Approximation Algorithms

- Approximation Algorithms for Optimization Problems: Types
- Absolute Approximation Algorithms
- Inapproximability by Absolute Approximate Algorithms
- Relative Approximation Algorithm
- InApproximability by Relative Approximate Algorithms
- Polynomial Time Approximation Schemes
- Fully Polynomial Time Approximation Schemes

Imdad Ullah Khan

Negative Results for Absolute Approximation

- Absolute approximation algorithms are the most desired
  - For large objective values, small additive error is negligible
- Generally absolute approximation algorithms exists for problems where the optimal value lie in a small range
- The hardness of such problems is determining the exact value of the optimum solution within this range
- An absolute approximate algorithm finds solution within a small range and uses the fact that the range is small to get a tight guarantee
- Not many hard problems have an absolute approximation algorithm
- Typically such impossibility of absolute approximation (inapproximability) results use the scaling method

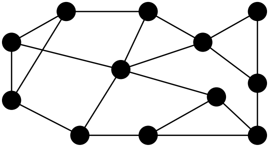
Negative Result by Scaling

Broad idea of scaling

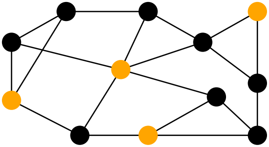
- Scale up certain parameters associated with the instance
- Then show that if there is an absolute approximate algorithm for the scaled up instance, then the solution can be rescaled to get an optimum solution for the original instance
- This yields an efficient algorithm to solve the NP-HARD optimization problem, which by our assumption of $\mathrm{P} \neq \mathrm{NP}$ is not possible

Maximum Independent Set Problem

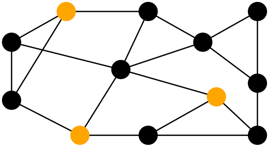
An independent set in G is subset of vertices no two of which are adjacent



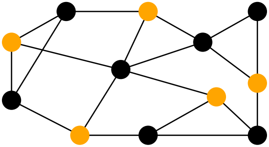
A graph on 12 vertices



An independent set of size 4



An independent set of size 3



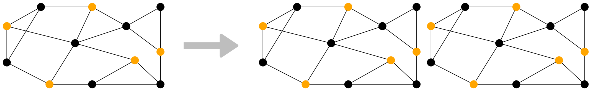
An independent set of size 5 (max)

The MAX-IND-SET(G) problem (MIS): Find a max independent set in G?

MIS: Impossibility of absolute approximation

If $\mathrm{P} \neq \mathrm{NP}$, then there is no polynomial time k-absolute approximation algorithm for the MIS problem

- Suppose there is a k-absolute approximation algorithm $\mathcal{A}$
- Scale the original instance I by a factor of 2 (call this instance $2\mathrm{I}$)



In I Max Ind. set is of size 5 **In $2I$ Max Ind. set is of size 10**

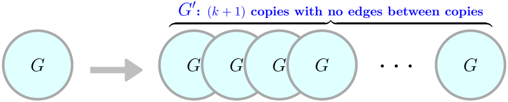
Note: $\mathbf{f}(\textsc{opt}(\mathbf{2}\mathrm{I})) = \mathbf{2f}(\textsc{opt}(\mathrm{I}))$

- Run $\mathcal{A}$ on $2\mathrm{I}$ to get an ind-set of size $\geq f(\textsc{opt}(2\mathrm{I})) - k = 2f(\textsc{opt}(\mathrm{I})) - k$
- This gives an independent set in I of size $\geq f(\textsc{opt}(\mathrm{I})) - {}^{k}/_{2}$
- We got a better ${}^{k}/_{2}$-absolute approximate algorithm
- Repeat the scaling trick, the approximation guarantee drops below 1
- Using integrality of optimal solution we get an optimal solution

MIS: Impossibility of absolute approximation

If $\mathrm{P} \neq \mathrm{NP}$, then there is no polynomial time k-absolute approximation algorithm for the MIS problem

- Suppose there is a k-absolute approximation algorithm $\mathcal{A}$
- Scale the original instance G by a factor of $(k+1)$ (call it G')



Note: $\mathbf{f(OPT(G')) = (k+1)f(OPT(G))}$

- $\mathcal{A}$ on G' gives ind-set of size $\geq f(\mathrm{OPT}(G')) - k = (k+1)f(\mathrm{OPT}(G)) - k$
- We get an ind-set in G of size $\geq f(\mathrm{OPT}(G)) - {}^{k}/_{k+1} \geq f(\mathrm{OPT}(G))$
- Hence we get a maximum independent set in G (of size $f(\mathrm{OPT}(G))$
- Thus, we solved MIS problem in poly-time and proved $\mathrm{P} = \mathrm{NP}$

The KNAPSACK Problem

Input:

- Items: $U = \{a_1, \ldots, a_n\}$ ▷ Fixed order
- Weights: $w : U \to \mathbb{Z}^+$ ▷ $(w_1, \ldots, w_n)$
- Values: $v : U \to \mathbb{Z}^+$ ▷ $(v_1, \ldots, v_n)$
- Capacity: $C \in \mathbb{Z}^+$

Output:

- A subset $S \subset U$
- Capacity constraint:

$$\sum_{a_i \in S} w_i \leq C$$

- Objective: Maximize

$$\sum_{a_i \in S} v_i$$

If $\mathrm{P} \neq \mathrm{NP}$,then there is no polynomial time k-absolute approximation algorithm for the KNAPSACK problem

- Suppose $\mathcal{A}$ is a k-absolute approximation algorithm for KNAPSACK
- Consider an instance $\mathrm{I} = [U, w, v, C]$
- Make an instance $\mathrm{I}' = [U, w, v', C]$, with $v'_i = 2k \cdot v_i$

  ▷ Note that $f(\textsc{opt}(\mathrm{I}')) = (2k)f(\textsc{opt}(\mathrm{I}))$
- Run $\mathcal{A}$ on I' to get a $S \subseteq U$ of total capacity $\leq C$ and total value $\geq f(\textsc{opt}(\mathrm{I}')) - k \;=\; (2k)f(\textsc{opt}(\mathrm{I})) - k$
- The same subset S is a solution of I of value (by v) $(2kf(\textsc{opt}(\mathrm{I}))-k)/2k = f(\textsc{opt}(\mathrm{I})) - 1/2$
- By integrality, S is an optimal solution to I, contradicting $\mathrm{P} \neq \mathrm{NP}$