Design and Analysis of Algorithms

Approximation Algorithms – Part 3

Pan Peng

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

Outline

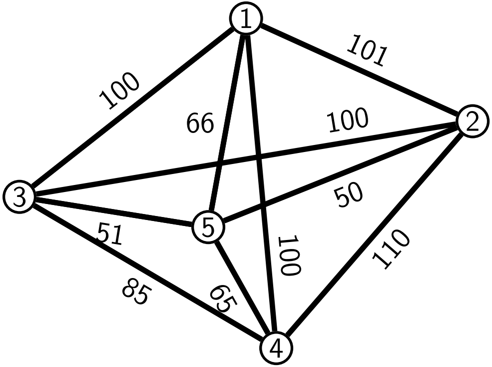
- Approximation algorithms for
    - TSP
    - Max 2-SAT

Travelling salesman problem: a 2-approximation algorithm

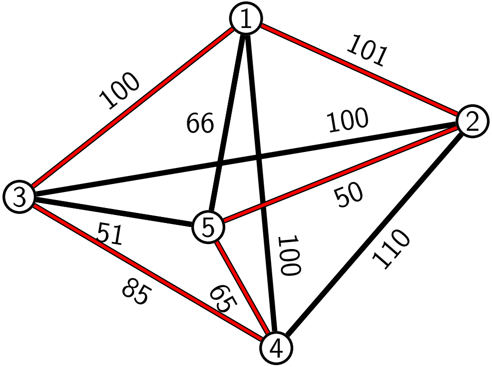
Travelling salesman problem (TSP)

- A salesman must travel between n cities
- Each city is connected to other cities with different distances
- The salesman wants to start and finish at a city after having visited each other city exactly once with minimum total distance

An example



An example



Travelling salesman problem (TSP)

Input: a complete weighted undirected graph $G = (V, E, w)$ with non-negative edge weights $w(e)$'s

Objective: find a cycle that visits each vertex exactly once and has the minimum total weight.

• The total weight (or value) of a cycle is the sum of the weights associated with the edges in the cycle.

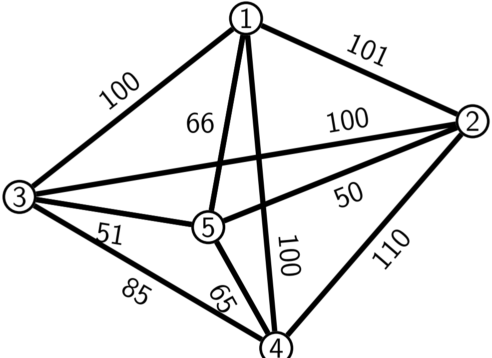
We assume that the edge weights satisfy triangle inequality:

- for any three vertices u, v, x in G, $w(u, v) \leq w(u, x) + w(x, v)$
- TSP with triangle inequality is known as metric TSP
- Mertic TSP is **NP**-hard.

Tools: Minimum spanning tree (MST)

- An MST T of a weighted graph G is a subset of the edges that connects all the vertices together, without any cycles and with the minimum possible total edge weight, i.e., $\sum_{e \in T} w(e)$ is minimum.

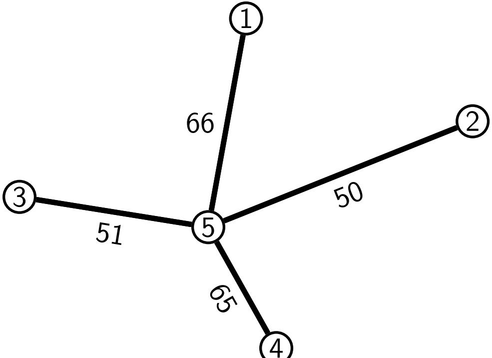
Fact: An MST of a weighted graph G can be found in polynomial time (e.g. by Boruvka's algorithm, Prim's algorithm or Kruskal's algorithm,).



Tools: Minimum spanning tree (MST)

- An MST T of a weighted graph G is a subset of the edges that connects all the vertices together, without any cycles and with the minimum possible total edge weight, i.e., $\sum_{e \in T} w(e)$ is minimum.

