More Approximation Algorithms

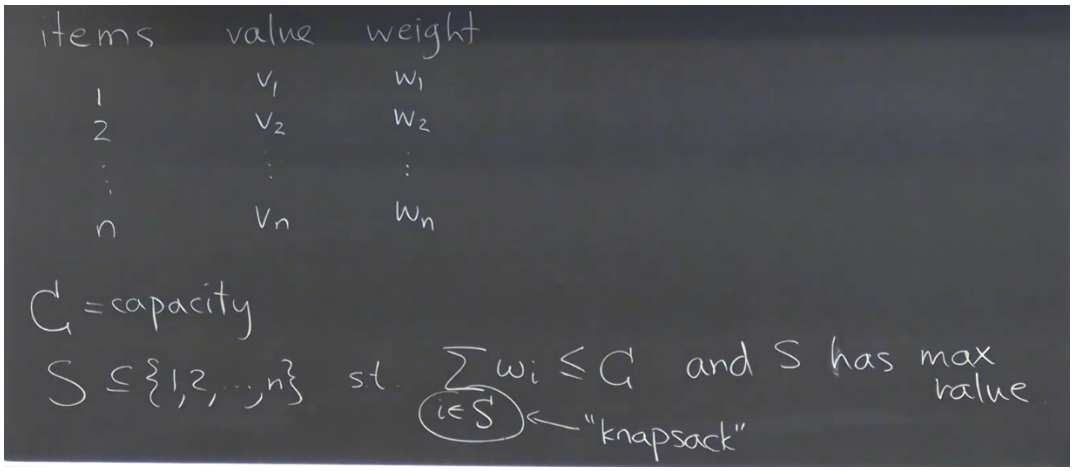
Frank

December 13, 2021

1 Knapsack PTAS

1.1 Problem

- Summary of the problem:



- Previously from DP: Let $0 \leq i \leq n$, $0 \leq c \leq C$

  Let $k(i, c) =$ max value obtained by knapsack of items $1..i$ of capacity c

  Then return $k(n, C)$ for our solution

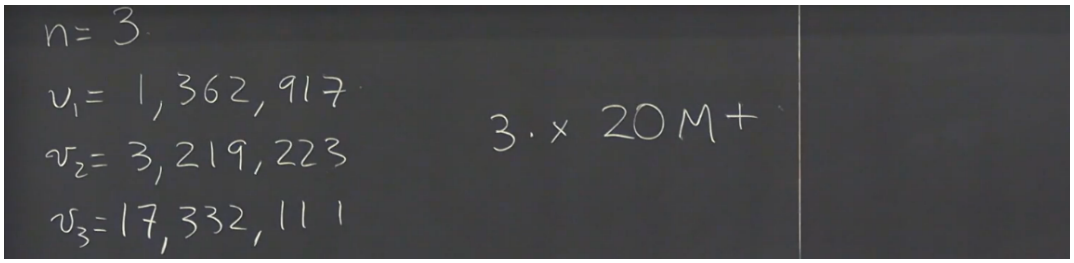
  Also consider $W(i, v) =$ minimum weight required by a knapsack of items $1..i$ to achieve value $\geq v$

  Then return max v s.t. $W(n, v) \leq C$

  The weight version of the algorithm has complexity $O(n \cdot \sum_{i=1}^{n} v_i)$

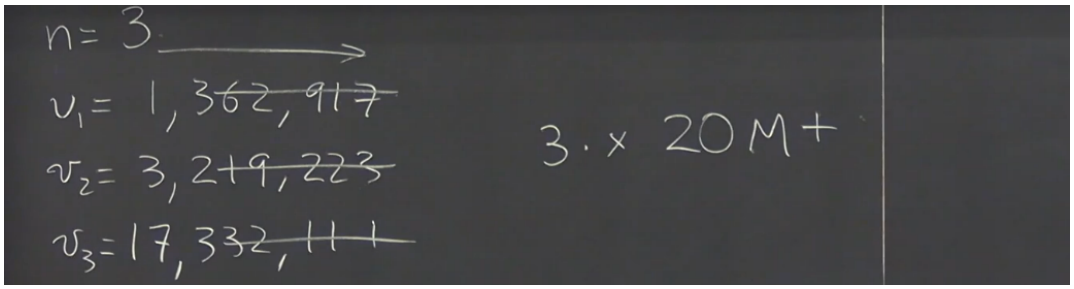
1.2 Approximation Algorithm

- For example, given this example:



We would have terrible time complexity, since the complexity depends on the weights of the items.

