Approximation Algorithms

for the Weighted and Unweighted Vertex Cover
and the Metric Travelling Salesman Problem (TSP)

Andreas Kalyfommatos

Π.Μ.Σ Α.Λ.ΜΑ
December 11, 2025

Introduction

In computer science, many of the most fascinating and practically important problems are, unfortunately, **NP-hard**. This means that finding an **optimal solution** for large inputs is often **computationally infeasible**, requiring an impractically long amount of time. Like an old quote goes,

> *"Fast. Cheap. Reliable. Choose two."*

In much the same way, if $P \neq NP$, we cannot simultaneously design algorithms that:

1. always find optimal solutions,
2. run in polynomial time, and
3. work for all possible instances.

To handle NP-hard optimization problems, we must relax at least one of these requirements—typically trading off **optimality for efficiency**, which leads to the study of **approximation algorithms**.

α-approximation defintion

Definition:

Let P be a minimization problem, and I be an instance of P. Let A be an algorithm that finds a feasible solution to instances of P. Let $A(I)$ be the cost of the solution returned by A for the instance I, and OPT(I) be the cost of the optimal solution (mimimum) for I. Then, A is said to be an α-approximation algorithm, where $\alpha > 1$, if

$$\forall I, \quad \frac{A(I)}{OPT(I)} \leq \alpha.$$

Vertex Cover

Input

An undirected graph $G = (V, E)$.

Problem

Find a subset of vertices $S \subseteq V$ such that:

- Covers all edges: every edge $e \in E$ has at least one endpoint in S.
- Has **minimum cardinality**:

$$|S| = \text{minimum possible number of vertices covering all edges.}$$

Maximal Matching

Definition

A **matching** in a graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that no two edges in M share a common endpoint.

Maximal Matching

A matching M is called **maximal** if no additional edge can be added to M without violating the matching property.

Approx-Vertex-Cover Algorithm

Algorithm 1 Approx-Vertex-Cover(G)

$C \leftarrow \emptyset$
while $E \neq \emptyset$ **do**
    pick any $\{u, v\} \in E$
    $C \leftarrow C \cup \{u, v\}$
    delete all edges incident to either u or v
end while
return C

Observation

The set of edges picked by this algorithm is a **maximal matching** M, since no two selected edges share a vertex. An equivalent description:

1. Find a maximal matching M.
2. Return all endpoints of edges in M.

Analysis of Approximation Algorithm

Claim 1

This algorithm returns a vertex cover.
Proof: Every edge in M is covered. If an edge $e \notin M$ were uncovered, then $M \cup \{e\}$ would be a matching, contradicting maximality of M. ■

Claim 2

The cover has size at most $2\times$ optimal.
Proof: The optimal vertex cover must cover all edges in M, so it must contain at least one endpoint of each, implying

$$|C^*| = OPT(I) \geq |M|.$$

Our algorithm returns $A(I) = 2|M|$, since we double count each edge, hence

$$A(I) = 2|M| \leq 2|C^*| = 2 \times OPT(I).$$

Therefore, the algorithm is a **2-approximation**. ■

Is $\alpha = 2$ a Tight Bound?

Question: Is it possible that this algorithm can do better than a 2-approximation?

Answer: We can show that 2-approximation is a *tight bound* by a tight example.

Tight Example: Consider a complete bipartite graph of n black nodes on one side and n red nodes on the other side, denoted $K_{n,n}$.

Complete Bipartite Graph Example

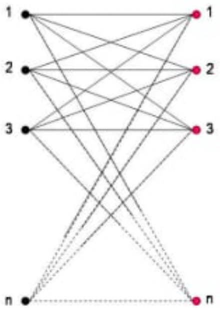

Figure 3: $K_{n,n}$- complete bipartite graph

Counterexample

Notice that the size of any maximal matching of this graph equals:

$$|M| = n.$$

So the $\textsc{Approx-Vertex-Cover}(\text{G})$ algorithm returns a cover of size $2n$:

$$A(K_{n,n}) = 2n.$$

But the optimal solution is clearly:

$$OPT(K_{n,n}) = n.$$

Discussion: Tightness of 2-Approximation

Note that a tight example needs to have arbitrarily large size in order to prove tightness of analysis; otherwise, we could just use brute force for small graphs and the approximation algorithm for large ones, to avoid that bound. Then the algorithm wouldn't truly be "forced" to suffer the worst-case factor of 2.

Here, this example shows that the algorithm gives a 2-approximation no matter what the size n is.

