christofides algorithm time complexity:Its complexity is **O(n^2 \* log(n))** where n is the number of nodes.30 Jul 2021

# Difference between NP hard and NP complete problem

* **Difficulty Level :** [Hard](https://www.geeksforgeeks.org/hard/)
* **Last Updated :** 04 Oct, 2021

**Prerequisite:** [NP-Completeness](https://www.geeksforgeeks.org/np-completeness-set-1/)

**NP Problem:**   
The NP problems set of problems whose solutions are hard to find but easy to verify and are solved by [Non-Deterministic Machine](https://www.geeksforgeeks.org/difference-between-deterministic-and-non-deterministic-algorithms/) in polynomial time.

[NP-Hard Problem](https://www.geeksforgeeks.org/tag/nphard/)**:**   
A Problem X is NP-Hard if there is an NP-Complete problem Y, such that Y is reducible to X in polynomial time. NP-Hard problems are as hard as NP-Complete problems. NP-Hard Problem need not be in NP class.

[NP-Complete Problem](https://www.geeksforgeeks.org/algorithms-gq/np-complete-gq/)**:**

A problem X is NP-Complete if there is an NP problem Y, such that Y is reducible to X in polynomial time. NP-Complete problems are as hard as NP problems. A problem is NP-Complete if it is a part of both NP and NP-Hard Problem. A non-deterministic  Turing machine can solve NP-Complete problem in polynomial time.

**Difference between NP-Hard and NP-Complete**:

| NP-hard | NP-Complete |
| --- | --- |
| NP-Hard problems(say X) can be solved if and only if there is a NP-Complete problem(say Y) that can be reducible into X in polynomial time. | NP-Complete problems can be solved by a non-deterministic Algorithm/Turing Machine in polynomial time. |
| To solve this problem, it do not have to be in NP . | To solve this problem, it must be both NP and NP-hard problems. |
| Do not have to be a Decision problem. | It is exclusively a Decision problem. |
| Example: Halting problem, Vertex cover problem, etc. | Example: Determine whether a graph has a Hamiltonian cycle, Determine whether a Boolean formula is satisfiable or not, Circuit-satisfiability problem, etc. |

Does genetic algorithm guarantee global optimum?

Due to its random nature, **the genetic algorithm improves the chances of finding a global solution**. Thus they prove to be very efficient and stable in searching for global optimum solutions.

Is simulated annealing guaranteed to find a solution?

In practice, therefore, simulated annealing **cannot be guaranteed to find the globally optimal solution**, but it does usually produce a good solution. As with the genetic algorithm, repeated runs should provide further good solutions.

Due to the probabilistic development of the solution, **GA do not guarantee optimality** even when it may be reached.

You have a wrong concept of optimal solution and best solution. They both are the same: the best solution is the optimal solution. Genetic Algorithms give a good quality solution, and not the best/optimal.

Genetic Algorithms can not guarantee that the solution found be the optimal solution because Genetic algorithms do not explore all the solution space of the problem. Therefore, they can only guarantee a good quality solution.

Only the exact algorithms can guarantee the optimal solution.

My best regards,

Simulated annealing is guaranteed to converge to an optimum solution, while genetic algorithms do not have such a guarantee.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| question | Time complexity | algorithms | Neighborhood Function | The number of new configs | Objective function(cost/energy function) | representation |
| 3-SAT | NP-complete |  | 1. Flipping on bit in *ϕ*. 2. Pick a non-satisfied clause at random and flip one of it’s variables in *ϕ* |  | *C*(*F*(*ϕ*)) the number of clauses in *F* that are satisfied. | 1011011, 8 varibles |
| TSP | TSP-DP,NP-complete  TSP-OPT  NP-hard | Construction Heuristics  **Nearest Neighbour**  **O**(**n2**)*.*  **The Insertion heuristic**  (local minimal)  **O**(**n3** )  Hamiltonian Cycle 1/2(n-1)! | 1. changing the order of two consecutive cities on *H*. (usefuless) 2. replacing two   non-consecutive edges   1. *3-change modifications* | 1. 1/2n(n-3) 2. O(n^3) |  | S=(1,2,5,4,3,1) |
| WGBP | NP-complete |  | 1.**by swapping a node** in *L* with **a node** from *R*.  2.*swapping two nodes* from *L* with  *two nodes* from *R*. |  | the sum of the weights of the edges between sets *L* and *R*. | L=(3,4,6,1,10),R=(2,5,7,8,9) |
| WITNESS |  |  |  |  |  | Prim(*n*2) *> n*2  2 ln *n*  2 ln *n*  *n* |
| Miller-Rabin Test | *O*(log3 *n*) |  |  |  |  |  |
| *0-1knapsack problem* |  |  |  |  |  |  |
| Shortest path | O(v+ELOGN) | Dijstra’s Algorithm |  |  |  |  |
| Min-Spanning Tree | 1. O(v^2), o(Elogv) 2. O(elogv) | Prim, Kruskal |  |  |  |  |

Hamilton cycle: 1/2(n-1)!

N complete graph : 1/2n(n-1)

N tree: n-1

Spanning tree:n^(n-2)

你好，宜伦。  
我认为我们不能在这个问题中使用 2\*ln(n) / n，因为 n=8<9。  
在素数定理中，当 n>=9 时，我们有 Prim(n^2)=n^2/2ln(n)。