Greedy Strategy (I)

Methodology, Examples, and Theories

Xiaofeng Gao

Department of Computer Science and Engineering
Shanghai Jiao Tong University, P.R.China

Approximation Seminar IV

Outline

The Goal

- Given:
  - An instance of the problem specifies a set of items
- Goal:
  - Determine a subset of the items that satisfies the problem constraints
  - Maximize or minimize the measure function
- Steps:
  - Sort the items according to some criterion
  - Incrementally build the solution starting from the empty set
  - Consider items one at a time, and maintain a set of “selected" items
  - Terminate when break the problem constraints

Elements of the Greedy Strategy

- A greedy algorithm makes a sequence of choices. For each decision point in the algorithm, the choice that seems best at the moment is chosen.
- It is Primal Infeasible algorithm: construct a solution during the course of the algorithm.
- Benefits:
  - Easy to implement;
  - Have good running time in practice;
  - Natural choices for heuristics.

Self-Reducibility Property

- Subproblem forms a link
  - Greedy Strategy
  - Local Search
- Subproblem forms a tree
  - Divide-and-Conquer
  - Partition
  - Geometric Cut
- Subproblem forms an acyclic graph
  - Dynamic Programming
  - Recurrences

Maximum Independent Set Problem

Definition

Instance: Given a graph $G = (V, E)$

Solution: An independent set $V' \subseteq V$ on G, such that for any $(u, v) \in E$, either $u \notin V'$ or $v \notin V'$.

Measure: Cardinality of the independent set, $|V'|$.

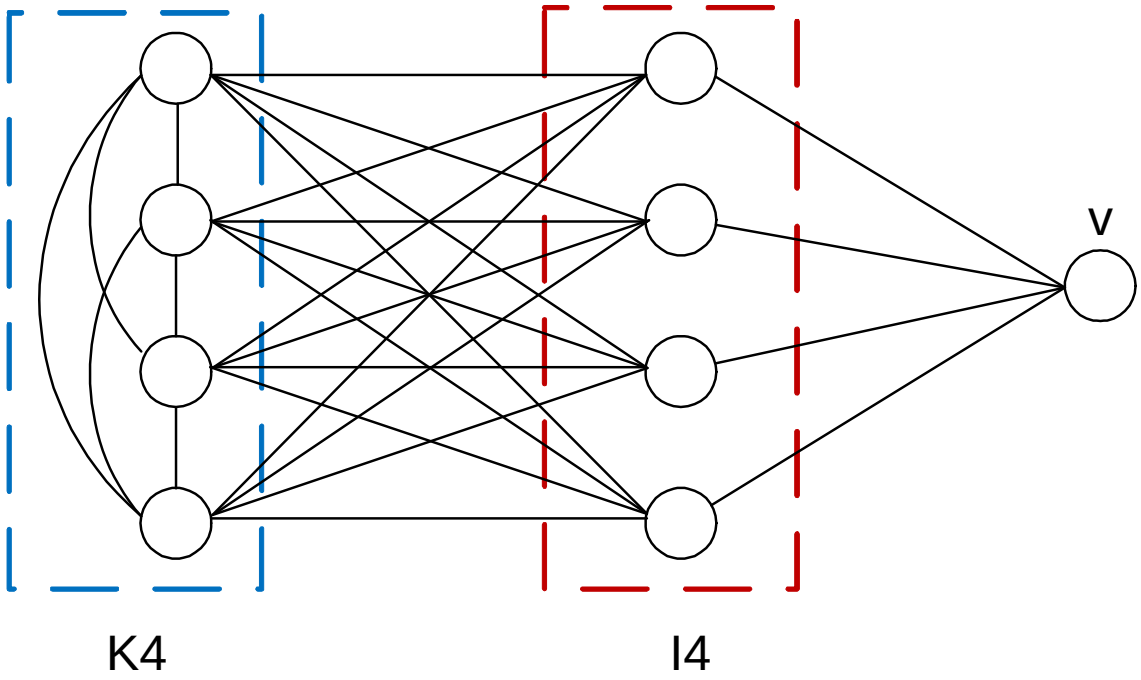
Greedy Algorithm

Algorithm 1 Greedy Independent Set

Input: Graph $G = (V, E)$.
Output: Independent Node Subset $V' \subseteq V$ in G.

