Introduction to Approximation Algorithms

Dr. Gautam K. Das

Departmet of Mathematics
Indian Institute of Technology Guwahati, India
gkd@iitg.ernet.in

February 19, 2016

Outline of the lecture

- Background
- Definition of approximation algorithms
- Some examples of approximation algorithms
- Polynomial time approximation scheme (PTAS)
- Conclusion

Optimization Problem

1. In mathematics and computer science, an optimization problem is the problem of finding the best solution from all feasible solutions.
2. The objective may be either min. or max. depending on the problem considered.
3. A large number of optimization problems which are required to be solved in practice are NP-hard.
4. For such problems, it is not possible to design algorithms that can find exactly optimal solution to all instances of the problem in polynomial time in the size of the input, unless $P = NP$.

Copying with NP-hardness

- Brute-force algorithms
  1. Develop clever enumeration strategies.
  2. Guaranteed to find optimal solution.
  3. No guarantees on running time.
- Heuristics
  1. Develop intuitive algorithms.
  2. Guaranteed to run in polynomial time.
  3. No guarantees on quality of solution.
- Approximation algorithms
  1. Guaranteed to run in polynomial time.
  2. Guaranteed to get a solution which is close to the optimal solution (a.k.a near optimal).
  3. *Obstacle : need to prove a solution's value is close to optimum value, without even knowing what the optimum value is !*

Approximation Algorithm : Definition

Given an optimization problem $\mathcal{P}$, an algorithm $\mathcal{A}$ is said to be an *approximation algorithm* for $\mathcal{P}$, if for any given instance I, it returns an approximate solution, that is a feasible solution.

Types of approximation

$\mathcal{P}$ An optimization problem

$\mathcal{A}$ An approximation algorithm

I An instance of $\mathcal{P}$

$\mathcal{A}^*(I)$ Optimal value for the instance I

$\mathcal{A}(I)$ Value for the instance I generated by $\mathcal{A}$

1. Absolute approximation
   - $\mathcal{A}$ is an *absolute approximation algorithm* if there exists a constant k such that, for every instance I of $\mathcal{P}$, $|\mathcal{A}^*(I) - \mathcal{A}(I)| \leq k$.
   - For example, Planar graph coloring.
2. Relative approximation
   - $\mathcal{A}$ is an *relative approximation algorithm* if there exists a constant k such that, for every instance I of $\mathcal{P}$, $\max\{\frac{\mathcal{A}^*(I)}{\mathcal{A}(I)}, \frac{\mathcal{A}(I)}{\mathcal{A}^*(I)}\} \leq k$.
   - Vertex cover.

Examples we discuss in this lecture

1. Vertex Cover
2. Traveling Salesman Problem
3. Bin Packing

1) Vertex Cover

Instance An undirected graph $G = (V, E)$.

Feasible Solution A subset $C \subseteq V$ such that at least one vertex of every edge of G belongs C.

Value The value of the solution is the size of the cover, $|C|$, and the goal is to minimize it.

Example

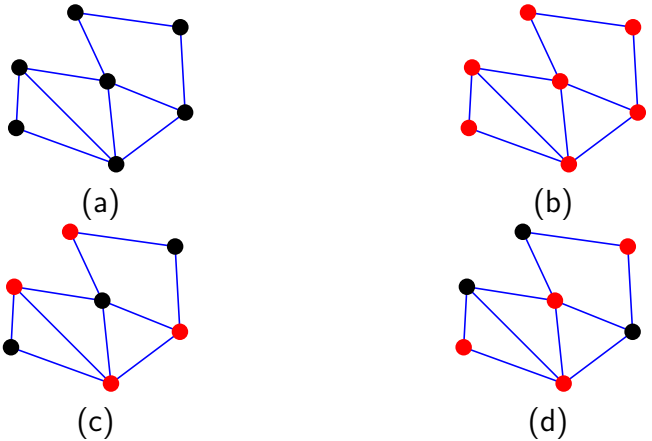


Figure: (a) An undirected graph (b) A trivial vertex cover (c) A vertex cover (d) An other vertex cover

Greedy algorithm 1

1. $C \leftarrow \emptyset$
2. pick any edge $(u, v) \in E$.
3. $C = C \cup \{u, v\}$
4. remove all edges incident on either u or v from E.
5. repeat the process till all edges are removed from E.
6. return C.

