CS 341: ALGORITHMS

Lecture 6: greedy algorithms II

Readings: see website

Trevor Brown
https://student.cs.uwaterloo.ca/~cs341
trevor.brown@uwaterloo.ca
OPTIMALITY PROOF
for greedy interval selection
Goal: choose as many disjoint intervals as possible, (i.e., without any overlap)

Algorithm:

Sort the intervals in increasing order of finishing times. At any stage, choose the earliest finishing interval that is disjoint from all previously chosen intervals (i.e., the local evaluation criterion is $f_i$).
PROVING OPTIMALITY

- Consider an input $A[1..n]$
- Let $G$ be the greedy solution
- Let $O$ be an optimal solution
- “Greedy stays ahead” argument

  - Intuition: out of the a given set of intervals, greedy picks as many as optimal
How to compare G and O? Imagine reordering O to match G!
CRUCIAL: We are NOT assuming the optimal algorithm uses the same sort order!

We are merely imagining reordering the intervals chosen by the optimal algorithm so we can easily compare their finish times to intervals in G.
Now $O'$ and $G$ are both ordered by increasing finish time.

This ordering helps us leverage what we know about $G$ in our comparison with $O'$.

Argue for a prefix of the intervals sorted this way, $G$ chooses as many as $O'$. 
COMPARING O’ WITH G

Looks like $f(G_1) \leq f(O'_1)$ and $f(G_2) \leq f(O'_2)$ … Is $f(G_i) \leq f(O'_i)$ for all $i$?

If this trend holds in general, then out of the intervals with finish time $\leq f(O'_i)$

G chooses as many intervals as O!
**PROVING LEMMA:** \( f(G_i) \leq f(O'_i) \) FOR ALL \( i \)

<table>
<thead>
<tr>
<th>( O' )</th>
<th>( O'_1 )</th>
<th>( O'_2 )</th>
<th>( \ldots )</th>
<th>( O'_{i-1} )</th>
<th>( O'_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G )</td>
<td>( G_1 )</td>
<td>( G_2 )</td>
<td>( \ldots )</td>
<td>( G_{i-1} )</td>
<td></td>
</tr>
</tbody>
</table>

Base case: \( f(G_1) \leq f(O'_1) \) since \( G \) chooses the interval with the earliest finish time first.
**PROVING LEMMA:** \( f(G_i) \leq f(O_i') \) FOR ALL \( i \)

<table>
<thead>
<tr>
<th>O'</th>
<th>( O'_1 )</th>
<th>( O'_2 )</th>
<th>...</th>
<th>( O'_{i-1} )</th>
<th>( O'_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>( G_1 )</td>
<td>( G_2 )</td>
<td>...</td>
<td>( G_{i-1} )</td>
<td></td>
</tr>
</tbody>
</table>

Inductive step: assume \( f(G_{i-1}) \leq f(O'_{i-1}) \). Show \( f(G_i) \leq f(O'_i) \).

- Since \( O' \) is feasible, \( f(O'_{i-1}) \leq s(O'_i) \)
- So \( f(G_{i-1}) \leq s(O'_i) \)
- So \( G \) can choose \( O'_i \) if it has the smallest finish time
- So \( f(G_i) \leq f(O'_i) \)
Suppose $|O'| > |G|$ to obtain a contradiction

- So if $G$ chooses $k$ intervals, $O'$ chooses at least $k + 1$
- By the lemma, $f(G_k) \leq f(O_k)$
- Since $O'$ is feasible, $f(O'_k) \leq s(O'_{k+1})$
- But then $G$ can, and would, pick $O'_{k+1}$

- Contradiction!

So assumption $|O'| > |G|$ is wrong!

So $G$ is optimal
A DIFFERENT PROOF

“Slick” ad-hoc approaches are sometimes possible…
Let \( F = \{f_{i_1}, \ldots, f_{i_k}\} \) be the finishing times of the intervals in \( X \)

No interval finishes strictly to the left

No interval starts strictly to the right

Would be chosen by greedy! (contradiction)

No interval in \( X \) is strictly between these points!