But consider removing the "details", the digits starting from the right. Then we can scale down the time complexity very quickly.



- But how much do we remove? It depends on the number of items and how much details we need to accurately represent the items's weights.

1: **procedure** D-KNAPSACK
2: Discard all items with weight $> C$ ▷ will never pick these items
3: $v_{max} \leftarrow max_{1 \leq i \leq n} v_i$
4: $\hat{v}_i \leftarrow \left\lfloor \frac{v_i}{v_{max}} \cdot \frac{n}{\epsilon} \right\rfloor$ ▷ "scaled values", where ϵ is the desired accuracy
5: run DP algorithm using scaled values $\hat{v}_i$
6: **return** set $\hat{S}$ that is optimal for scaled values

---

- Where $\frac{v_i}{v_{max}}$ is the scale down factor that reduces the weights of the items to a fractional value and $\frac{n}{\epsilon}$ is the scale up factor that makes the weights distinguishible from one another.

1.3 Running Time

- Dominated by running DP algorithm
- Consider $\hat{v}_i = \left\lfloor \frac{v_i}{v_{max}} \cdot \frac{n}{\epsilon} \right\rfloor \leq \frac{n}{\epsilon}$

  $\sum_{i=1}^{n} \hat{v}_i \leq \frac{n^2}{\epsilon} \implies$ Running time is $O(n \cdot \sum \hat{v}_i) = O(\frac{n^3}{\epsilon})$ (Fully polynomial)

1.4 Error Analysis

- Let $\hat{S}$ = set of items returned by our algorithm
- Let $S^\star$ = optimal knapsack
- Want: $\sum_{i \in \hat{S}} v_i \geq (1-\epsilon) \sum_{i \in S^\star} v_i$

  Let $\sum_{i \in S^\star} v_i = k^\star$
- Let $\hat{v}_i = \left\lfloor \frac{v_i}{v_{max}} \cdot \sigma \right\rfloor$

  $\implies \hat{v}_i \leq \frac{v_i}{v_{max}} \cdot \sigma$

  $\implies v_i \geq \frac{v_{max}}{\sigma} \cdot \hat{v}_i$
- Then consider that $\sum_{i \in \hat{S}} v_i \geq \sum_{i \in \hat{S}} \frac{v_{max}}{\sigma} \cdot \hat{v}_i = \frac{v_{max}}{\sigma} \cdot \sum_{i \in \hat{S}} \hat{v}_i$

  Claim that $\frac{v_{max}}{\sigma} \cdot \sum_{i \in \hat{S}} \hat{v}_i \geq \frac{v_{max}}{\sigma} \cdot \sum_{i \in S^\star} \hat{v}_i$, which holds because $\hat{S}$ is optimal for scaled values $\hat{v}_i$'s
- Then see that $\frac{v_{max}}{\sigma} \cdot \sum_{i \in S^\star} \hat{v}_i = \frac{v_{max}}{\sigma} \sum_{i \in S^\star} \left\lfloor \frac{v_i}{v_{max}} \cdot \sigma \right\rfloor \geq \frac{v_{max}}{\sigma} \sum_{i \in S^\star} (\frac{v_i}{v_{max}} \sigma - 1)$

  $= \frac{v_{max}}{\sigma} \sum_{i \in S^\star} \frac{\sigma}{v_{max}} \cdot v_i - \frac{v_{max}}{\sigma} \sum_{i \in S^\star} 1$

  $= \sum_{i \in S^\star} v_i - \frac{v_{max}}{\sigma} |S^\star|$

  $= k^\star - \frac{v_{max}}{\sigma} |S^\star|$

  Note that $v_{max} \leq k^\star$ and $|S^\star| = n$, then we get

  $\geq k^\star - k^\star \frac{n}{\sigma} = k^\star (1 - \frac{n}{\sigma})$

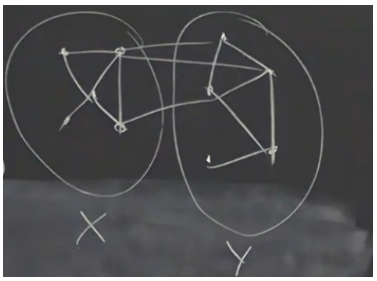
  Choose $\epsilon = \frac{n}{\sigma} \implies \sigma = \frac{n}{\epsilon}$