Algorithm 1 with an example

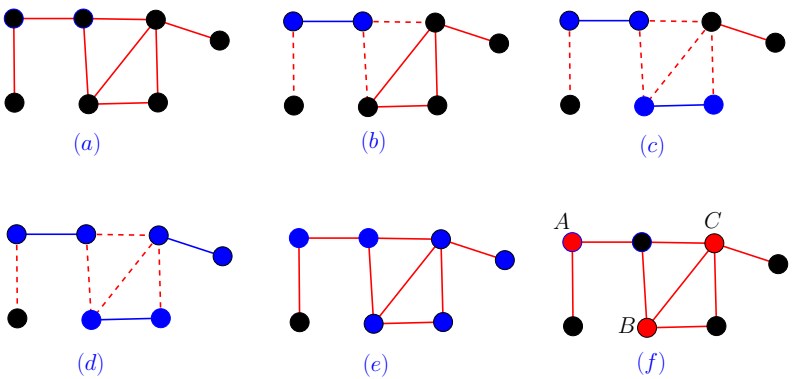


Figure: Execution of algorithm 1

- $|C| = 6$ (blue vertices) and $|C^*| = 3$ (red vertices)
- However if we had picked the edge (B, C) we would have $|C| = 4$

Performance analysis of algorithm 1

1. The edges picked by the algorithm is a *maximal matching* (say M), hence C is a vertex cover.
2. The algorithm runs in time polynomial of input size.
3. The optimum vertex cover (say C^*) must cover every edge in M.
4. Hence C^* contains at least one of the end points of each edge in M, implies $|C^*| \geq |M|$.
5. $|C| = 2 * |M| \leq 2 * |C^*|$, where C^* is an optimal solution.
6. Thus there is a 2-factor approximation algorithm for vertex cover problem.

Tight example : $K_{n,n}$

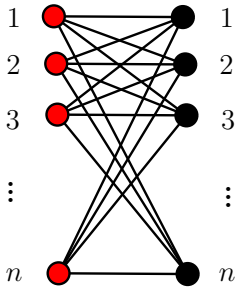


Figure: Complete bipartite graph $k_{n,n}$: a tight example for algorithm 1

- Size of any maximal matching of this graph $= n$, hence $|M| = n$.
- So, our algorithm always produces a cover C of size $2n$.
- But, clearly the optimal solution size $= n$.

Greedy algorithm 2

1. $C \leftarrow \emptyset$
2. take a vertex $v \in V$ of maximum degree $\left(_{\text{tie can be broken arbitrarily}}\right)$.
3. $C = C \cup \{v\}$
4. remove all edges incident on v from E.
5. repeat the process till all edges are removed from E.
6. return C.