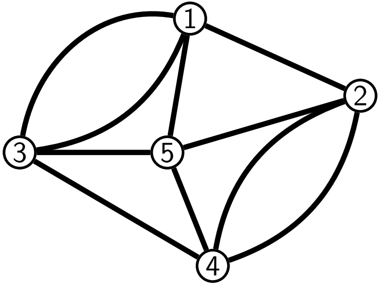
Fact: An MST of a weighted graph G can be found in polynomial time (e.g. by Boruvka's algorithm, Prim's algorithm or Kruskal's algorithm,).



Tools: Eulerian tour (or cycle)

- an Eulerian trail of a (multi-)graph G is a trail that visits every edge exactly once (allowing for revisiting vertices).
- an Eulerian tour (or cycle) of a (multi-)graph G is an Eulerian trail that starts and ends on the same vertex.

Fact: A connected graph has an Eulerian tour iff it has no vertices of odd degree; If exist, it can be found in linear time.



TSP: an algorithm

FindTour

1. Compute an MST T
2. Construct a graph H in which all edges of T are duplicated
3. Find an Eulerian tour C in H (each edge is traversed exactly once).
4. Convert to TSP: if a vertex v is visited twice, create a shortcut from the vertex before v in the tour to the one after v.

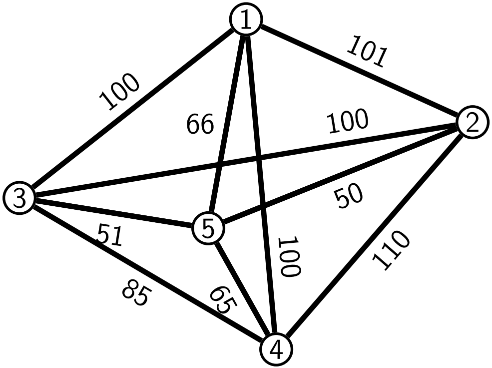
TSP: an algorithm

FindTour

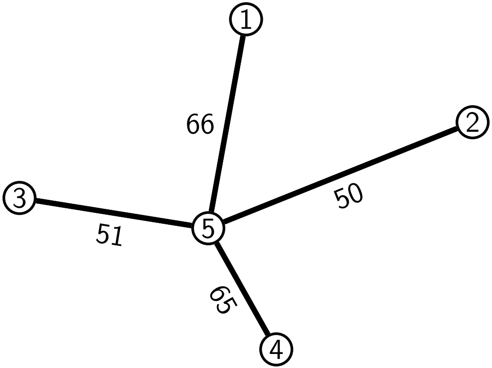
1. Compute an MST T
2. Construct a graph H in which all edges of T are duplicated
3. Find an Eulerian tour C in H (each edge is traversed exactly once).
4. Convert to TSP: if a vertex v is visited twice, create a shortcut from the vertex before v in the tour to the one after v.

