- Defined an optimization problem Π, a feasible solution of Π, and an optimal solution of Π. Explained these terms for computing a minimum spanning tree and a maximum independent set for graphs.
- An *approximation algorithm* for an optimization problem outputs a solution in polynomial-time for all instances of the problem and ensures that the value (cost) of the solution is within a (multiplicative/additive) factor of the value of an optimal solution for that problem instance.
  * Approximation algo outputs a feasible solution and the cost of that solution is 'close' to the cost of optimal solution.
  * When the time complexity to obtain an optimal solution is high, these algorithms help in obtaining an approximate solution with lesser time complexity.
  * Especially, approximation algorithms are useful when the problem is (NP-)hard.
- For minimization problems (ex. vertex cover), an α-approximation algorithm outputs solution in the interval $[OPT, \alpha OPT]$, where $1 \leq \alpha$.

  For maximization problems (ex. independent set), an α-approximation algorithm outputs solution in the interval $[\beta OPT, OPT]$ where $0 < \beta \leq 1$.
- A minimization problem has a polynomial time $f(n)$-approximation algorithm whenever the cost of the solution output by that algorithm is upper bounded by $f(n) \cdot OPT$ and it computes this solution in time polynomial in the size of the input instance (which is n). For example, $f(n)$ could be $c \lg n, 2^{\lg^{1-\epsilon} n}$, or n^{ϵ}.

  A minimization problem is said to have a polynomial time α-factor approximation algorithm if it outputs a solution of cost at most $\alpha \cdot OPT$, for a constant $\alpha > 1$, in time polynomial in the length of the input. A problem belongs to complexity class APX if there is a polynomial time α-approximation algorithm for a constant $\alpha > 1$.

  An algorithm $\mathcal{A}$ is said to be an approximation scheme for a minimization problem Π if on input instance I of Π and input parameter $\epsilon > 0$, the scheme $\mathcal{A}$ outputs a solution of cost at most $(1 + \epsilon) \cdot OPT$.
  $\mathcal{A}$ is called a polynomial time approximation scheme (PTAS), if for each fixed $\epsilon > 0$, its running time is upper bounded by a polynomial in the size of the instance. For example, the time complexity of $\mathcal{A}$ could be $(\frac{1}{\epsilon})^{1/\epsilon^2} n^{1/\epsilon^3}$. The PTAS complexity class comprises of problems having a PTAS for any $\epsilon > 0$.

  A problem Π is said to have an efficient-PTAS (EPTAS) if there is a PTAS for any $\epsilon > 0$ to output a solution whose cost is upper bounded by $(1+\epsilon) \cdot OPT$ while its time complexity is of the form $f(\frac{1}{\epsilon}) n^{O(1)}$. For example, the time complexity could be $(\frac{1}{\epsilon})^{1/\epsilon^2} n^{O(1)}$. The EPTAS complexity class comprises of problems having an EPTAS for any $\epsilon > 0$.

  A problem is said to have a fully-PTAS (FPTAS) if there is a PTAS for any $\epsilon > 0$ to output a solution whose cost is upper bounded by $(1+\epsilon) \cdot OPT$ while its time complexity is of the form $(\frac{1}{\epsilon})^{O(1)} n^{O(1)}$. The FPTAS complexity class comprises of problems having an FPTAS for any $\epsilon > 0$.

  For optimization problems in class NP, it is immediate to note that, P $\subseteq$ FPTAS $\subseteq$ EPTAS $\subseteq$ PTAS $\subseteq$ APX $\subseteq$ NP. For a problem not in P, it is desirable that it belongs to FPTAS.

- A minimization problem is *APX-complete* whenever there exists some constant $\alpha > 1$ such that no polynomial time approximation algorithm for this problem can have an approximation factor less than α, unless $P = NP$. Analogous definitions are applicable to complexity classes PTAS, EPTAS, and FPTAS. The examples include max SAT, max 3SAT, min vertex cover, max independent set, min dominating set, max cut, metric TSP, multiway cut, Steiner tree, and integer multicommodity flow. There are also problems that are $\Theta(1)$-appoximable but not known to be in APX-complete; examples include bin packing, min. max.-degree spanning, min edge coloring.

- If an NP-hard optimization (minimization) problem Π admits an FPTAS, then Π also admits a pseudo-polynomial time algorithm. (For any instance I of Π, let I_u be same as I except that numbers in I are encoded in unary (whereas in I these are assumed to be encoded in binary). By setting the value of ϵ to $1/poly(|I_u|)$, FPTAS produces a solution of value at most $OPT + 1$ and its running time is a polynomial in $|I_u|$.) Hence, assuming $P \neq NP$, a strongly NP-hard problem Π cannot have a FPTAS. Refer to note on complexity of pseudpolynomial time algorithms.

- There are also several other classes of inapproximability that are studied as well:

  The set of problems which do not have polynomial-time $c \lg n$-apprx algo for any $0 < c < 1$ unless $P \neq NP$. The examples include set cover, dominating set, hitting set, and minimum connected dominating set.

  The set of problems which do not have polynomial-time $2^{\lg^{1-\epsilon} n}$-apprx algo for any $0 < \epsilon < 1$ unless there exists $n^{poly(logn)}$-time deterministic algorithm for each problem in NP. The examples include label cover, nearest lattice vector, and longest path.

  The set of problems that does not have polynomial-time n^{ϵ}-apprx algo for every $0 < \epsilon < 1$ unless $P \neq NP$. The examples include maximum clique, graph coloring, and maximum set packing.

  There are also classes of hard problems given the hardness of max version of label cover (ex. set cover), min version of label cover (ex. generalized direct Steiner tree), and by believing the unique games conjecture (ex. multicut).

- An (α, β)-approximation algorithm outputs a solution of cost at most $\alpha(OPT) + \beta$. Hence, α is called a multiplicative approximation factor and β an additive approximation factor.

  An *asymptotic PTAS (APTAS)* is a family of algorithm $\{A_\epsilon\}$ along with a constant c where there is an algorithm A_ϵ for each $\epsilon > 0$ such that A_ϵ returns a solution of value at most $(1 + \epsilon)OPT + c$ for minimization problems. As we shall see in detail later, for those problem that have the rescaling property, the small additive term effect is annulled.

- A problem is in randomized approximation class PRAS whenever there is a Las Vegas algorithm with the cost of the solution output by it is $\leq (1 + \epsilon)OPT$ with high probability while the running time is polynomial in n for any fixed ϵ.
  Analogous definitions are applicable to $f(n)$-factor randomized approximable, RAPX, EPRAS, FPRAS, and APRAS.

- Typical pipeline followed: (i) problem description, hardness results, esp., lower bounds on approximability, best approximation result known for the problem; (ii) making a few observations, designing an approximation algorithm; (iii) arguing the algorithm outputs a feasible solution; (iv) upper bounding the

approximation factor; (v) providing a sequence of tight examples; (vi) analyzing the time complexity; (vii) whether it indeed yields an approximation scheme, esp., a PTAS, EPTAS, FPTAS, or an APTAS.