Algorithm 2 with an example

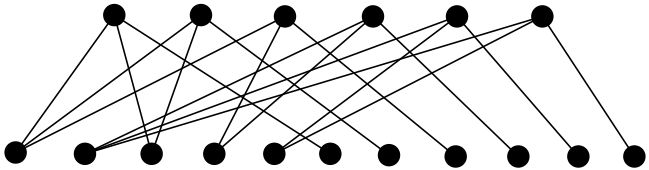


Figure: Execution of algorithm 2

Algorithm 2 with an example

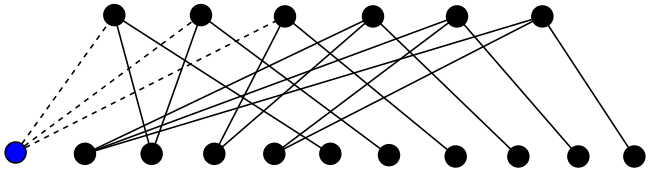


Figure: Execution of algorithm 2

Algorithm 2 with an example

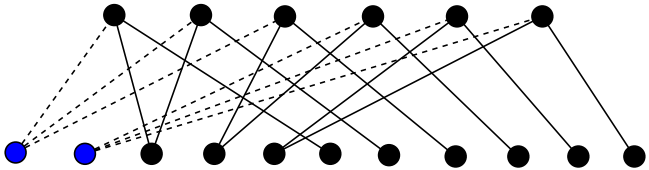


Figure: Execution of algorithm 2

Algorithm 2 with an example

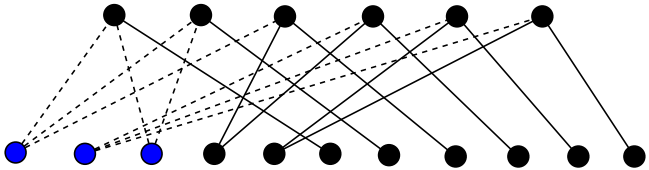


Figure: Execution of algorithm 2

Algorithm 2 with an example

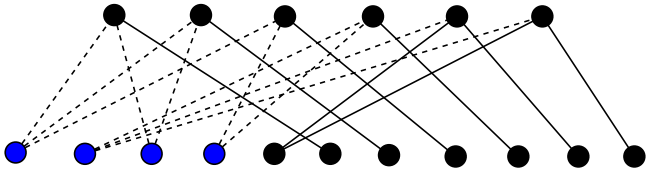


Figure: Execution of algorithm 2

Algorithm 2 with an example

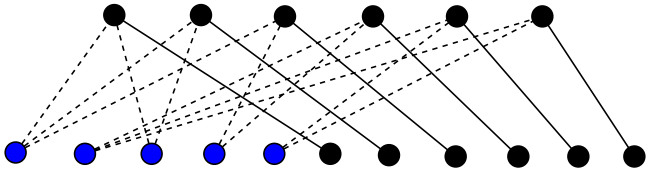


Figure: Execution of algorithm 2

Algorithm 2 with an example

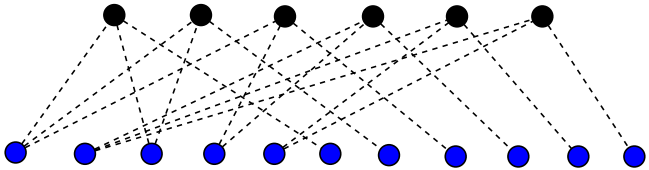
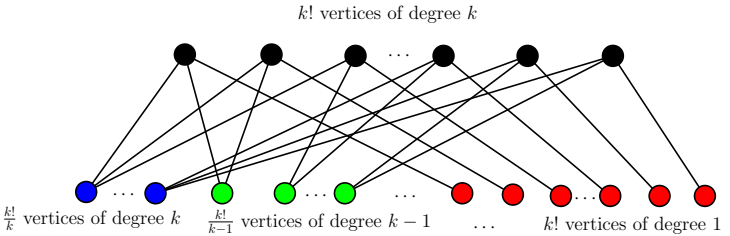


Figure: Execution of algorithm 2

Performance analysis of algorithm 2

Generalizing the previous example



1. The vertices picked by the algorithm forms a vertex cover.
2. The algorithm runs in polynomial time of input size.
3. Optimum solution C^* contains all top vertices, $|C^*| = k!$.
4. Solution C given by algorithm 2 contains all bottom vertices.
5. Hence $|C| = k!(\frac{1}{k} + \frac{1}{(k-1} + \cdots + 1) \approx k! \log k = \log k |C^*|$

Open problem

- Design an approximation algorithm which gives a better approximation.
- A better approximation ratio for the vertex cover problem by [Karakostas, 2009] (Ratio : $2 - \frac{1}{\sqrt{\log n}}$)
- There is no α-approximation algorithm for vertex cover with $\alpha < \frac{7}{6}$ unless $P = NP$ [Håstad, 2001].

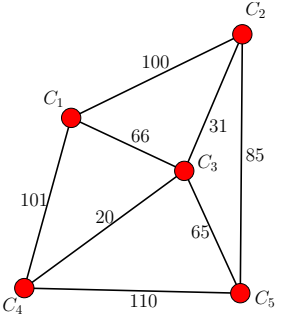
2) Traveling salesman problem (TSP)

- The TSP describes a salesman who must travel between n cities.
- The order in which he visits cities does not matter and he visits each city during his trip, and finishes where he was at first.
- Each city is connected to other cities by airplanes, or by road or railway.
- The salesman wants to keep the travel cost he travels as low as possible.

