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

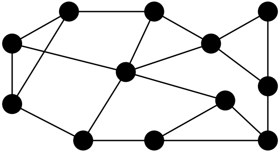
Negative Result by Scaling

Broad idea of scaling

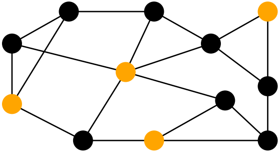
1. Scale up certain parameters associated with the instance
2. Then show that if there is an absolute approximate algorithm for the scaled up instance, then the solution can be rescaled to get an optimum solution for the original instance
3. Conclude that this is an efficient algorithm to solve the NP-HARD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

Maximum Independent Set Problem

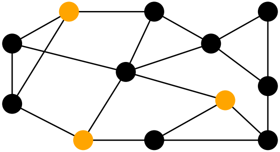
An independent set in G is subset of vertices no two of which are adjacent



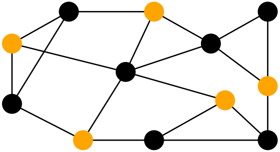
A graph on 12 vertices



An independent set of size 4



An independent set of size 3



An independent set of size 5 (max)

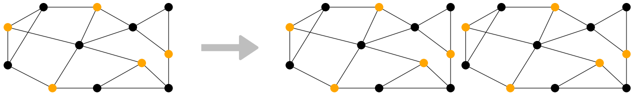
The MAX-IND-SET(G) problem (MIS): Find a max independent set in G?

MIS: Impossibility of absolute approximation

$\mathrm{P} \neq \mathrm{NP} \implies$ there is no poly-time k-absolute approximation algorithm for MIS

Proof: Suppose there is a k-absolute approximation algorithm $\mathcal{A}$

Scale the original instance I by a factor of 2 (call this instance 2I)



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

Note: $f(\text{OPT}(2\mathrm{I})) = 2f(\text{OPT}(\mathrm{I}))$

Run $\mathcal{A}$ on 2I to get an ind-set of size $\geq f(\text{OPT}(2\mathrm{I})) - k = 2f(\text{OPT}(\mathrm{I})) - k$

▷ This gives an independent set in I of size $\geq f(\text{OPT}(\mathrm{I})) - k/2$

We got a better algorithm — a $k/2$-absolute approximate algorithm

Repeat the scaling trick until the approximation guarantee drops below 1

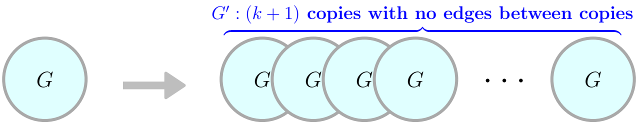
Using integrality of optimal solution we get an optimal solution

MIS: Impossibility of absolute approximation

$\mathrm{P} \neq \mathrm{NP} \implies$ there is no poly-time k-absolute approximation algorithm for MIS

Proof: Suppose there is a k-absolute approximation algorithm $\mathcal{A}$

Scale the original instance G by a factor of $(k+1)$ (call this instance G')



Note: $f(OPT(G')) = (k+1)f(OPT(G))$

$\mathcal{A}$ on G' gives ind-set of size $\geq f(\text{OPT}(G')) - k = (k+1)f(\text{OPT}(G)) - k$

We get an ind-set in G of size $\geq f(\text{OPT}(G)) - {}^{k}/_{k+1} \geq f(\text{OPT}(G))$

Hence we get a maximum independent set in G (of size $f(\text{OPT}(G))$

Thus, we solved MIS problem in poly-time and proved $\mathrm{P} = \mathrm{NP}$

The KNAPSACK Problem

Input:

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

Output:

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

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

- Objective: Maximize

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

KNAPSACK: Impossibility of Absolute Approximation

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

Proof: Suppose there is a k-absolute approximation algorithm $\mathcal{A}$

Consider an instance $\mathrm{I} = [U, w, v, C]$, with $v : U \to \mathbb{Z}^+$

Make an instance $\mathrm{I}' = [U, w, v', C]$, $v'(u) = 2k \cdot v(u)$

▷ Note: $f(\mathrm{OPT}(\mathrm{I}')) = (2k)f(\mathrm{OPT}(\mathrm{I}))$

Run $\mathcal{A}$ on I' to get a $S \subseteq U$ of total capacity $\leq C$ and total value $\geq f(\mathrm{OPT}(\mathrm{I}')) - k = 2kf(\mathrm{OPT}(\mathrm{I})) - k$

S is also a solution of I of value (by v) $^{2kf(\mathrm{OPT}(\mathrm{I}))-k}/_{2k} = f(\mathrm{OPT}(\mathrm{I})) - \frac{1}{2}$

By integrality, S is an optimal solution to I, contradicting $\mathrm{P} \neq \mathrm{NP}$