1: $V' = \emptyset$;
2: $U = V$;
3: **while** $U \neq \emptyset$ **do**
4: x = vertex of minimum degree in the graph induced by U.
5: $V' = V' \cup \{x\}$.
6: Eliminate x and all its neighbors from U.
7: **end while**
8: Return V'.

Worst Case Example



Let K_4 be a clique with four nodes and I_4 an independent set of four nodes. v is the first to be chosen by algorithm, and the resulting solution contains this node and exactly one node of K_4. The optimal solution contains I_4. Thus $\frac{m^*(X)}{m_g(X)} \geq \frac{n}{2}$.

Approximation Ratio

Theorem

Given a graph G with n vertices and m edges, let $\delta = \frac{m}{n}$. *The approximation ratio of Greedy Independent Set is*

$$\frac{m^*(X)}{m_g(X)} \leq \delta + 1. \qquad (Poly - APX)$$

Proof.

- Define V^* the optimal independent set for G.
- x_i the vertex chosen at i^{th} iteration of Greedy Algorithm.
- d_i the degree of x_i, then each time remove $d_i + 1$ vertices.
- k_i the number of vertices in V^* that are among $d_i + 1$ vertices deleted in the i^{th} iteration.

Proof (2)

- Since algorithm stops when all vertices are eliminated,

$$\sum_{i=1}^{m_g(G)} (d_i + 1) = n.$$

- k_i represent distinct vertices set in V^*,

$$\sum_{i=1}^{m_g(G)} k_i = |V^*| = m^*(G).$$

- Each iteration the degree of the deleted vertices is at least $d_i(d_i + 1)$ and an edge cannot have both its endpoints in V^*, the number of deleted edges is at least $\frac{d_i(d_i+1)+k_i(k_i-1)}{2}$,

$$\sum_{i=1}^{m_g(G)} \frac{d_i(d_i + 1) + k_i(k_i - 1)}{2} \leq m = \delta n.$$

Proof (3)

- Adding three inequalities together, we have

$$\sum_{i=1}^{m_g(G)} \Big(d_i(d_i+1)+k_i(k_i-1)+(d_i+1)+k_i\Big) \le 2\delta n+n+m^*(G)$$

$$\Longrightarrow \sum_{i=1}^{m_g(G)} \left((d_i+1)^2+k_i^2\right) \le n(2\delta+1)+m^*(G).$$

- By applying the Cauchy-Schwarz Inequality, the left part is minimized when $d_i+1=\frac{n}{m_g(G)}$ and $k_i=\frac{m^*(G)}{m_g(G)}$, hence,

$$\frac{n^2+m^*(G)^2}{m_g(G)} \le \sum_{i=1}^{m_g(G)} \left((d_i+1)^2+k_i^2\right) \le n(2\delta+1)+m^*(G),$$

C-S: $\left(\sum_{i=1}^{n} x_i\right)^2 \le \sum_{i=1}^{n} x_i^2$, equality holds when $x_1=\cdots=x_n$.

Proof (4)

- Thus,

$$m_g(G) \geq \frac{n^2 + m^*(G)^2}{n(2\delta+1) + m^*(G)} = m^*(G)\frac{\frac{n^2}{m^*(G)} + m^*(G)}{n(2\delta+1) + m^*(G)}$$

- We have

$$\frac{m^*(G)}{m_g(G)} \leq \frac{2\delta + 1 + \frac{m^*(G)}{n}}{\frac{n}{m^*(G)} + \frac{m^*(G)}{n}}$$

- When $m^*(G) = n$, the right-hand inequality is maximized,

$$\frac{m^*(G)}{m_g(G)} \leq \frac{2\delta + 1 + 1}{1 + 1} = \delta + 1.$$

Note: $\max(m) = \frac{n(n-1)}{2}$ when G is a K_n clique, and $\delta = \frac{n-1}{2}$.