So, in addition to the intervals in \( X \), only the following types of intervals are possible

Contains \( f_{i_1} \)

Contains \( f_{i_2} \)

Contains \( f_{i_1} \) and \( f_{i_2} \)

Thus, every interval contains some finishing time in \( F \)

And, two intervals in \( O \) cannot contain the same element of \( F \)

So, there must be as many finishing times in \( F \) as there are intervals in \( O \). QED
KNAPSACK PROBLEMS
Knapsack

Instance: Profits $P = [p_1, \ldots, p_n]$; weights $W = [w_1, \ldots, w_n]$; and a capacity, $M$. These are all positive integers.

Feasible solution: An $n$-tuple $X = [x_1, \ldots, x_n]$ where $\sum_{i=1}^{n} w_ix_i \leq M$. 

Gotta respect the weight limit $M$...
Problem 4.4

Knapsack

Instance: Profits $P = [p_1, \ldots, p_n]$; weights $W = [w_1, \ldots, w_n]$; and a capacity, $M$. These are all positive integers.

Feasible solution: An $n$-tuple $X = [x_1, \ldots, x_n]$ where $\sum_{i=1}^{n} w_i x_i \leq M$.

In the 0-1 Knapsack problem (often denoted just as Knapsack), we require that $x_i \in \{0, 1\}$, $1 \leq i \leq n$.

In the Rational Knapsack problem, we require that $x_i \in \mathbb{Q}$ and $0 \leq x_i \leq 1$, $1 \leq i \leq n$.

Find: A feasible solution $X$ that maximizes $\sum_{i=1}^{n} p_i x_i$.

0-1 Knapsack: NP Hard. Probably requires exponential time to solve...

Rational knapsack: Can be solved in polynomial time by a greedy alg!

Lets discuss this now… other one later
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

- **Strategy 1**: consider items in **decreasing** order of **profit** (i.e., we maximize the local evaluation criterion \( p_i \))

- Let’s try an example input
  - Profits \( P = [20, 50, 100] \)
  - Weights \( W = [10, 20, 10] \)
  - Weight limit \( M = 10 \)

- Algorithm selects last item for 100 profit
  - Looks optimal in this example
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

- **Strategy 1**: consider items in **decreasing** order of profit (i.e., we maximize the local evaluation criterion $p_i$)
- How about a **second example input**
  - Profits $P = [20, 50, 100]$
  - Weights $W = [10, 20, 100]$
  - Weight limit $M = 10$
- Algorithm selects last item for 10 profit
  - **Not optimal!**
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

- **Strategy 2:** consider items in increasing order of weight (i.e., we minimize the local evaluation criterion $w_i$)

- **Counterexample**
  - Profits $P = [20, 50, 100]$
  - Weights $W = [10, 20, 100]$
  - Weight limit $M = 10$
  - Algorithm selects first item for 20 profit
    - It **could** select half of second item, for 25 profit!
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

- **Strategy 3:** consider items in decreasing order of profit divided by weight (i.e., we maximize local evaluation criterion $p_i/w_i$)

- Let’s try our first example input
  - Profits $P = [20, 50, 100]$
  - Weights $W = [10, 20, 10]$
  - Weight limit $M = 10$

- Profit divided by weight
  - $P/W = [2, 2.5, 10]$

- Algorithm selects last item for 100 profit (optimal)
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

- **Strategy 3**: consider items in **decreasing** order of **profit divided by weight** (i.e., we maximize local evaluation criterion $p_i/w_i$)

Let’s try our second example input

- Profits $P = [20, 50, 100]$
- Weights $W = [10, 20, 100]$
- Weight limit $M = 10$

- Profit divided by weight
  - $P/W = [2, 2.5, 1]$
- Algorithm selects second item for 25 profit (optimal)

It turns out strategy #3 is optimal...
Preprocess(A[1..n], M) // A[i] = (p_i, w_i)
sort A by decreasing profit divided by weight
let p[1..n] be the profits in A
let w[1..n] be the weights in A
return GreedyRationalKnapsack(p, w, M)

