

Greedy algorithms

David Kauchak
cs302
Spring 2012



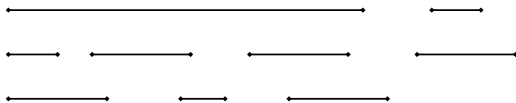
Administrative

- Should be all caught up on grading
- Assignment out today (back to the normal routine)



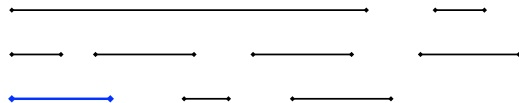
Interval scheduling

- Given n activities $A = [a_1, a_2, \dots, a_n]$ where each activity has start time s_i and a finish time f_i . Schedule as many as possible of these activities such that they don't conflict.



Interval scheduling

- Given n activities $A = [a_1, a_2, \dots, a_n]$ where each activity has start time s_i and a finish time f_i . Schedule as many as possible of these activities such that they don't conflict.

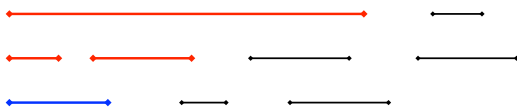


Which activities conflict?



Interval scheduling

Given n activities $A = [a_1, a_2, \dots, a_n]$ where each activity has start time s_i and a finish time f_i . Schedule *as many as possible* such that they don't *conflict*.



Which activities conflict?

Simple recursive solution

Enumerate all possible solutions and find which schedules the most activities

```

INTERVALSCHEDULE-RECURSIVE(A)
1  if A = {}
2     return 0
3  else
4     max = -∞
5     for all a ∈ A
6         A' ← A minus a and all conflicting activities with a
7         s = INTERVALSCHEDULE-RECURSIVE(A')
8         if s > max
9             max = s
10    return 1 + max
  
```

Simple recursive solution

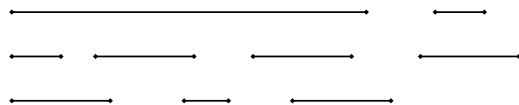
- Is it correct?
 - max{all possible solutions}
- Running time?
 - $O(n!)$

```

INTERVALSCHEDULE-RECURSIVE(A)
1  if A = {}
2     return 0
3  else
4     max = -∞
5     for all a ∈ A
6         A' ← A minus a and all conflicting activities with a
7         s = INTERVALSCHEDULE-RECURSIVE(A')
8         if s > max
9             max = s
10    return 1 + max
  
```

Can we do better?

- Dynamic programming (next class)
 - $O(n^2)$
- Greedy solution – Is there a way to repeatedly make local decisions?
 - Key: we'd still like to end up with the *optimal* solution

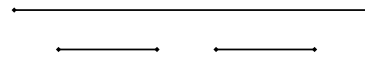


Overview of a greedy approach

- Greedily pick an activity to schedule
- Add that activity to the answer
- Remove that activity and all conflicting activities. Call this A' .
- Repeat on A' until A' is empty

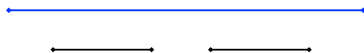
Greedy options

- Select the activity that starts the earliest, i.e. $\operatorname{argmin}\{s_1, s_2, s_3, \dots, s_n\}$?



Greedy options

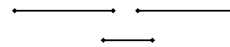
- Select the activity that starts the earliest?



non-optimal

Greedy options

- Select the shortest activity, i.e. $\operatorname{argmin}\{f_1-s_1, f_2-s_2, f_3-s_3, \dots, f_n-s_n\}$



Greedy options

- Select the shortest activity, i.e. $\operatorname{argmin}\{f_1-s_1, f_2-s_2, f_3-s_3, \dots, f_n-s_n\}$

non-optimal

Greedy options

- Select the activity with the smallest number of conflicts

Greedy options

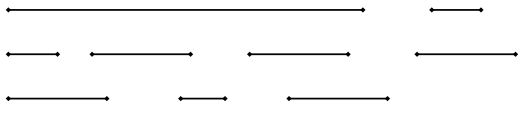
- Select the activity with the smallest number of conflicts

Greedy options

- Select the activity with the smallest number of conflicts

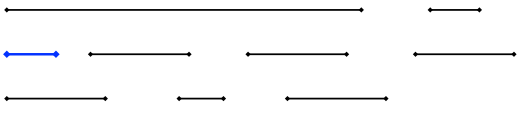
Greedy options

- Select the activity that ends the earliest, i.e. $\operatorname{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?