Minimum Max-Lateness Scheduling

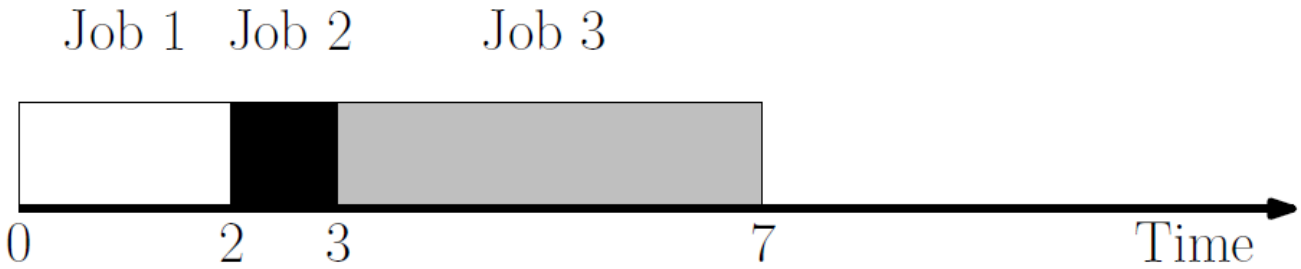
Problem

Instance: Given n jobs and a single machine m. Each job j has a release date r_j, a due date d_j, and a complete time C_j.

Solution: Starting from time 0, if m can process at most one job at a time and must process a job until its completion once it has begun processing, we want a scheduling of the jobs.

Measure: The maximum lateness. $L_{max} = \max_{j=1,\cdots,n} L_j$.

An Example



$$
\begin{array}{lllll}
j=1 & p_1=2 & r_1=0 & d_1=-1 & C_1=2 \quad L_1=2-(-1)=3 \\
j=2 & p_2=1 & r_2=2 & d_2=1 & C_2=3 \quad L_2=3-1=2 \\
j=3 & p_3=4 & r_3=1 & d_3=10 & C_3=7 \quad L_3=7-10=-3
\end{array}
$$

$$L_{max} = \max\{L_1, L_2, L_3\} = 3$$

Notation

S: a subset of jobs.

$r(S)$: Minimum release date of S. $r(S) = \min_{j \in S} r_j$.

$p(S)$: Overall processing time. $p(S) = \sum_{j \in S} p_j$.

$d(S)$: Maximum due date. $d(S) = \max_{j \in S} d_j$.

L^*_{max}: The optimal value.

Assume that all due dates are negative, which implies that the optimal value is always positive.

Lower Bound

Lemma

For each subset S of jobs, $L^*_{max} \geq r(S) + p(S) - d(S)$.

Proof.

Consider the optimal schedule as a schedule for the jobs in the subset S. Let job j be the last job in S to be processed.

Since non of the jobs in S can be processed before $r(S)$, and in total they require $p(S)$ time units of processing, it follows that job j cannot complete any earlier than time $r(S) + p(S)$.

The due date of job j is $d(S)$ or earlier, and so the lateness of job j in this schedule is at lease $r(S) + p(S) - d(S)$. Hence, $L^*_{max} \geq r(S) + p(S) - d(S)$. □

Earliest Due Date Greedy Algorithm

Algorithm 2 Greedy EDD

Input: Set J with n jobs, each j with r_j, p_j, and d_j.
Output: A processing order O (schedule) of n jobs.

1: Sort J in non-decreasing order with respect to d_j.
2: $t = 0$.
3: **while** $J \neq \varnothing$ **do**
4: Pick the first $j' \in J$ with $r_j \leq t$;
5: Insert j' into O;
6: $J = J \backslash \{j'\}$, $t = t + p_j$;
7: Mark m as processing during $[t, t + p_j)$;
8: **end while**
9: Return O

Performance Analysis

Theorem

Greedy EDD is a 2-APX for the problem of minimizing the maximum lateness on a single machine subject to release dates with negative due dates.

Proof: Consider O by greedy EDD, and let job j be a job of maximum lateness in O ($L_{max} = C_j - d_j$).

Find the earliest point in time $t \leq C_j$ such that the machine was processing without any idle time for the entire period $[t, C_j)$.

Several jobs may be processed in this time interval; we only require that the machine not be idle for some interval of positive length within it.

Proof (Continue)

Let S be the set of jobs that are processed in the interval $[t, C_j)$. By our choice of t, non of these jobs were available just prior to t (clearly, at least one job in S is available at time t); hence, $r(S) = t$.
Since only jobs in S are processed throughout this time interval, $p(S) = C_j - t = C_j - r(S)$. Thus, $C_j \leq r(S) + p(S)$.
Since $d(S) < 0$, we can apply previous Lemma:

$$L^*_{max} \geq r(S) + p(S) - d(S) \geq r(S) + p(S) \geq C_j.$$

On the other hand, when $S = \{j\}$, by Lemma

$$L^*_{max} \geq r_j + p_j - d_j \geq -d_j.$$

Combining two inequalities, we have $L_{max} = C_j - d_j \leq 2L^*_{max}$.