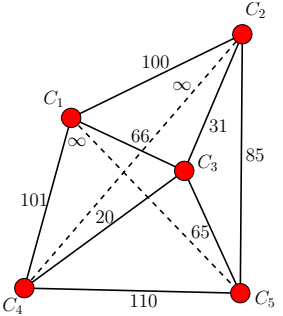
Reducing TSP to graph problem

Instance	A complete weighted undirected graph $G = (V, E)$ with non-negative edge weights.
Feasible Solution	Find a cycle that visits each vertex exactly once.
Value	The value of the solution is the sum of the weights associated with edges in the cycle, and the goal is to find a cycle whose total weight is minimum.

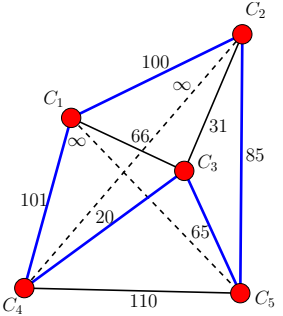
Example



Example



Example

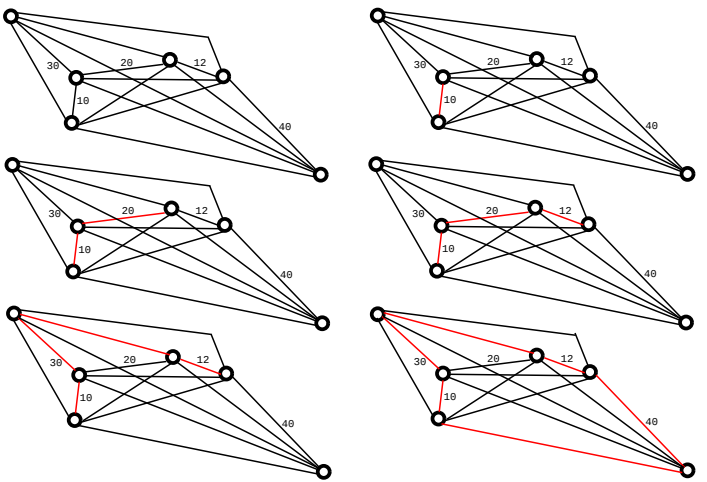


Hardness

1. Bad news : Hard to approximate !
   - For any $c > 1$, there is no polynomial time algorithm which can approximate TSP within a factor of c, unless $P = NP$.
   - In fact, the existence of an $O(2^n)$ - approximation algorithm would imply that $P = NP$.
   - A simple reduction from Hamiltonian cycle.
2. Good news : Easy to approximate if edge weights satisfy *triangle inequality*.
   - The triangle inequality holds in a (complete) graph with weight function w on the edges if for any three vertices u, v, x in the graph $w(u, v) \leq w(u, x) + w(x, v)$.
   - TSP with triangle inequality is also known as (a.k.a) Metric TSP.
   - Metric TSP is still NP-hard, but now we can approximate.

Algorithm 1 : Nearest addition algorithm

1. Start with a tour T, which initially includes two closest nodes $u, v \in V$ and let $S = \{u, v\}$.
2. Repeat until $|S| = n$
   (a) Find a node $v_j \in V \setminus S$ that is closest to S. Let v_j is closest to the node $v_i \in S$; further v_k ve the node following v_i in T.
   (b) Update T by detouring $v_i v_k$ by $v_i v_j v_k$, and set $S = S \cup \{v_j\}$



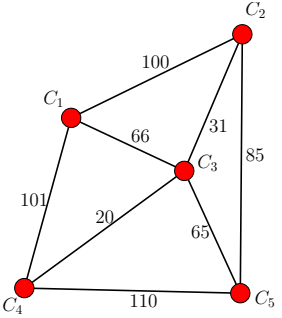
Analysis of Algorithm 1

1. The algorithm is closely related to Prim's MST algorithm.
2. The edges identified form a MST.
3. The cost of the optimal tour is at least the cost of MST on the same input.
4. The cost of the tour on the first two nodes v_i and v_j is exactly $2c_{i,j}$.
5. Consider an iteration in which a node v_j is inserted between nodes v_i and v_k in the current tour.
6. Increase in the length of the tour is $c_{i,j} + c_{j,k} - c_{i,k}$.
7. But, by the triangle inequality $c_{j,k} \leq c_{j,i} + c_{i,k}$
8. Increase in cost in this iteration is at most $c_{i,j} + c_{j,i} = 2c_{i,j}$.
9. Hence the final tour has cost at most twice the cost of the MST.
10. Thus Algorithm 1 is a 2-factor approximation algorithm.