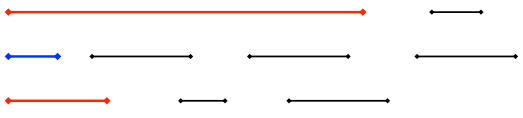
Greedy options

- Select the activity that ends the earliest, i.e. $\operatorname{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?



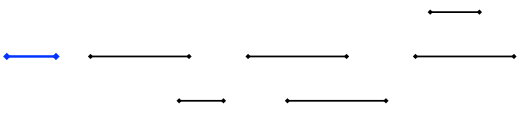
Greedy options

- Select the activity that ends the earliest, i.e. $\operatorname{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?



Greedy options

- Select the activity that ends the earliest, i.e. $\operatorname{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?



Greedy options

- Select the activity that ends the earliest, i.e. $\text{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?

Greedy options

- Select the activity that ends the earliest, i.e. $\text{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?

Greedy options

- Select the activity that ends the earliest, i.e. $\text{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?

Greedy options

- Select the activity that ends the earliest, i.e. $\text{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?

Greedy options

- Select the activity that ends the earliest, i.e. $\operatorname{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?



Greedy options

- Select the activity that ends the earliest, i.e. $\operatorname{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?



Multiple optimal solutions

Greedy options

- Select the activity that ends the earliest, i.e. $\operatorname{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?



Greedy options

- Select the activity that ends the earliest, i.e. $\operatorname{argmin}\{f_1, f_2, f_3, \dots, f_n\}$?



Efficient greedy algorithm

- Once you've identified a reasonable greedy heuristic:
 - Prove that it always gives the correct answer
 - Develop an efficient solution

Is our greedy approach correct?

"Stays ahead" argument:

show that no matter what other solution someone provides you, the solution provided by your algorithm always "stays ahead", in that no other choice could do better

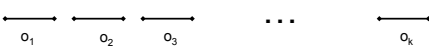
Is our greedy approach correct?

"Stays ahead" argument

- Let $r_1, r_2, r_3, \dots, r_k$ be the solution found by our approach

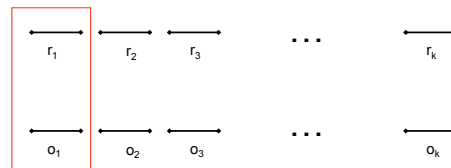


- Let $o_1, o_2, o_3, \dots, o_k$ of another optimal solution



- Show our approach "stays ahead" of any other solution

Stays ahead



Compare first activities of each solution

Stays ahead

$\text{finish}(r_1) \leq \text{finish}(o_1)$

Stays ahead

We have **at least** as much time as any other solution to schedule the remaining 2...k tasks

An efficient solution

```

INTERVALSCHEDULE-GREEDY(A)
1  sort A based on finish times  $f_i$ 
2  for  $i \leftarrow 1$  to  $n$ 
3      add  $a_i$  to  $R$ 
4       $finish \leftarrow f_i$ 
5      while  $s_i < finish$ 
6           $i \leftarrow i + 1$ 
7  return  $R$ 

```

Running time?

```

INTERVALSCHEDULE-GREEDY(A)
1  sort A based on finish times  $f_i$ 
2  for  $i \leftarrow 1$  to  $n$ 
3      add  $a_i$  to  $R$ 
4       $finish \leftarrow f_i$ 
5      while  $s_i < finish$ 
6           $i \leftarrow i + 1$ 
7  return  $R$ 

```

$\Theta(n \log n)$

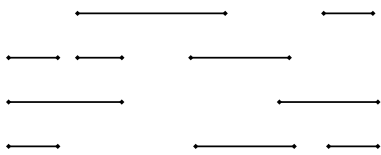
$\Theta(n)$

Better than:
 $O(n!)$
 $O(n^2)$

Overall: $\Theta(n \log n)$

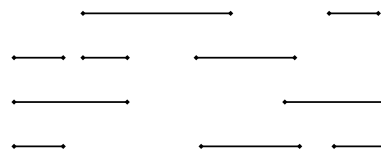
Scheduling *all* intervals

- Given n activities, we need to schedule **all** activities. **Goal:** minimize the number of resources required.