note: the starting vertex in the Eulerian tour can be arbitrary

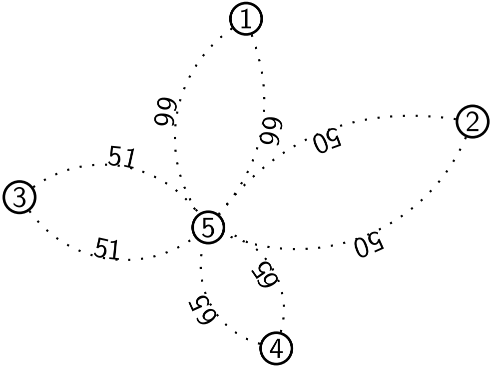
An example



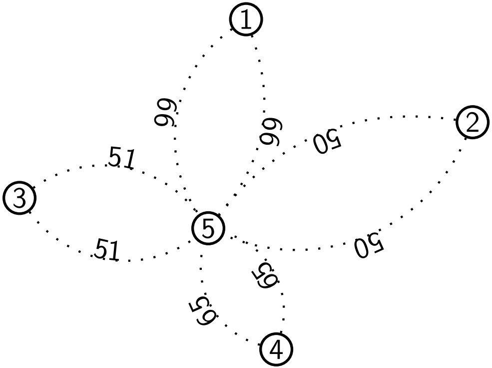
An example



An example

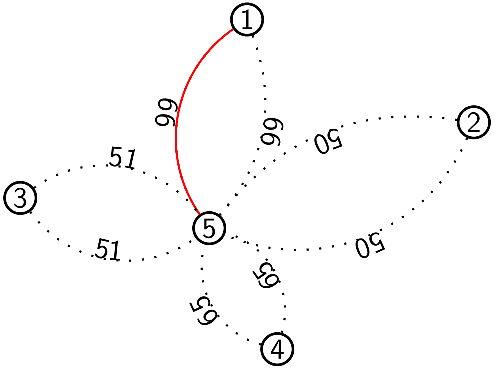


An example



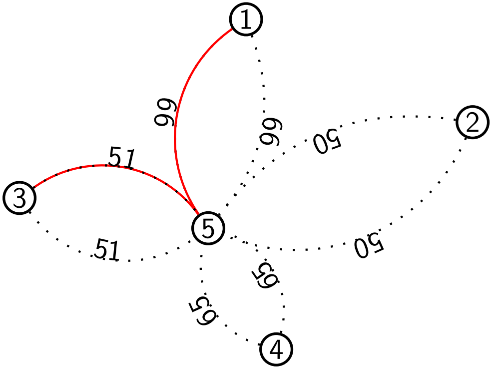
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

An example



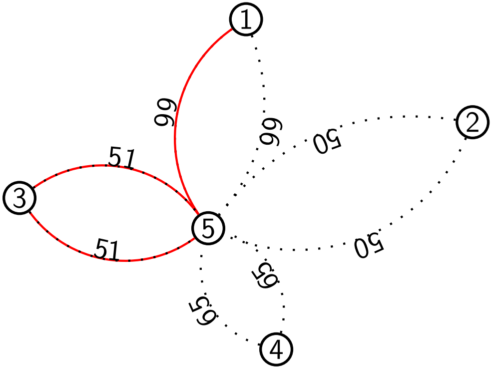
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

An example



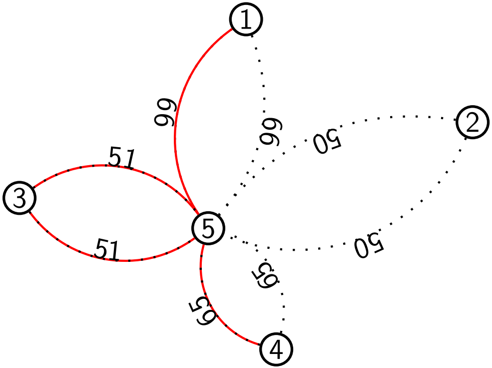
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

An example



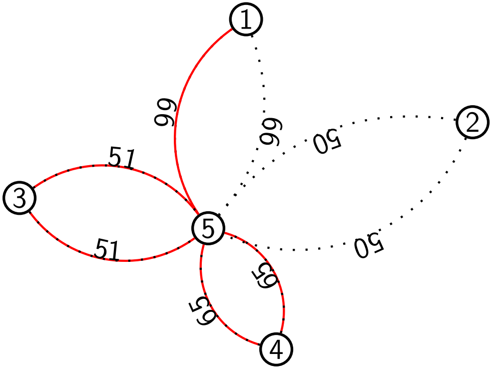
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

An example



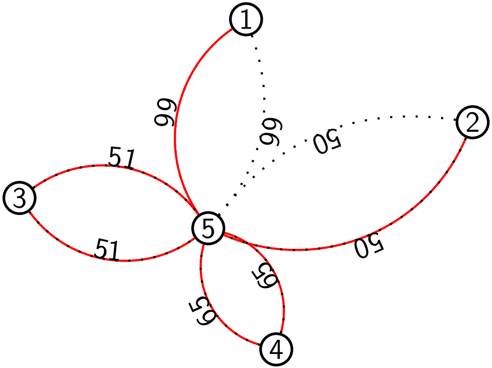
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