GreedyRationalKnapsack(p[1..n], w[1..n], M)
X = [0, ..., 0]  // No items are chosen yet
weight = 0  // Current weight of knapsack

for i = 1..n  // For all items
  if weight + w[i] > M then
    X[i] = (M - weight) / w[i]
    break
  else
    X[i] = 1
    weight = weight + w[i]

return X

Either X=(1,1,...,1,0,...,0) or X=(1,1,...,1,x_i,0,...,0) where x_i ∈ (0,1)
Preprocess(A[1..n], M) // A[i] = (p_i, w_i)
    sort A by decreasing profit divided by weight
    let p[1..n] be the profits in A
    let w[1..n] be the weights in A
    return GreedyRationalKnapsack(p, w, M)

GreedyRationalKnapsack(p[1..n], w[1..n], M)
    X = [0, ..., 0]
    weight = 0
    for i = 1..n
        if weight + w[i] > M then
            X[i] = (M - weight) / w[i]
            break
        else
            X[i] = 1
            weight = weight + w[i]
    return X

Running time complexity?
Can do preprocessing in $\Theta(n \log n)$
Create array in $\Theta(n)$ time
$\Theta(n)$ iterations each doing $\Theta(1)$ work
Total $\Theta(n \log n)$ (or $\Theta(n)$ if input is already sorted)
INFORMAL FEASIBILITY ARGUMENT
(should be good enough to show feasibility on assessments)

- **Feasibility**: all $x_i$ are in $[0, 1]$ and total weight is $\leq M$
- Either everything fits in the knapsack, or:
- When we exit the loop, **weight is exactly $M$**
- Every time we write to $x_i$ it’s either 0, 1 or
  $(M - \text{weight})/w_i$ where $\text{weight} + w[i] > M$
  - Rearranging the latter we get $(M - \text{weight})/w_i < 1$
  - And $\text{weight} \leq M$, so $(M - \text{weight})/w_i \geq 0$
  - **So, we have** $x_i \in [0, 1]$
MINOR MODIFICATION TO FACILITATE **FORMAL** PROOF

```
GreedyRationalKnapsack(p[1..n], w[1..n], M)

X = [0, ..., 0]
weight = 0

for i = 1..n
    if weight + w[i] > M then
        X[i] = (M - weight) / w[i]
        weight = M
        break
    else
        X[i] = 1
        weight = weight + w[i]

return X
```

Does NOT change behaviour of the algorithm at all!

Optional slide, just for your notes
FORMAL FEASIBILITY ARG

- Loop invariant: \( \forall i : x_i \in [0,1] \)
  - and \( \text{weight} = \sum_{i=1}^{n} w_i x_i \leq M \)

- Base case. Initially \( \text{weight} = 0 \) and \( \forall i : x_i = 0 \).
  - So \( 0 = \text{weight} = \sum_{i=1}^{n} w_i \cdot 0 = \sum_{i=1}^{n} w_i x_i \leq M \)

- Inductive step.
  - Suppose invariant holds at start of iteration \( i \)
  - Let \( \text{weight}', x_i' \) denote values of \( \text{weight}, x_i \) at end of iteration \( i \)
  - Prove invariant holds at end of iteration \( i \)
  - i.e., \( \forall i : x_i' \in [0,1] \) and \( \text{weight}' = \sum_{i=1}^{n} w_i x_i' \leq M \)
**FORMAL FEASIBILITY ARG**

- **WTP:** \( \forall i : x'_i \in [0, 1] \)
  and \( weight' = \sum_{i=1}^{n} w_i x'_i \leq M \)