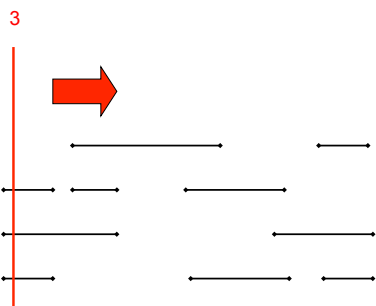


Greedy approach?

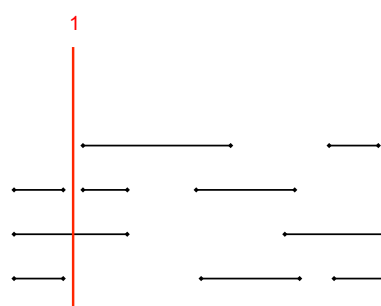
The best we could ever do is the maximum number of conflicts for any time period



Calculating max conflicts efficiently



Calculating max conflicts efficiently



Calculating max conflicts efficiently

A diagram illustrating the calculation of maximum conflicts. A vertical red line is positioned at a point labeled '3'. Three horizontal intervals are shown: one above the line, one to the left of the line, and one to the right of the line. The number '3' is written above the line, indicating the count of intervals intersected at that point.

Calculating max conflicts efficiently

A diagram illustrating the calculation of maximum conflicts. A vertical red line is positioned at a point labeled '1'. One horizontal interval is shown intersecting the line. The number '1' is written above the line, indicating the count of intervals intersected at that point.

Calculating max conflicts efficiently

A diagram illustrating the calculation of maximum conflicts. A vertical red line is positioned at a point labeled '3'. Three horizontal intervals are shown: one above the line, one to the left of the line, and one to the right of the line. The number '3' is written above the line. To the right of the line, there are three red dots, possibly representing a sequence of points or a specific set of intervals.

Calculating max conflicts

```

ALLINTERVALSCHEDULECOUNT(A)
1  Sort the start and end times, call this X
2  current ← 0
3  max ← 0
4  for i ← 1 to length[X]
5      if xi is a start node
6          current ++
7      else
8          current --
9      if current > max
10         max ← current
11 return max
    
```

Correctness?

We can do no better than the max number of conflicts. This exactly counts the max number of conflicts.

```

ALLINTERVALSCHEDULECOUNT(A)
1  Sort the start and end times, call this X
2  current ← 0
3  max ← 0
4  for i ← 1 to length[X]
5      if xi is a start node
6          current ++
7      else
8          current --
9      if current > max
10         max ← current
11  return max
  
```

Runtime?

$O(2n \log 2n + n) = O(n \log n)$

```

ALLINTERVALSCHEDULECOUNT(A)
1  Sort the start and end times, call this X
2  current ← 0
3  max ← 0
4  for i ← 1 to length[X]
5      if xi is a start node
6          current ++
7      else
8          current --
9      if current > max
10         max ← current
11  return max
  
```

Horn formulas

- Horn formulas are a particular form of boolean logic formulas
- They are one approach to allow a program to do logical reasoning
- Boolean variables: represent some event
 - x = the murder took place in the kitchen
 - y = the butler is innocent
 - z = the colonel was asleep at 8 pm

Implications

- Left-hand side is an AND of any number of positive literals
- Right-hand side is a single literal

$$z \wedge y \Rightarrow x$$

If the colonel was asleep at 8 pm **and** the butler is innocent **then** the murder took place in the kitchen

x = the murder took place in the kitchen
y = the butler is innocent
z = the colonel was asleep at 8 pm

Implications

- Left-hand side is an AND of any number of positive literals
- Right-hand side is a single literal

$$\Rightarrow x$$

the murder took place in the kitchen

x = the murder took place in the kitchen
y = the butler is innocent
z = the colonel was asleep at 8 pm