  Then we get $= k^\star (1 - \epsilon)$, as wanted.
- Then for example, if we want the solution to be within $\frac{1}{100}$ of optimal, then we would get a complexity constant of 100 ($\epsilon = \frac{1}{100}$). If we want the solution to be within $\frac{1}{n}$ of optimal, then we would get a complexity of $O(\frac{n^3}{\frac{1}{n}}) = O(n^4)$

2 Max Cut Approximation (via Local Search)

2.1 Problem

- Cut of undirected graph $G = (V, E)$
- Partition (X, Y) of V



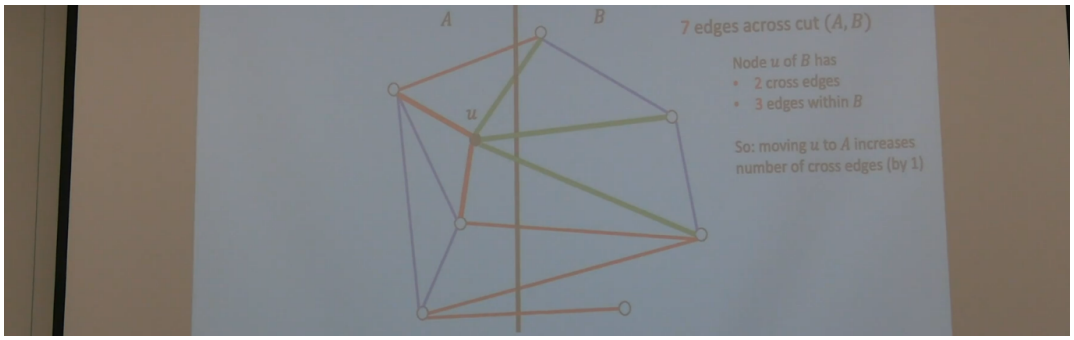
  Where cross edges of $(X, Y) = \{e = \{u, v\} : u \in X, v \in Y\}$
- Max cut is cut with maximum number of edges
- **Input**: Undirected graph $G = (V, E)$
- **Output**: Max cut (X, Y) of G
- NP-hard problem
- 2-approximation algorithm based on local search

```
1: procedure LOCALSEARCH
2:     Start with any feasible solution S
3:     while (∃ better feasible solution S' "near" S) do
4:         S ← S'
5:     return S
```

- Where better means that the solution improves objective function
- The issue with local search is that termination may be an issue, and that the algorithm produces local optimal in general, not necessarily global optimal.
- Example local search algorithms are:
  - Ford-Fulkerson
  - Simplex algorithm

2.2 Algorithm

- Move nodes across the cut and see if we increase the net number of cross edges



Repeat this process until we have no more nodes to move across the cut that increases our net number of cross edges

---

1: **procedure** MAXCUT
2: $(X, Y) \leftarrow$ arbitrary cut of G
3: **while** ($\exists v$ s.t. moving v to other component of (X, Y) increases # cross edges) **do**
4: $(X, Y) \leftarrow$ cut resulting from moving v to other component
5: **return** (X, Y)

- Termination: after $\leq |E| = m$ iterations
- Running Time: $O(m(n + m))$

2.3 Error Analysis

- Let $(X, Y) =$ cut returned by our algorithm
- Want: number of cross edges of $(X, Y) \geq \frac{1}{2}$ number of cross edges of the maximum cut

  Proof:

  - Let $E^c =$ set of cross edges w.r.t. (X, Y)
  - $\forall v \in V$, $E_v =$ set of edges incident on v
  - $E_v^c = E_v \cap E^c$ [cross edges incident on v]
  - $E_v^{in} = E_v - E_v^c$ [internal edges incident on v]
  - Consider that $\forall v$:
    * $|E_v| = |E_v^c| + |E_v^{in}|$
    * $|E_v^{in}| \leq |E_v^c|$
      $\implies |E_v| \leq 2 \cdot |E_v^c|$
      $\implies \sum_{v \in V} |E_v| \leq 2 \cdot \sum_{v \in V} |E_v^c| = 2 \cdot (2 \cdot |E^c|)$, since we are counting each cross edge twice, once from X side and once from Y side
    * Also consider that $\sum_{v \in V} |E_v| = 2 \cdot |E|$, since we are counting each edge twice
      $\implies |E^c| \geq \frac{1}{2}|E|$ and $|E| \geq$ number of cross edges of max cut

2.4 Weighted Max Cut

- edges have weights
- weight of cut = sum of weights of cross edges
- Want: Max weighted cut
- Can use same algorithm, but algorithm is now pseudopolynomial.
- polynomial time approximation algorithm based on similar idea, but is much more intricate.