- **Case 1:** \( weight + w_i \leq M \)
  - \( x'_i = 1 \) which is in \([0, 1]\) (by line 11)
  - \( weight' = weight + w_i \) (by line 12)
    and this is \( \leq M \) by the case
  - \( weight' = \sum_{k=1}^{n} x_k w_k + w_i \) (by invariant)
  - \( weight' = \sum_{k=1}^{n} x_k w_k + x'_i w_i \) (since \( x'_i = 1 \))
  - And \( x'_k = x_k \) for all \( k \neq i \) and \( x_i = 0 \) so \( \sum_{k=1}^{n} x'_k w_k = x'_i w_i + \sum_{k=1}^{n} x_k w_k \)
  - Rearrange to get \( \sum_{k=1}^{n} x_k w_k = (\sum_{k=1}^{n} x'_k w_k - x'_i w_i) \)
  - So \( weight' = (\sum_{k=1}^{n} x'_k w_k - x'_i w_i) + x'_i w_i = \sum_{k=1}^{n} x'_k w_k \)
WTP: $\forall i : x'_i \in [0, 1]$
and $weight' = \sum_{i=1}^{n} w_i x'_i \leq M$

Case 2: $weight + w_i > M$

- We have $w_i > M - weight$ (by case)
  and $M - weight \geq 0$ (by invariant)

- So $0 \leq \frac{M - weight}{w_i} < 1$ which means $x'_i \in [0, 1)$

- $weight' = M = weight + (M - weight)$ (by line 8)

- $weight' = \sum_{k=1}^{n} x_k w_k + (M - weight)$ (by invariant)

- But $x'_k = x_k$ for all $k \neq i$ and $x_i = 0$ so
  $\sum_{k=1}^{n} x'_k w_k = x'_i w_i + \sum_{k=1}^{n} x_k w_k$

- Rearrange to get $\sum_{k=1}^{n} x_k w_k = (\sum_{k=1}^{n} x'_k w_k - x'_i w_i)$

- So $weight' = (\sum_{k=1}^{n} x'_k w_k - x'_i w_i) + (M - weight)$

- And $M - weight = x'_i w_i$ so $weight' = \sum_{k=1}^{n} x'_k w_k$
EXCHANGE ARGUMENT
for proving optimality
For simplicity, assume that the profit / weight ratios are all distinct, so
\[ \frac{p_1}{w_1} > \frac{p_2}{w_2} > \ldots > \frac{p_n}{w_n}. \]
Suppose the greedy solution is \( X = (x_1, \ldots, x_n) \) and the optimal solution is \( Y = (y_1, \ldots, y_n) \).
We will prove that \( X = Y \), i.e., \( x_j = y_j \) for \( j = 1, \ldots, n \). Therefore there is a unique optimal solution and it is equal to the greedy solution.

Suppose \( X \neq Y \). To obtain a contradiction

Pick the smallest integer \( j \) such that \( x_j \neq y_j \). \( X \) and \( Y \) are identical up to \( x_j \) and \( y_j \), respectively
What’s the relationship between $x_j$ and $y_j$?
j = first index where the solutions differ

Can we have \( y_j > x_j \)?

No! Greedy would take more of item \( j \) if it could.
Greedy solution $X$

Optimal solution $Y$

$j$ = first index where the solutions differ

Must have $y_j < x_j$
Greedy solution $X$

Optimal solution $Y$

$j = \text{first index where the solutions differ}$

No! It would be worth less than $X$

Can $Y$ be all zeros after $y_j$?
Greedy solution $X$

Optimal solution $Y$

$j$ = first index where the solutions differ

Must exist $k > j$ such that $y_k > 0$

But, by our sort order, item $j$ is worth more (per unit of weight) than item $k$!

Remove some of item $k$ and replace it with some of item $j$?

How much of item $k$ should we remove?
Since item j is worth **more per unit weight**, replacing **even a tiny amount** of item k with item j will improve the solution.

So, we remove an infinitesimal $\delta > 0$ of weight of item k, and add $\delta$ weight of item j.
Greedy solution X

Optimal solution Y

\( j = \) first index where the solutions differ

To move \( \delta \) weight from item \( k \) to item \( j \)...

What fraction of item \( j \) are we adding?

\( y'_j = y_j + \frac{\delta}{w_j} \)

What fraction of item \( k \) are we removing?