Negative clauses

An OR of any number of negative literals

$$\bar{u} \vee \bar{t} \vee \bar{y}$$

not every one is innocent

u = the constable is innocent
t = the colonel is innocent
y = the butler is innocent

Goal

- Given a horn formula (i.e. set of implications and negative clauses), determine if the formula is satisfiable (i.e. an assignment of true/false that is consistent with all of the formula)

$$\Rightarrow x \quad x \wedge u \Rightarrow z$$

$$\Rightarrow y \quad \bar{x} \vee \bar{y} \vee \bar{z}$$

u x y z
0 1 1 0

Goal

- Given a horn formula (i.e. set of implications and negative clauses), determine if the formula is satisfiable (i.e. an assignment of true/false that is consistent with all of the formula)

$$\Rightarrow x \quad x \wedge y \Rightarrow z$$

$$\Rightarrow y \quad \bar{x} \vee \bar{y} \vee \bar{z}$$

u x y z
not satisfiable

Goal

- Given a horn formula (i.e. set of implications and negative clauses), determine if the formula is satisfiable (i.e. an assignment of true/false that is consistent with all of the formula)

$$\begin{array}{lll} \Rightarrow x & x \wedge z \Rightarrow w & w \wedge y \wedge z \Rightarrow x \\ x \Rightarrow y & x \wedge y \Rightarrow w & \bar{w} \vee \bar{x} \vee \bar{y} \end{array}$$

?

Goal

- Given a horn formula (i.e. set of implications and negative clauses), determine if the formula is satisfiable (i.e. an assignment of true/false that is consistent with all of the formula)

$$x \wedge u \Rightarrow z \quad \text{implications tell us to set some variables to true}$$

$$\bar{x} \vee \bar{y} \vee \bar{z} \quad \text{negative clauses encourage us make them false}$$

A brute force solution

- Try each setting of the boolean variables and see if any of them satisfy the formula
- For n variables, how many settings are there?
 - 2^n

A greedy solution?

$$\begin{array}{lll} \Rightarrow x & x \wedge z \Rightarrow w & w \wedge y \wedge z \Rightarrow x \\ x \Rightarrow y & x \wedge y \Rightarrow w & \bar{w} \vee \bar{x} \vee \bar{y} \end{array}$$

$$w \quad 0$$

$$x \quad 0$$

$$y \quad 0$$

$$z \quad 0$$

A greedy solution?

$$\Rightarrow x \quad x \wedge z \Rightarrow w \quad w \wedge y \wedge z \Rightarrow x$$

$$x \Rightarrow y \quad x \wedge y \Rightarrow w \quad \bar{w} \vee \bar{x} \vee \bar{y}$$

$$w \ 0$$

$$x \ 1$$

$$y \ 0$$

$$z \ 0$$

A greedy solution?

$$\Rightarrow x \quad x \wedge z \Rightarrow w \quad w \wedge y \wedge z \Rightarrow x$$

$$x \Rightarrow y \quad x \wedge y \Rightarrow w \quad \bar{w} \vee \bar{x} \vee \bar{y}$$

$$w \ 0$$

$$x \ 1$$

$$y \ 1$$

$$z \ 0$$

A greedy solution?

$$\Rightarrow x \quad x \wedge z \Rightarrow w \quad w \wedge y \wedge z \Rightarrow x$$

$$x \Rightarrow y \quad x \wedge y \Rightarrow w \quad \bar{w} \vee \bar{x} \vee \bar{y}$$

$$w \ 1$$

$$x \ 1$$

$$y \ 1$$

$$z \ 0$$

A greedy solution?

$$\Rightarrow x \quad x \wedge z \Rightarrow w \quad w \wedge y \wedge z \Rightarrow x$$

$$x \Rightarrow y \quad x \wedge y \Rightarrow w \quad \bar{w} \vee \bar{x} \vee \bar{y}$$

$$w \ 1$$

$$x \ 1$$

$$y \ 1$$

$$z \ 0$$

not satisfiable

A greedy solution

HORN(H)

```

1 set all variables to false
2 for all implications  $i$ 
3   if EMPTY(LHS( $i$ ))
4     RHS( $i$ )  $\leftarrow$  true
5 changed  $\leftarrow$  true
6 while changed
7   changed  $\leftarrow$  false
8   for all implications  $i$ 
9     if LHS( $i$ ) = true and !RHS( $i$ ) = true
10      RHS( $i$ )  $\leftarrow$  true
11      changed = true
12 for all negative clauses  $c$ 
13   if  $c$  = false
14     return false
15 return true