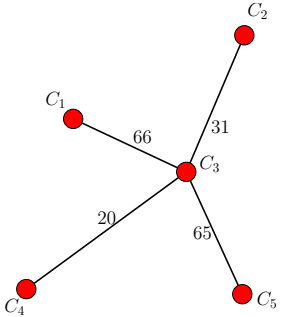
Algorithm 2 : Double tree algorithm

1. Compute a MST.
2. Replace each edge of MST by two copies of itself.
3. Find an Eulerian tour.
4. Shortcut the tour to get a Hamiltonian cycle.
5. Return the Hamilton cycle.

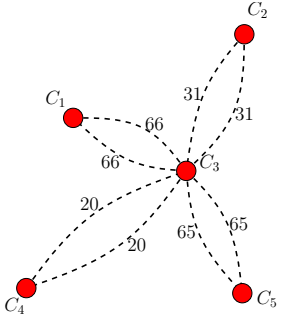
Example



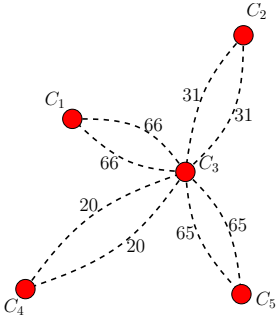
Example



Example



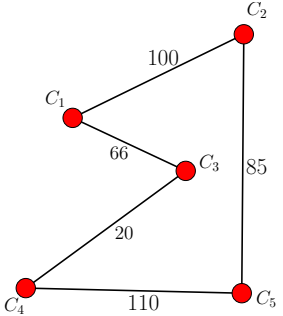
Example



Eulerian tour

$C_1, C_3, C_4, C_3, C_5, C_3, C_2, C_3, C_1$

Example



Eulerian tour

$C_1, C_3, C_4, C_3, C_5, C_3, C_2, C_3, C_1$

Hamilton Cycle

$C_1, C_3, C_4, C_5, C_2, C_1$

Analysis of Algorithm 2

1. Let C^* be the cost of an optimal tour.
2. The cost of MST $\leq C^*$
3. Cost of the Eulerian tour $\leq 2C^*$.
4. Cost of the Hamilton cycle $\leq$ Cost of the Eulerian tour (by triangle inequality).
5. Thus Algorithm 2 is a 2-factor approximation algorithm.

Some results

1. It is possible to get approximation factor 1.5 (Christofide's algorithm).
2. Unless $P = NP$, for any constant $\alpha < \frac{220}{219} \approx 1.0045$, no α-approximation algorithm for metric TSP exists.

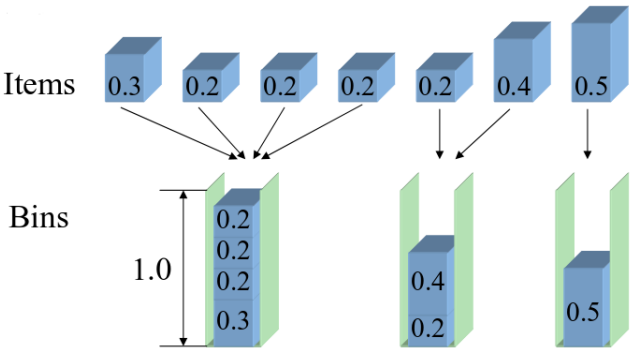
3) Bin Packing

Instance n items with sizes $a_1, a_2, \ldots, a_n$ $(0 < a_i \leq 1)$.

