Approximation Algorithms

Many of the NP-Complete problems are most naturally expressed as optimization problems: TSP, Graph Coloring, Vertex Cover etc.

It is widely believed That $\mathbf{P} \neq \mathbf{NP}$ so that it is impossible to solve the problems in polynomial time.

An *approximation algorithm* for solving an optimization problem corresponding to a decision problem in NP is an algorithm which in polynomial time finds an approximative solution which is guaranteed to be close to the optimal solution.
Approximation of Vertex Cover

\textbf{ApproxVertexCover}(G = (V, E))

1. \( C \leftarrow \emptyset \)
2. \textbf{while} \( E \neq \emptyset \)
3. \textbf{Chose an arbitrary edge} \((u, v) \in E\)
4. \( C \leftarrow C \cup \{u\} \cup \{v\} \)
5. \textbf{Remove all edges in} \( E \) \textbf{which contains} \( u \) \textbf{or} \( v \)
6. \textbf{return} \( C \)

The algorithm always returns a vertex cover. When an edge is removed both of its vertices are added to \( C \).

Now consider the edge \((u, v)\). At least one of the vertices \( u \) and \( v \) must be in an optimal vertex cover.
\( \Rightarrow \) The vertex cover returned by the algorithm cannot be more than twice the size of an optimal vertex cover.

Time-complexity: \( O(|E|) \)
To measure approximability

The Approximation Quotient for an algorithm is

\[ \max \frac{\text{approx}}{\text{opt}} \quad \text{for minimization problems} \]

\[ \max \frac{\text{opt}}{\text{approx}} \quad \text{for maximization problems} \]

This means that the quotient is always \( \geq 1 \) with equality if the algorithm always returns the optimal solution.

In all other cases the quotient is a measure of how far from the optimal solution we can get in the worst case.

The algorithm for finding minimal vertex covers has approximation quotient 2 since it returns a vertex cover at most twice as large as the minimal one.
Degrees of approximability

There is a difference between the NP-Complete problems regarding how hard they are to approximate:

- For some problems you can, for every $\epsilon > 0$, find a polynomial algorithm with approximation quotient $1 + \epsilon$. Ex.: The Knapsack Problem

- Other problems can be approximated within a constant $> 1$ but not arbitrarily close to $1$ $P \neq NP$. Ex.: Vertex Cover

- Then there are problems that cannot be approximated within any constant if $P \neq NP$. Ex.: Maximal Clique
Approximation of TSP

We show that TSP $\notin$ APX, i.e. TSP cannot be approximated. Assume, to reach a contradiction, that TSP can be approximated within a factor $B$.

Reduction from Hamiltonian Cycle:

Hamiltoncykel($G$)

1. $n \leftarrow |V|$  

2. foreach $(v_i, v_j) \in E$  

3. $w(p_i, p_j) \leftarrow 1$  

4. foreach $(v_i, v_j) \notin E$  

5. $w(p_i, p_j) \leftarrow |V|B$  

6. if $\text{TSAPPROX}(p_i, t) \leq |V|B$  

7. return TRUE  

8. return FALSE

If TSAPPROX can approximate TSP within factor $B$, then the algorithm decides in polynomial time if there is a Hamiltonian Cycle in $G$ or not. That is impossible!
Approximation of TSP with the triangle inequality

This is a special case of TSP which can be approximated.

**The triangle inequality:** \( w(i, j) \leq w(i, k) + w(k, j) \) for all nodes \( i, j, k \).

The triangle inequality shows that if \( i, j, k_1, k_2, ..., k_s \) form a cycle in the graph, we have \( w(i, j) \leq w(i, k_s) + w(k_s, k_{s-1}) + ...w(k_1, j) \).

TSP with the triangle inequality is called \( \Delta \) TSP.

**Theorem:** \( \Delta \) TSP is NP-Complete.
Assume that we have a minimal spanning tree $T$ in the graph. If we go back and forth along the edges in $T$ we get a walk of length $2w(T)$ where $w(T)$ is the weight sum of the edges in $T$. This walk of course is no solution to the TSP-problem since it is not a cycle. Now, let $C$ be an optimal cycle.

\[ w(C) = OPT. \] Since $C$ is a spanning tree + an edge, we get \[ w(T) \leq w(C). \]

\[ 2 \cdot w(T) \leq 2 \cdot w(C) \leq 2 \cdot OPT \]

We can rearrange the walk along the tree $T$ to a cycle $C_1$ by visiting the nodes in the order that is given by the inorder ordering of the nodes in the tree.

**Claim:** \[ w(C_1) \leq 2 \cdot w(T) \]
This can be shown by repeated use of the triangle inequality.

We now get:

\[ w(C) \leq w(C_1) \leq 2 \cdot w(T) \leq 2 \cdot w(C) \]

we set \( APP = w(C_1) \). We the get:

\[ OPT \leq APP \leq 2 \cdot OPT \]

We can compute \( APP \) in polynomial time. The approximation quotient is \( B = 2 \).

There are more advanced algorithms for approximation of \( \Delta \) TSP One is Christofides algorithm. It uses the same ideas as our algorithm but has an approximation quotient \( \frac{3}{2} \).