\( y'_k = y_k - \frac{\delta}{w_k} \)

Modified optimal solution \( Y' \)

fraction of item in knapsack

1

0

Item 1

Item 2

Item \( j \)

Item n

0

1

1

0

Item 1

Item 2

Item \( j \)

Item n

0

1

1

0

Item 1

Item 2

Item \( j \)

Item n

0

1

1

0
The idea is to show that

\( Y' \) is feasible, and

\[ \text{profit}(Y') > \text{profit}(Y) \]

This contradicts the optimality of \( Y \) and proves that \( X = Y \).

To show \( Y' \) is feasible, we show \( y'_k \geq 0, y'_j \leq 1 \) and \( \text{weight}(Y') \leq M \).
FEASIBILITY OF $Y'$

- To show $Y'$ is feasible, we show $y_k' \geq 0$, $y_j' \leq 1$ and weight($Y'$) $\leq M$
- Let's show $y_k' \geq 0$
  - By definition, $y_k' = y_k - \frac{\delta}{w_k}$
  - So, $y_k' \geq 0$ iff $y_k - \frac{\delta}{w_k} \geq 0$ iff $\delta \leq y_k w_k$
  - And we know $y_k$ and $w_k$ are both positive
  - So, this constrains $\delta$ to be smaller than this positive number
  - Therefore, it is possible to choose positive $\delta$ s.t. $y_k' \geq 0$

Existence proof, but a non-constructive one
FEASIBILITY OF $Y'$

- To show $Y'$ is feasible, we show $y'_k \geq 0$, $y'_j \leq 1$ and $\text{weight}(Y') \leq M$
- Now let's show $y'_j \leq 1$
  - By definition, $y'_j = y_j + \frac{\delta}{w_j}$
  - So, $y'_j \leq 1$ iff $y_j + \frac{\delta}{w_j} \leq 1$ iff $\delta \leq (1 - y_j)w_j$
  - Recall $y_j < x_j$, so $y_j < 1$, which means $(1 - y_j) > 0$
  - So, this constrains $\delta$ to be smaller than some positive number
Finally, we show $\text{weight}(Y') \leq M$

Recall changes to get $Y'$ from $Y$

- We move $\delta$ weight from item $k$ to item $j$
- This does not change the total weight!

So $\text{weight}(Y') = \text{weight}(Y) \leq M$

Therefore, $Y'$ is feasible!
SUPERIORITY OF $Y'$

- Finally we compute $\text{profit}(Y')$

$$\text{profit}(Y') = \text{profit}(Y) + \frac{\delta}{w_j} p_j - \frac{\delta}{w_k} p_k$$

- $= \text{profit}(Y) + \delta \left(\frac{p_j}{w_j} - \frac{p_k}{w_k}\right)$

- Since $j$ is before $k$, and we consider items with more profit per unit weight first, we have $\frac{p_j}{w_j} > \frac{p_k}{w_k}$.

- So, if $\delta > 0$ then $\delta \left(\frac{p_j}{w_j} - \frac{p_k}{w_k}\right) > 0$

- Since we can choose $\delta > 0$, we have $\text{profit}(Y') > \text{profit}(Y)$.

(Fraction of item $j$ added) \times (profit for item $j$)

(Fraction of item $k$ removed) \times (profit for item $k$)

Contradicts optimality of $Y$!
So assumption $X \neq Y$ is bad. Therefore, $X$ is optimal.
WHAT IF ELEMENTS DON’T HAVE DISTINCT PROFIT/WEIGHT RATIOS?

Covering the next 9 slides is homework!
OPTIMALITY PROOF WITHOUT DISTINCTNESS

- There may be many optimal solutions

- **Key idea:** Let $Y$ be an optimal solution that **matches** $X$ on a maximal number of indices

- **Observe:** if $X$ is really optimal, then $Y = X$

- Suppose not for contra
  - We will modify $Y$, preserving its optimality, but making it match $X$ on **one more index** (a contradiction!)
Greedy solution $X$

Optimal solution $Y$

$j =$ first index where the solutions differ

$y_j \neq x_j$
Greedy solution $X$

Optimal solution $Y$

fraction of item in knapsack

Item 1  Item 2  ...  Item j

$0 \leq x_1 \leq x_2 \leq \ldots \leq x_j \leq 1$

Item 1  Item 2  ...  Item j

$0 \leq y_1 \leq y_2 \leq \ldots \leq y_{j-1} < y_j \leq 1$

Must have $y_j < x_j$
Must exist \( k > j \) such that \( y_k > x_k \) because weight of \( X \) and \( Y \) must be the same.