An example



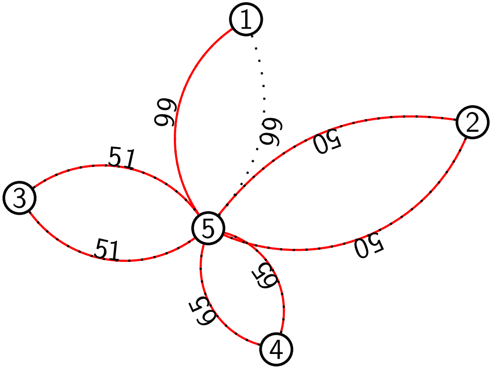
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

An example



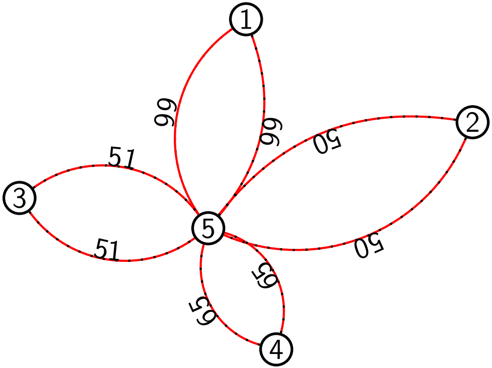
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

An example



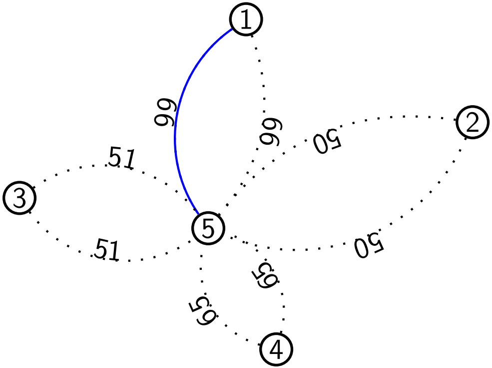
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

An example



Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

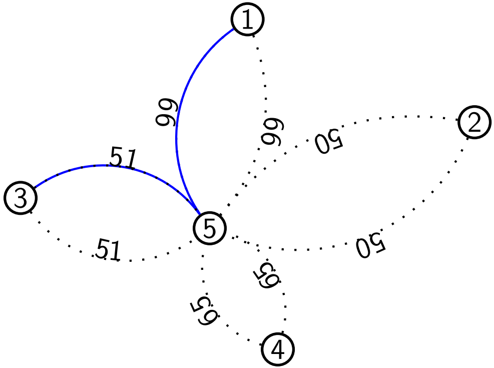
An example



Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

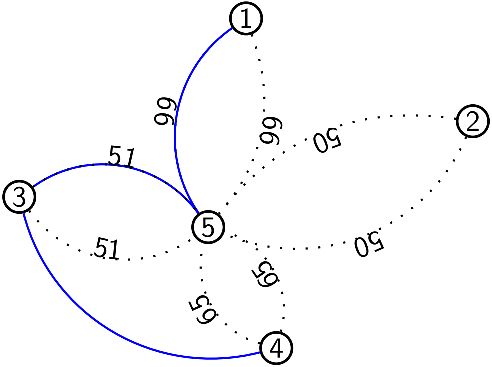
Output cycle: $1, 5, 3, \quad 4, \quad 2, \quad 1$

An example



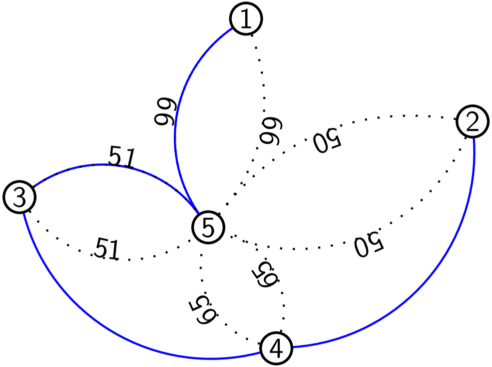
Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$
Output cycle: $1, 5, 3, \quad 4, \quad 2, \quad 1$

An example



Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$
Output cycle: $1, 5, 3, \quad 4, \quad 2, \quad 1$