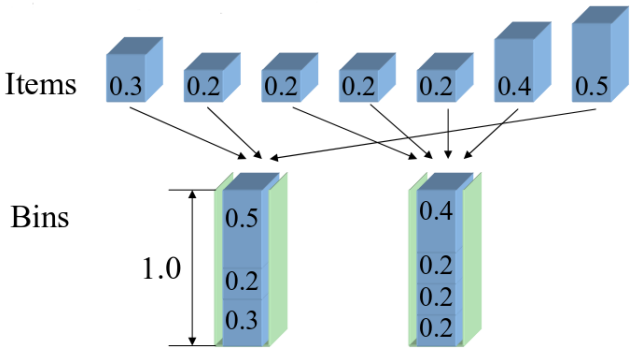
Feasible Solution A packing in unit-sized bins.

Value The value of the solution is the number of bins used, and the goal is to minimize the number.

Example



Example



Algorithm 1: First Fit

1. Place the items in the order in which they arrive.
2. Place the next item into the lowest numbered bin in which it fits.
3. If it does not fit into any open bin, start a new bin.

Ex: Consider the set of items
$S = \{0.4, 0.8, 0.5, 0.1, 0.7, 0.6, 0.1, 0.4, 0.2, 0.2\}$ and bins of size 1

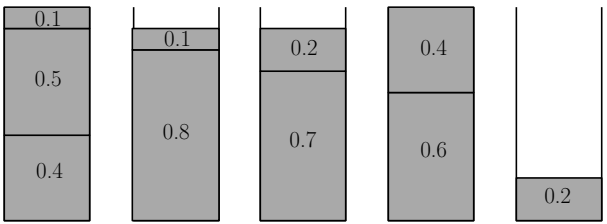


Figure: Packing under First Fit

Analysis of First fit algorithm

1. Let C^* be the optimum number of bins.
2. Suppose our algorithm uses C bins.
3. Then, at least $(C-1)$ bins are more than half full (since we never have two bins less than half full).
4. $C^* \geq \sum_{i=1}^{n} a_i > \frac{C-1}{2} \Longrightarrow 2C^* > C-1 \Longrightarrow 2C^* \geq C$ (as all are integers)
5. Hence 2-factor

Algorithm 2: Next fit algorithm

1. Place the items in the order in which they arrive.
2. Place the next item into the current bin if it fits.
3. If it does not, close that bin and start a new bin.

Ex: Consider the set of items
$S = \{0.4, 0.8, 0.5, 0.1, 0.7, 0.6, 0.1, 0.4, 0.2, 0.2\}$ and bins of size 1

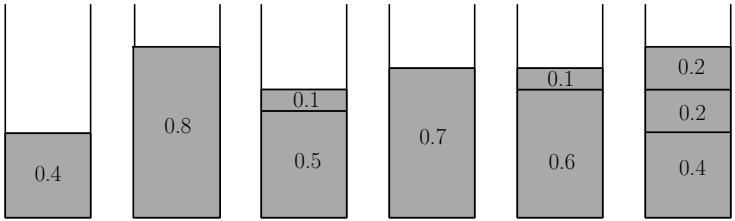


Figure: Packing under Next Fit

Algorithm 3: Best fit algorithm

1. Place the items in the order in which they arrive.
2. Place the next item into that bin which will leave the least room left over after the item is placed in the bin.
3. If it does not fit in any bin, start a new bin.

Ex: Consider the set of items
$S = \{0.4, 0.8, 0.5, 0.1, 0.7, 0.6, 0.1, 0.4, 0.2, 0.2\}$ and bins of size 1

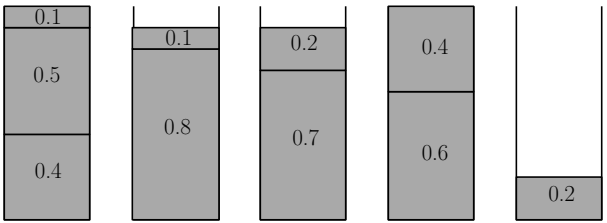


Figure: Packing under Best Fit

- Algorithm 2 and Algorithm 3 are 2-factor approximation algorithms.
- All the 3 algorithms are heuristics.
- Unless $P = NP$, there can not exist a α-approximation algorithm for the bin packing problem for any $\alpha < \frac{3}{2}$.
  - Reduction from *Partition problem*