Maximum Knapsack Problem

Problem

Instance: *Given finite set X of items and a positive integer b, for each $x_i \in X$, it has value $p_i \in Z^+$ and size $a_i \in Z^+$.*

Solution: *A set of items $Y \subseteq X$ such that $\sum_{x_i \in Y} a_i \leq b$.*

Measure: *Total value of the chosen items, $\sum_{x_i \in Y} p_i$.*

Greedy Algorithm

Algorithm 3 Greedy Knapsack

Input: X with n item and b; for each $x_i \in X$, value p_i, and a_i.
Output: Subset $Y \subseteq X$ such that $\sum_{x_i \in Y} a_i \leq b$.

1: Sort X in non-increasing order with respect to the ratio $\frac{p_i}{a_i}$
▷ Let $x_1, \cdots, x_n$ be the sorted sequence
2: $Y = \emptyset$;
3: **for** $i = 1$ to n **do**
4: **if** $b \geq a_i$ **then**
5: $Y = Y \cup \{x_i\}$;
6: $b = b - a_i$;
7: **end if**
8: **end for**
9: Return Y

Time Complexity

Theorem

Greedy Knapsack has time complexity $O(n \log n)$.

Proof.

Consider items in non-increasing order with respect to the profic/occupancy ratio.
(1). To sort the items, it requires $O(n \log n)$ times;
(2). and then the complexity of the algorithm is linear in their number.
Thus the total running time is $O(n \log n)$. □

Approximation Ratio

Theorem

The solution of Greedy Knapsack can be arbitrarily far from the optimal value.

Proof.

- Consider an instance X of Maximum Knapsack with n items. $p_i = a_i = 1$ for $i = 1, \cdots, n-1$. $p_n = b - 1$ and $a_n = b = kn$ where k is an arbitrarily large number.
- Let $m^*(X)$ be the size of optimal solution, and $m_g(X)$ the size of Greedy Knapsack solution. Then, $m^*(X) = b - 1$, while $m_g(X) = n - 1$,
- $\frac{m^*(X)}{m_g(X)} > \frac{kn-1}{n-1} > k$.

Improvement

The poor behavior of Greedy Knapsack if due to the fact that the algorithm does not include the element with highest profit in the solution while the optimal solution contains only this element.

Theorem

Given an instance X of the Maximum Knapsack problem, let $m_H(X) = \max\{p_{max}, m_g(X)\}$. where p_{max} is the maximum profit of an item in x. Then $m_H(x)$ satisfies the following inequality:

$$\frac{m^*(X)}{m_H(X)} < 2. \quad \textit{(Constant-Factor Approximation)}$$

Proof (1)

Let j be the first index of an item in the order that cannot be included. The profit achieved so far (up to item j) is:

$$\bar{p_j} = \sum_{i=1}^{j-1} p_i \leq m_g(X).$$

The total occupancy (size) is

$$\bar{a_j} = \sum_{i=1}^{j-1} a_i \leq b.$$

We then show that $m^*(X) < \bar{p_j} + p_j$.

Proof (2)

x_i are ordered by $\frac{p_i}{a_i}$, so any exchange of any subset of the chosen items $x_1, \cdots, x_{j-1}$ with any subset of the unchosen items $x_j, \cdots, x_n$ doesn't increase $\bar{a}_j$, and doesn't increase the overall profit. Thus m^* is bounded by $\bar{p}_j$ plus the maximum profit from filling the remaining space.

Since $\bar{a}_j + a_j > b$ (otherwise x_j will be selected), we obtain:

$$m^*(X) \leq \bar{p}_j + (b - \bar{a}_j)\frac{p_j}{a_j} < \bar{p}_j + p_j.$$

Proof (3)

To complete the proof we consider two possible cases.

