedy-Set-Cover is a polynomial time $\alpha - \mathit{approximation}$ algorithm, where

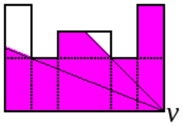
$$\alpha \;=\; H\left(\mathit{max}\,\{|S| \,:\, S \in F\}\right) \tag{2}$$

and $H(d)$ = the d^{th} harmonic number, which is equal to $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{d} = \sum_{i=1}^{d} \frac{1}{i} = \log d + O(1)$ (from equation (A.7) in Appendix A).

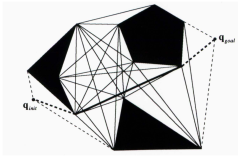
Check CLRS or `https://www.cs.dartmouth.edu/~ac/Teach/CS105-Winter05/Notes/wan-ba-notes.pdf`

Art Gallery Problems (AGP) – Guarding an art gallery

The region visible to *v*



Shortest Path / Visibility graph



- **Visibility** is central to many areas: sensor networks, wireless networks, security and surveillance, and architectural design.
- An art gallery can be viewed as a polygon with or without holes.

The classical Art Gallery Problem by Victor Klee (1973)

How many guards are always sufficient to guard any polygon with *n* vertices (with a 360° view, unlimited range)?

AGP: Combinatorial bounds

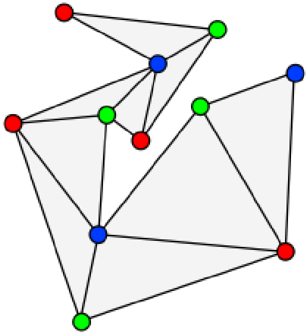
Chvatal's art gallery theorem (1975)

To cover a polygon of n vertices, $\lfloor \frac{n}{3} \rfloor$ stationary guards are always sufficient (and occasionally necessary).

"*A proof from* **THE BOOK**" by **Fisk** (1978):

- The polygon may be partitioned into $n-2$ triangles by adding $n-3$ internal diagonals.

  The dual graph of a triangulated simple polygon is a tree.
- The triangulation graph can always be 3-coloured. (adjacent vertices must have distinct colour)
- Vertices having the same colour form a guard set.
- One of the colours is used by at most $\lfloor n/3 \rfloor$ vertices. Place guards at these vertices.



AGP: Combinatorial bounds

For orthogonal polygons (Kahn, Klawe and Kleitman, 1980, O'Rourke, 1983)

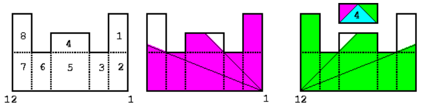
To cover a polygon of n vertices, $\lfloor \frac{n}{4} \rfloor$ stationary guards are always sufficient (and occasionally necessary).

Extensions

- Polygons with or without holes;
- Different types of guards: **stationary** guards (point guards, vertex guards), **mobile** guards (edge guards), . . . ;
- Distinct notions of visibility: (un)limited range, 2π or α-view ($\pi/2$- or π-floodlights), . . .

References: books by O'Rourke, Ghosh. . . ; surveys by Shermer, Urrutia. . . ; two handbooks; several papers

AGP: Hardness



Stationary guards, unlimited visibility range, magnitude 2π. Two points p and q in P see each other if $\overline{pq} \cap Ext(P) = \emptyset$.

How many guards are always sufficient?

Two classical AGP theorems for n-vertex simple polygons: $\lfloor n/3 \rfloor$ guards are sufficient and occasionally necessary [Chvátal, 1975]; $\lfloor n/4 \rfloor$ for orthogonal polygons [Kahn, Klawe & Kleitman, 1983].

What is the fewest number needed for an given polygon P?

NP-hard [Lee & Lin, 1986], even for ortho-polygons [Schuchardt & Hecker, 1995]. APX-hard [Eidenbenz et al., 2001].
Could it be solved exactly in poly-time for some subclasses?

$O(\log n)$-approximation algorithm for MVG by Ghosh

MVG is **NP-hard**. Can we find approximate solutions with provable quality?

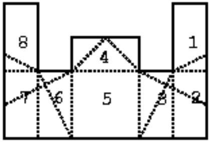
- The problem is APX-hard [Eidenbenz et al., 2001].
- **Ghosh's Algorithm** (1987, 2010) yields a $O(\log n)$-approximation:

1. Consider the decomposition Π induced by the visibility regions to reduce MINIMUM VERTEX GUARD to MINIMUM SET COVER.

   The set A will be the set of pieces to be covered (i.e., Π).

   $\mathcal{F} = \{G_v \mid v \in V\}$, where G_v is set of pieces that v sees.

2. Apply the **greedy algorithm** to solve the MINIMUM SET COVER problem. Approximation ratio $O(\log n)$. Place a guard in the vertex that sees more pieces not covered yet...



- Running time of Ghosh's algorithm: $\mathcal{O}(n^5 \log n)$, improved to $\mathcal{O}(n^4)$ for simple polygons and $\mathcal{O}(n^5)$ for polygons with holes.