Polynomial time approximation scheme (PTAS)

1. A PTAS is a family of algorithms $\{A_\epsilon\}$.
2. There is an algorithm for every $\epsilon > 0$.
3. $\{A_\epsilon\}$ is a $(1+\epsilon)$- approximation algorithm for minimization problems.
4. $\{A_\epsilon\}$ is a $(1-\epsilon)$-approximation algorithm for maximization problems.
5. The running time is required to be polynomial in n for every fixed ϵ but can be different for different ϵ.
6. As the ϵ decreases the running time of the algorithm can increase rapidly, e.g., $O(n^{\frac{2}{\epsilon}})$.
7. We have a Fully PTAS (FPTAS) when its running time is polynomial not only in n but also in $\frac{1}{\epsilon}$, e.g., $O((\frac{1}{\epsilon})^3 n^2)$.
8. Bin Packing and MTSP can not admit a PTAS unless $P = NP$. Why?
9. However there is a PTAS for Euclidean TSP (given a set P of points in the Euclidean plane, find a tour of minimum of cost that visits all the points of P [Arora, 1996]).

Shifting Strategy [Hochbaum and Maass, 1985]

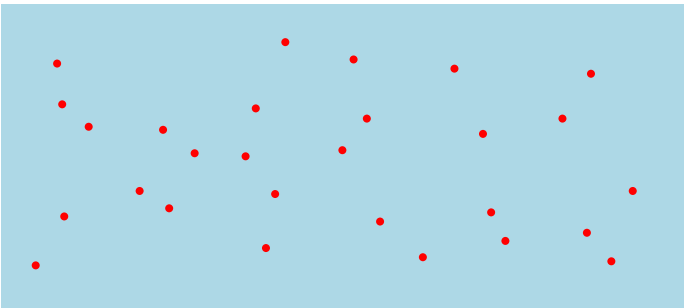


Figure: Points in the plane in an enclosed area

Shifting Strategy [Hochbaum and Maass, 1985]

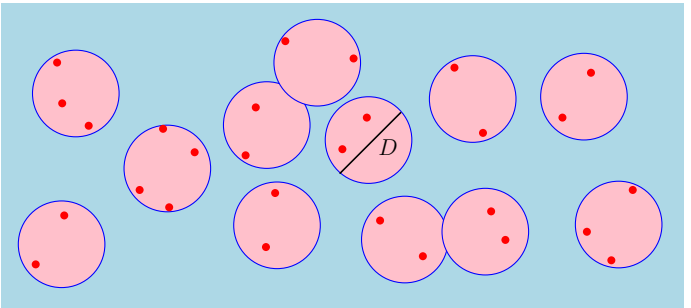


Figure: Covering the points with minimum number of disks of diameter D

Shifting Strategy [Hochbaum and Maass, 1985]

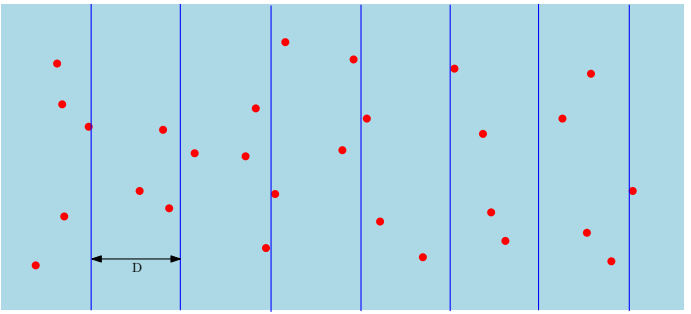


Figure: The area I is subdivided into vertical strips of width D

Shifting Strategy [Hochbaum and Maass, 1985]