Conclusion

The 2-approximation bound for the Vertex Cover algorithm is **tight**.

Weighted Vertex Cover

Input

An undirected graph $G = (V, E)$ with:

- Vertex weights $w_i \geq 0$ for each vertex $i \in V$.

Problem

Find a subset of vertices $S \subseteq V$ that:

- Covers all edges: every edge $e \in E$ has at least one endpoint in S.
- Has **minimum total weight**:

$$\text{weight}(S) = \sum_{i \in S} w_i .$$

Weighted Vertex Cover: IP Formulation

Given a graph $G = (V, E)$ and vertex weights $w_v \geq 0$, find a minimum-weight vertex cover.

Integer Program

$$\begin{aligned} \min \ & \sum_{v \in V} w_v x_v \\ \text{s.t.} \ & x_u + x_v \geq 1, \quad \forall (u, v) \in E \\ & x_v \in \{0, 1\}, \quad \forall v \in V \end{aligned}$$

LP Relaxation

Relax integrality: $0 \leq x_v \leq 1$. Let x^* be an optimal LP solution (which we can compute in polynomial time).

Rounding Algorithm and 2-Approximation Proof

Algorithm

1. Solve the LP relaxation and obtain x^*.
2. Return

$$C = \{ v \in V \mid x_v^* \geq 1/2 \}.$$

Feasibility

For every edge (u, v), we have $x_u^* + x_v^* \geq 1$. Hence at least one endpoint has $x_v^* \geq 1/2$. Thus C is a valid vertex cover.

2-approximation proof

Key inequality

If $v \in C$, then $x_v^* \geq \frac{1}{2}$, so

$$w_v \leq 2 w_v x_v^*.$$

Summing over all selected vertices:

$$w(C) = \sum_{v \in C} w_v \leq \sum_{v \in C} 2 w_v x_v^* \leq \sum_{v \in V} 2 w_v x_v^*.$$

Conclusion

$$w(C) \leq 2 \sum_{v \in V} w_v x_v^* = 2\, LP^* \leq 2\, OPT.$$

Thus the rounding algorithm is a **2-approximation**.

Optimality of the Factor 2

- The analysis is tight: there exist instances where $w(C) = 2 \times \text{OPT}$.
- Under the **Unique Games Conjecture**, no polynomial-time algorithm can achieve an approximation factor better than $2 - \varepsilon$ for any $\varepsilon > 0$.
- Therefore, the LP-rounding algorithm is essentially **best possible**.

Metric Traveling Salesman Problem (Metric TSP)

Input

A complete graph $G = (V, E)$ with:

- Edge costs $c_{ij} \geq 0$ for all edges.
- Costs satisfy the **triangle inequality**:

$$c_{ij} \leq c_{ik} + c_{kj}, \quad \text{for all } i, j, k \in V.$$

Problem

Find a **Hamiltonian cycle** that visits each vertex exactly once and returns to the start, with **minimum total cost**.

Minimum Spanning Tree (MST)

Input

A connected, undirected graph $G = (V, E)$ with:

- Edge costs $c_e \geq 0$ for each edge $e \in E$.

Problem

Find a spanning tree $T \subseteq E$ that:

- Connects all vertices in V.
- Has **minimum total cost**:

$$\text{cost}(T) = \sum_{e \in T} c_e.$$

Minimum Cost Perfect Matching

Input

A complete graph $G = (V, E)$ with:

- $|V|$ vertices (where $|V|$ is even).
- Edge costs $c_e \geq 0$ for each edge $e \in E$.

Output

A perfect matching $M \subseteq E$ that **minimizes the total cost**:

$$\sum_{e \in M} c_e$$

subject to:

1. Every vertex in V is incident to **exactly one** edge in M.

Note that (1) is equivalent to: No two edges in M share a common vertex.

Useful Lemmas

Lemmas (Basic Graph Theory)

Let G be a graph. Then:

- the number of vertices in G having odd degree is even.
- if G is complete and has an even number of vertices, then it has a perfect matching.
- if G is connected and every vertex has an even degree, then it has an Eulerian tour.

Christofides' Algorithm: Steps and Complexity

Algorithm Steps:

1. **Compute MST:** Find a Minimum Spanning Tree T of the graph $G = (V, E)$.
2. **Find odd-degree vertices:** Let $O = \{v \in V : \deg_T(v) \text{ is odd}\}$.
3. **Minimum-cost perfect matching:** Compute a perfect matching M on $G[O]$ that minimizes $\text{cost}(M) = \sum_{e \in M} c_e$.
4. **Form Eulerian multigraph:** Combine edges: $H = T \cup M$. All vertices in H have even degree.
5. **Construct Hamiltonian cycle:** Find an Eulerian tour in H, then **shortcut** repeated vertices using the triangle inequality to obtain a hamiltonian cycle C.