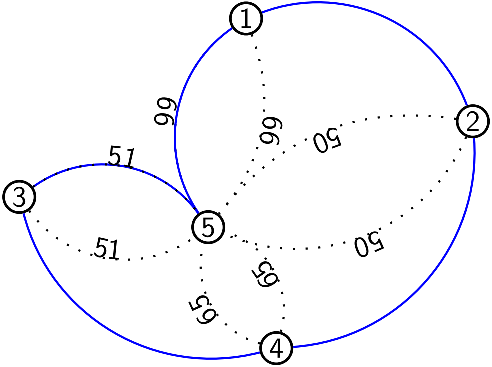
An example



Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

Output cycle: $1, 5, 3, \quad 4, \quad 2, \quad 1$

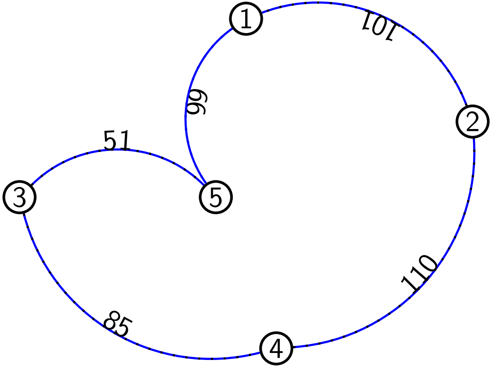
An example



Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$

Output cycle: $1, 5, 3, \quad 4, \quad 2, \quad 1$

An example



Eulerian tour: $1, 5, 3, 5, 4, 5, 2, 5, 1$
Output cycle: $1, 5, 3, \quad 4, \quad 2, \quad 1$

TSP: analysis of the algorithm

Theorem: FindTour is a 2-approximation algorithm of the Metric TSP problem.

Proof

Runs in polynomial time: by the definition of the algorithm

Correctness

- Let C^* be the cost of an optimal tour
- The cost of MST $\leq C^*$

  as removing an arbitrary edge from the optimal tour results in a spanning tree, whose weight is at least the cost of MST
- Cost of the Eulerian tour $\leq 2C^*$

  as the cost of the Eulerian tour $= 2\times$ the cost of MST
- Cost of the final output $\leq$ Cost of the Eulerian tour

  (by triangle inequality)
- Thus, FindTour is a 2-approximation algorithm

Travelling salesman problem: a 1.5-approximation algorithm

TSP: Christofides-Serdyukov algorithm

FindTour-2

1. Compute an MST T
2. Compute a minimum cost perfect matching M on the set of odd-degree vertices of MST T; Add M to T to obtain an Eulerian graph H
3. Find an Eulerian tour C in H (each edge is traversed exactly once).
4. Convert to TSP: go through the vertices in the same order of T, skipping vertices that were already visited. More precisely, if a vertex v is visited twice, create a shortcut from the vertex before v in the tour to the one after v.

Analysis of FindTour-2

Claim
Let $V' \subseteq V$ be a set such that $|V'|$ is even. Let M be a minimum cost perfect matching on V'. Then $\text{cost}(M) \leq C^*/2$, where C^* is the cost of an optimal TSP tour.

Analysis of FindTour-2

Claim

Let $V' \subseteq V$ be a set such that $|V'|$ is even. Let M be a minimum cost perfect matching on V'. Then $\mathrm{cost}(M) \le C^*/2$, where C^* is the cost of an optimal TSP tour.

Proof

- Consider an optimal TSP tour with cost C^*. Let $\mathcal{T}'$ be the tour on V' by shortcutting vertices in $V \setminus V'$.

Analysis of FindTour-2

Claim

Let $V' \subseteq V$ be a set such that $|V'|$ is even. Let M be a minimum cost perfect matching on V'. Then $\mathrm{cost}(M) \leq C^*/2$, where C^* is the cost of an optimal TSP tour.

Proof

- Consider an optimal TSP tour with cost C^*. Let $\mathcal{T}'$ be the tour on V' by shortcutting vertices in $V \setminus V'$.
- By triangle inequality,

$$\mathrm{cost}(\mathcal{T}') \leq C^*$$

Analysis of FINDTOUR-2

Claim

Let $V' \subseteq V$ be a set such that $|V'|$ is even. Let M be a minimum cost perfect matching on V'. Then $\mathrm{cost}(M) \leq C^*/2$, where C^* is the cost of an optimal TSP tour.