- If $p_j \leq \bar{p}_j$, then

$$m^*(X) < \bar{p}_j + p_j \leq 2\bar{p}_j \leq 2m_g(X) \leq 2m_H(X).$$

- If $p_j > \bar{p}_j$, then $p_{max} > \bar{p}_j$, and

$$m^*(X) < \bar{p}_j + p_{max} \leq 2p_{max} \leq 2m_H(X)$$

Thus Greedy Knapsack is 2-approximation.

Further Improvement

(1) Consider a fixed profit p, X can be divided into two groups: one of items with profit no greater than a (X_p), while another of items with profits greater than p ($\overline{X}_p$). Note that for any feasible solution,

$$|\overline{X}_p| \leq \frac{m^*(X)}{p} \leq \frac{2m_H(X)}{p}.$$

(2) Run Brute-Force approach for $\overline{X}_p$

(3) Append some items from X_p by *Improved Greedy Knapsack* (*IGK*).

(4) The combination is still a polynomial time algorithm.

Polynomial Time Approximation Scheme

Algorithm 4 Greedy Knapsack PTAS

1: Run $IGK(X)$ and get a solution $m_H(X)$.
2: Set $p \leftarrow \epsilon \cdot m_H(X)$. $X_p = \{x_i \in X \mid p_i \leq p\}$
3: Reorder $X_p = \{x_1, \cdots, x_m\}$ by p_i/a_i $(m \leq n)$
4: **for** Any $|X'| \leq 2/\epsilon$ with $X' \subseteq X_p$ **do**
5: **if** $\sum_{x_i \in X'} a_i > b$ **then** $P(X') \leftarrow 0$;
6: **elseif** $\sum_{i=1}^{m} a_i \leq b - \sum_{x_i \in X'} a_i$ **then** $P(X') \leftarrow \sum_{i=1}^{m} p_i + \sum_{x_i \in X'} p_i$;
7: **else** $P(X') \leftarrow \sum_{i=1}^{k} p_i + \sum_{x_i \in X'} p_i$; $(\sum_{i=1}^{k} a_i \leq b - \sum_{x_i \in X'} a_i < \sum_{i=1}^{k+1} a_i)$
8: **end for**
9: Return $m_P(X) = \max\{P(X') \mid X' \subseteq \{x_{m+1}, \cdots, x_n\}, |X'| \leq 2/\epsilon\}$

Time Complexity

Theorem

The time complexity of Greedy PTAS is $O(n^{1+2/\epsilon})$.

Proof.

There are totally $C_n^{n-m} = C_n^m$ number of X' to consider, each of which cost at most $O(n)$ for the greedy approach.

Since $m \leq 2/\epsilon$, the total complexity is $O(n \cdot C_n^{2/\epsilon}) \leq O(n \cdot n^{2/\epsilon}) = O(n^{1+2/\epsilon})$.

ϵ is a given constant, so Greedy PTAS is a polynomial time algorithm for n (but not for $1/\epsilon$). □

Approximation Ratio

Theorem

Greedy PTAS is an $(1+\epsilon)$-APX for any given ϵ.

Proof. Similarly, let X^* be the optimal solution, then

$$\sum_{x_i \in X^*} p_i = m^*(X), \sum_{x_i \in X^*} a_i \leq a.$$

Let $\overline{X} = \{x_i \in X^* \mid p_i > p\}$, then $|\overline{X}| \leq \frac{m(X^*)}{p} \leq \frac{2m_H(X)}{p} = \frac{2}{\epsilon}$.

According to the greedy approach, X' will eventually be $\overline{X}$. Then according to the previous proof,

$$P(\overline{X}) \leq m^*(X) \leq P(\overline{X}) + p$$

Proof (2)

Since $m^P(X)$ is the maximum value among all $P(X')$, we have

$$P(\overline{X}) \leq m_P(X) \leq m^*(X) \leq P(\overline{X}) + p \leq m_P(X) + p.$$

If X_H is the resulting set of Greedy Knapsack, and $\overline{X_H} = \{x_i \in X_H \mid p_i > p\}$, then $P(\overline{X_H}) = m_H(X)$.

Also, $|\overline{X_H}| \leq \frac{m_H(X)}{a} \leq \frac{1}{\epsilon}$, we have $m_H(X) \leq m_P(X)$.

$m^*(X) \leq m_P(X) + p = m_P(X) + \epsilon \cdot m_H(X) \leq (1+\epsilon)m_P(X)$, which denotes that it is a $(1+\epsilon)$-APX. □