**Remove** some weight \( \delta \) of item \( k \) and **add** the same weight of item \( j \)

With the goal of making the solutions equal on index \( k \) or index \( j \)

\[
\text{Fraction we should add to } j \text{ to make solutions equal on index } j: x_j - y_j
\]

\[
\text{Weight to add: } w_j(x_j - y_j)
\]

\[
\text{Weight to remove: } w_k(y_k - x_k)
\]

Let \( \delta = \min\{w_j(x_j - y_j), w_k(y_k - x_k)\} \)

Observe \( \delta > 0 \)
Greedy solution $X$

Optimal solution $Y$

Suppose $\delta = w_k(y_k - x_k)$

Modified optimal solution $Y'$

In this case, since $\delta = w_k(y_k - x_k)$, we end up with $y'_k = x_k$

If $\delta$ were $w_j(x_j - y_j)$, we would have $y'_j = x_j$
To show $Y'$ is feasible, we show $\text{weight}(Y') \leq M$ and $y_k' \geq 0, y_j' \leq 1$.

**Modified optimal solution $Y'$**

- We move $\delta$ weight from item $k$ to item $j$.
- This does not change the total weight!
- So $\text{weight}(Y') = \text{weight}(Y) = M$.
FEASIBILITY OF $Y'$

- Showing $y_k' \geq 0$
  - By definition, $y_k' = y_k - \frac{\delta}{w_k} \geq 0$ iff $\delta \leq y_k w_k$
  - But $\delta$ is the minimum of $w_j(x_j - y_j)$ and $w_k(y_k - x_k) \leq w_k y_k$
  - And $w_k(y_k - x_k) \leq w_k y_k$ so $\delta \leq y_k w_k$

- Showing $y_j' \leq 1$
  - $y_j' = y_j + \frac{\delta}{w_j} \leq 1$ iff $\frac{\delta}{w_j} \leq 1 - y_j$ iff $\delta \leq w_j(1 - y_j)$ (rearranging)
  - $\delta \leq w_j(x_j - y_j)$ (definition of $\delta$)
  - and $w_j(x_j - y_j) \leq w_j(1 - y_j)$ (by feasibility of $X$, i.e., $x_j \leq 1$)
PROFIT OF $Y'$

- $\text{profit}(Y') = \text{profit}(Y) + \frac{\delta}{w_j} p_j - \frac{\delta}{w_k} p_k = \text{profit}(Y) + \delta \left( \frac{p_j}{w_j} - \frac{p_k}{w_k} \right)$

- Since $j$ is before $k$, and we consider items with more profit per unit weight first, we have $\frac{p_j}{w_j} \geq \frac{p_k}{w_k}$.

- Since $\delta > 0$ and $\frac{p_j}{w_j} \geq \frac{p_k}{w_k}$, we have $\delta \left( \frac{p_j}{w_j} - \frac{p_k}{w_k} \right) \geq 0$

- Since $Y$ is optimal, this cannot be positive

- So $Y'$ is a new optimal solution that matches $X$ on one more index than $Y$

- Contradiction: $Y$ matched $X$ on a maximal number of indices!
SUMMARIZING EXCHANGE ARGUMENTS

- If inputs are distinct
  - So there is a unique optimal solution
  - Let $O \neq G$ be an optimal solution that beats greedy
  - Show how to change $O$ to obtain a better solution

- If not
  - There may be many optimal solutions
  - Let $O \neq G$ be an optimal solution that matches greedy on as many choices as possible
  - Show how to change $O$ to obtain an optimal solution $O'$ that matches greedy for even more choices