Proof

- Consider an optimal TSP tour with cost C^*. Let $\mathcal{T}'$ be the tour on V' by shortcutting vertices in $V \setminus V'$.
- By triangle inequality,

$$\mathrm{cost}(\mathcal{T}') \leq C^*$$

- Note $\mathcal{T}'$ is the union of two perfect matchings on V', each consisting of alternate edges on $\mathcal{T}'$.

Analysis of FINDTOUR-2

Claim

Let $V' \subseteq V$ be a set such that $|V'|$ is even. Let M be a minimum cost perfect matching on V'. Then $\mathrm{cost}(M) \leq C^*/2$, where C^* is the cost of an optimal TSP tour.

Proof

- Consider an optimal TSP tour with cost C^*. Let $\mathcal{T}'$ be the tour on V' by shortcutting vertices in $V \setminus V'$.
- By triangle inequality,
$$\mathrm{cost}(\mathcal{T}') \leq C^*$$
- Note $\mathcal{T}'$ is the union of two perfect matchings on V', each consisting of alternate edges on $\mathcal{T}'$.
- Since M is a minimum cost perfect matching, we have $2\mathrm{cost}(M) \leq \mathrm{cost}(\mathcal{T}') \leq C^*$.

Analysis FindTour-2

Theorem: FindTour-2 is a 1.5-approximation algorithm of the Metric TSP problem.

Proof

Runs in polynomial time: by the definition of the algorithm

Correctness

- Let C^* be the cost of an optimal tour
- The cost of MST $\leq C^*$ (see before)
- Cost of perfect matching $\leq \frac{1}{2}C^*$
- Cost of final output $\leq 1.5C^*$
- Thus, FindTour-2 is a 1.5-approximation algorithm

Further discussion

Question: can we improve the previous approximation ratio?

Further discussion

Question: can we improve the previous approximation ratio?

- since 1976, Christofides-Serdyukov algorithm remained the method with the best approximation ratio until 2020

Further discussion

Question: can we improve the previous approximation ratio?

- since 1976, Christofides-Serdyukov algorithm remained the method with the best approximation ratio until 2020
- recently, the approximation ratio is improved to $1.5 - \varepsilon$, where $\varepsilon > 10^{-36}$ (see a paper published at [STOC 2021])

A (Slightly) Improved Approximation Algorithm for Metric TSP

Anna R. Karlin
karlin@cs.washington.edu
University of Washington
USA

Nathan Klein
nwklein@cs.washington.edu
University of Washington
USA

Shayan Oveis Gharan
shayan@cs.washington.edu
University of Washington
USA

ABSTRACT

For some $\epsilon > 10^{-36}$ we give a randomized $3/2 - \epsilon$ approximation algorithm for metric TSP.

In contrast, there have been major improvements to this algorithm for a number of special cases of TSP. For example, polynomial-time approximation schemes (PTAS) have been found for Euclidean [3, 36], planar [4, 25, 35], and low-genus metric [17] instances. In ad-

Max 2-SAT

Maximum 2-satisfiability problem (MAX 2-SAT)

Input: Boolean variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ in conjunctive normal form.

- each clause consists of two distinct literals t_1, t_2 out of $x_1, \ldots, x_n$ and $\neg x_1, \ldots, \neg x_n$ and contains at most one x_j or $\neg x_j$.

Objective: find a 0-1-assignment of the variables such that a maximum number of clauses is satisfied.

Maximum 2-satisfiability problem (MAX 2-SAT)

Input: Boolean variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ in conjunctive normal form.

- each clause consists of two distinct literals t_1, t_2 out of $x_1, \ldots, x_n$ and $\neg x_1, \ldots, \neg x_n$ and contains at most one x_j or $\neg x_j$.

Objective: find a 0-1-assignment of the variables such that a maximum number of clauses is satisfied.

Example instance:

$(x_1 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_4)$.

Maximum 2-satisfiability problem (MAX 2-SAT)

Input: Boolean variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ in conjunctive normal form.

- each clause consists of two distinct literals t_1, t_2 out of $x_1, \ldots, x_n$ and $\neg x_1, \ldots, \neg x_n$ and contains at most one x_j or $\neg x_j$.