Shortcutting in Christofides' Algorithm

Idea: After forming an Eulerian tour in $H = T \cup M$, some vertices may be visited multiple times. **Shortcutting** skips repeated vertices to create a Hamiltonian tour without increasing the cost (triangle inequality is used). **Example:**

- Eulerian tour (vertices may repeat):

$$A \to B \to C \to B \to D \to A.$$

- Shortcutting removes repeated visits to B:

$$A \to B \to C \to D \to A.$$

Why it works:

- Graph is **complete** $\Rightarrow$ edge $C \to D$ exists.
- **Triangle inequality:** $c(C,D) \leq c(C,B) + c(B,D) \Rightarrow$ shortcutting does not increase total cost.

Christofides' Algorithm: Steps and Complexity

Time Complexity:

- MST: $O(|E| \log |V|)$ (Prim or Kruskal).
- Matching: $O(|V|^3)$ (Edmonds' Blossom algorithm).
- Euler tour + shortcutting: $O(|E|)$.

Overall: $O(|V|^3)$

Analysis of Christofides' Algorithm

Theorem: Christofides' algorithm is a 1.5-approximation for the metric TSP. **Proof:**

1. Let OPT be the cost of an optimal TSP tour.
2. Compute a Minimum Spanning Tree T. Removing one edge from the optimal tour gives a spanning tree, so
$$\text{cost}(T) \leq \text{OPT}.$$
3. Let O be the set of vertices of **odd degree** in T. Since the number of odd-degree vertices is even, we can form a **minimum-cost perfect matching** M on $G[O]$. Now we shortcut the OPT tour to visit only the vertices from O forming a cycle $C_{|O|}$, where $|O|$ is even. The cycle can be split in two different perfect matchings, M_1, M_2, by picking edges alternately. Because M is minimum we now that
$$OPT \geq cost(M_1) + cost(M_2) \geq 2cost(M) \iff cost(M) \leq \frac{1}{2}OPT.$$

Analysis of Christofides' Algorithm

4. Add M to T to obtain a connected multigraph $H = T \cup M$. Every vertex in H has even degree, so H is **Eulerian**.

5. Find an Eulerian tour of H,**shortcut** repeated vertices and let C be the new tour. The triangle inequality ensures that shortcutting does not increase cost, so

$$\text{cost}(C) \leq \text{cost}(T) + \text{cost}(M).$$

6. Combining bounds:

$$\text{cost}(C) \leq \text{OPT} + \frac{1}{2}\text{OPT} = \frac{3}{2}\text{OPT}.$$

Conclusion: Christofides' algorithm always produces a tour within 1.5 times the optimal cost.

Breaking the 3/2 Barrier for Metric TSP

Main Result: There exists a randomized polynomial-time algorithm achieving an approximation ratio of

$$\frac{3}{2} - \varepsilon$$

for some extremely small $\varepsilon > 0$. This was the first improvement over the classical Christofides' bound of $3/2$.

Key Algorithmic Innovations:

1. **Linear Programming Relaxation:** Solve the Held–Karp (subtour elimination) LP to obtain a fractional solution x, which serves as a lower bound on the optimal TSP cost.
2. **Random Spanning Tree Sampling:** Instead of constructing a single MST, sample a spanning tree T from a carefully chosen probability distribution whose edge marginals match x. This ensures that, in expectation, T has a lower expected cost and better structural properties.

Breaking the 3/2 Barrier for Metric TSP

3. **Parity Correction (Matching):** Find a minimum-cost perfect matching on the odd-degree vertices of the sampled tree T. Combine the edges of T and the matching to obtain an Eulerian multigraph.
4. **Analysis via Structural Properties:** The algorithm's analysis shows that, in expectation, the matching cost is *strictly less* than $\frac{1}{2}\mathrm{OPT}$. This refined bound yields a total expected tour cost below $1.5 \times \mathrm{OPT}$.

Significance:

- Demonstrates that the 3/2 approximation ratio is not tight.
- Opens the door to future progress toward the conjectured 4/3 bound.
- Although the improvement ε is extremely small, it represents a major theoretical breakthrough.

THANK YOU FOR YOUR ATTENTION!