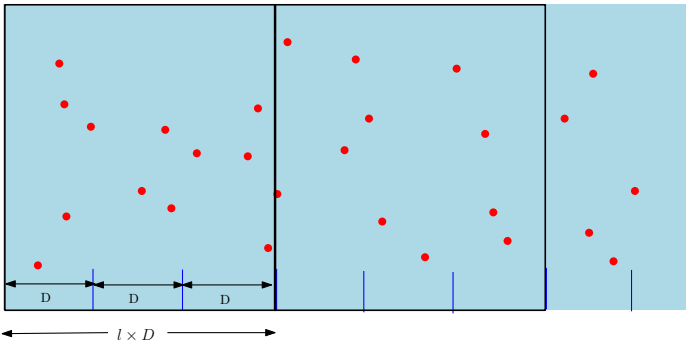


Figure: Groups of l consecutive strips

Shifting Strategy [Hochbaum and Maass, 1985]

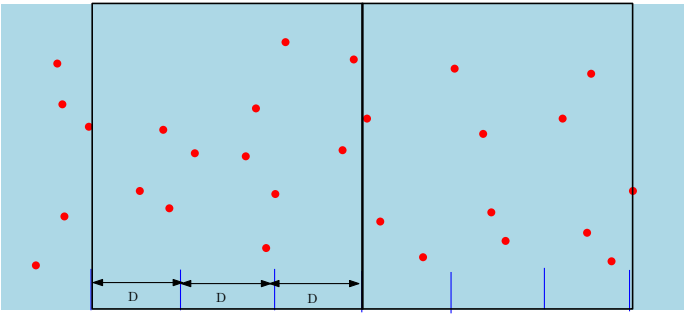


Figure: After first shift

Shifting Strategy [Hochbaum and Maass, 1985]

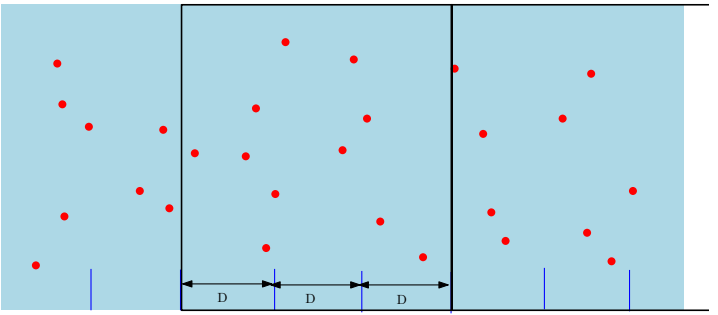


Figure: After second shift

Shifting Strategy [Hochbaum and Maass, 1985]

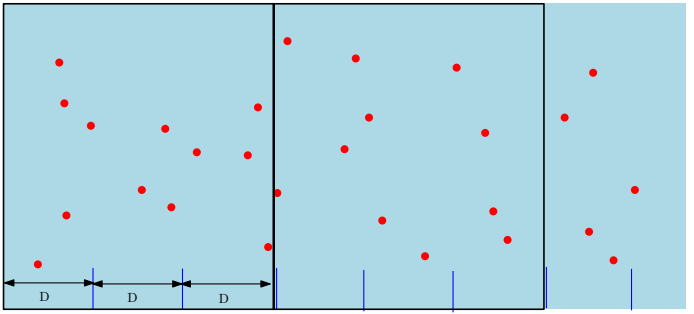


Figure: After third shift

Conclusion

1. Basics of approximation.
2. Need of approximation algorithms.
3. Some examples, Vertex cover, TSP, and Bin packing.
4. PTAS

Some books

1. *The Design of Approximation Algorithms* by David P. Williamson and David B. Shmoys, First Edition, 2011.
2. *Geometric Approximation Algorithms* by Sariel Har-Peled, First Edition, 2011.
3. *Approximation Algorithms* by Vijay V. Vazirani, First Edition.

References I


Arora, S. (1996).
Polynomial time approximation schemes for euclidean tsp and other geometric problems.
In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 2–11. IEEE.


Håstad, J. (2001).
Some optimal inapproximability results.
Journal of the ACM (JACM), 48(4):798–859.


Hochbaum, D. S. and Maass, W. (1985).
Approximation schemes for covering and packing problems in image processing and VLSI.
Journal of the ACM (JACM), 32(1):130–136.


Karakostas, G. (2009).
A better approximation ratio for the vertex cover problem.
ACM Transactions on Algorithms (TALG), 5(4):41.

Thank You!