Objective: find a 0-1-assignment of the variables such that a maximum number of clauses is satisfied.

Example instance:

$(x_1 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_4)$.

- Max 2-SAT is NP-hard

A randomized approximation algorithm for MAX 2-SAT

RANDOMASSIGN

1. Assign $x_i = 0$ or $x_i = 1$ with probability $\frac{1}{2}$ independently for each i.
2. output the values x_i, $1 \leq i \leq n$.

A randomized approximation algorithm for MAX 2-SAT

Theorem: The expected number of satisfied clauses given by $\textsc{RandomAssign}$ is at least $\frac{3}{4}\mathrm{OPT}$.

A randomized approximation algorithm for MAX 2-SAT

Theorem: The expected number of satisfied clauses given by RandomAssign is at least $\frac{3}{4}\mathrm{OPT}$.

Proof:

- Let Y_i be an indicator random variable for clause C_i.
  - i.e., $Y_i = 1$ if C_i is satisfied; and 0 otherwise

A randomized approximation algorithm for MAX 2-SAT

Theorem: The expected number of satisfied clauses given by $\textsc{RandomAssign}$ is at least $\frac{3}{4}\mathrm{OPT}$.

Proof:

- Let Y_i be an indicator random variable for clause C_i.
- For each C_i, the values of the two literals are independent.

A randomized approximation algorithm for MAX 2-SAT

> **Theorem:** The expected number of satisfied clauses given by RandomAssign is at least $\frac{3}{4}\mathrm{OPT}$.

Proof:

- Let Y_i be an indicator random variable for clause C_i.
- For each C_i, the values of the two literals are independent.
- $\Pr[Y_i = 0] = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$; $\Pr[Y_i = 1] = 1 - \frac{1}{4} = \frac{3}{4}$.

  - eg., $C_i = x_1 \vee \neg x_3$, then $Y_i = 0$, i.e., C_i is not satisfied if and only if $x_1 = 0$ and $x_3 = 1$.

A randomized approximation algorithm for MAX 2-SAT

Theorem: The expected number of satisfied clauses given by $\textsc{RandomAssign}$ is at least $\frac{3}{4}\mathrm{OPT}$.

Proof:

- Let Y_i be an indicator random variable for clause C_i.
- For each C_i, the values of the two literals are independent.
- $\Pr[Y_i = 0] = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$; $\Pr[Y_i = 1] = 1 - \frac{1}{4} = \frac{3}{4}$.
- Thus, $\mathrm{E}[Y_i] = 0 \cdot \frac{1}{4} + 1 \cdot \frac{3}{4} = \frac{3}{4}$ and for $Y = \sum_{i=1}^{m} Y_i$ we have

$$\mathrm{E}[Y] = \mathrm{E}[\sum_{i=1}^{m} Y_i] = \sum_{i=1}^{m} \mathrm{E}[Y_i] \quad \text{(by linearity of expectation)}$$

$$= \sum_{i=1}^{m} \frac{3}{4} = \frac{3}{4} \cdot m$$

A randomized approximation algorithm for MAX 2-SAT

Theorem: The expected number of satisfied clauses given by RandomAssign is at least $\frac{3}{4}\mathrm{OPT}$.

Proof:

- Let Y_i be an indicator random variable for clause C_i.
- For each C_i, the values of the two literals are independent.
- $\Pr[Y_i = 0] = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$; $\Pr[Y_i = 1] = 1 - \frac{1}{4} = \frac{3}{4}$.
- Thus, $\mathrm{E}[Y_i] = 0 \cdot \frac{1}{4} + 1 \cdot \frac{3}{4} = \frac{3}{4}$ and for $Y = \sum_{i=1}^{m} Y_i$ we have

$$\mathrm{E}[Y] = \mathrm{E}[\sum_{i=1}^{m} Y_i] = \sum_{i=1}^{m} \mathrm{E}[Y_i] \quad \text{(by linearity of expectation)}$$

$$= \sum_{i=1}^{m} \frac{3}{4} = \frac{3}{4} \cdot m$$

- Trivially $\mathrm{OPT} \leq m$ which ends the proof.