```



A greedy solution

HORN(H)

```

1 set all variables to false
2 for all implications  $i$ 
3   if EMPTY(LHS( $i$ ))
4     RHS( $i$ )  $\leftarrow$  true
5 changed  $\leftarrow$  true
6 while changed
7   changed  $\leftarrow$  false
8   for all implications  $i$ 
9     if LHS( $i$ ) = true and !RHS( $i$ ) = true
10      RHS( $i$ )  $\leftarrow$  true
11      changed = true
12 for all negative clauses  $c$ 
13   if  $c$  = false
14     return false
15 return true

```

set all variables of
the implications of
the form " $\Rightarrow x$ " to true



A greedy solution

HORN(H)

```

1 set all variables to false
2 for all implications  $i$ 
3   if EMPTY(LHS( $i$ ))
4     RHS( $i$ )  $\leftarrow$  true
5 changed  $\leftarrow$  true
6 while changed
7   changed  $\leftarrow$  false
8   for all implications  $i$ 
9     if LHS( $i$ ) = true and !RHS( $i$ ) = true
10      RHS( $i$ )  $\leftarrow$  true
11      changed = true
12 for all negative clauses  $c$ 
13   if  $c$  = false
14     return false
15 return true

```

if the all variables of
the lhs of an
implication are true,
then set the rhs
variable to true



A greedy solution

HORN(H)

```

1 set all variables to false
2 for all implications  $i$ 
3   if EMPTY(LHS( $i$ ))
4     RHS( $i$ )  $\leftarrow$  true
5 changed  $\leftarrow$  true
6 while changed
7   changed  $\leftarrow$  false
8   for all implications  $i$ 
9     if LHS( $i$ ) = true and !RHS( $i$ ) = true
10      RHS( $i$ )  $\leftarrow$  true
11      changed = true
12 for all negative clauses  $c$ 
13   if  $c$  = false
14     return false
15 return true

```

see if all of the
negative clauses are
satisfied



Correctness of greedy solution

- Two parts:
 - If our algorithm returns an assignment, is it a valid assignment?
 - If our algorithm does not return an assignment, does an assignment exist?

Correctness of greedy solution

- If our algorithm returns an assignment, is it a valid assignment?

```

HORN(H)
1 set all variables to false
2 for all implications i
3   if EMPTY(LHS(i))
4     RHS(i) ← true
5 changed ← true
6 while changed
7   changed ← false
8   for all implications i
9     if LHS(i) = true and !RHS(i) = true
10      RHS(i) ← true
11      changed = true
12 for all negative clauses c
13   if c = false
14     return false
15 return true
  
```

Correctness of greedy solution

- If our algorithm returns an assignment, is it a valid assignment?

```

HORN(H)
1 set all variables to false
2 for all implications i
3   if EMPTY(LHS(i))
4     RHS(i) ← true
5 changed ← true
6 while changed
7   changed ← false
8   for all implications i
9     if LHS(i) = true and !RHS(i) = true
10      RHS(i) ← true
11      changed = true
12 for all negative clauses c
13   if c = false
14     return false
15 return true
  
```

explicitly check all negative clauses

Correctness of greedy solution

- If our algorithm returns an assignment, is it a valid assignment?

```

HORN(H)
1 set all variables to false
2 for all implications i
3   if EMPTY(LHS(i))
4     RHS(i) ← true
5 changed ← true
6 while changed
7   changed ← false
8   for all implications i
9     if LHS(i) = true and !RHS(i) = true
10      RHS(i) ← true
11      changed = true
12 for all negative clauses c
13   if c = false
14     return false
15 return true
  
```

don't stop until all implications with all lhs elements true have rhs true

Correctness of greedy solution

- If our algorithm does not return an assignment, does an assignment exist?

```

HORN(H)
1 set all variables to false
2 for all implications i
3   if EMPTY(LHS(i))
4     RHS(i) ← true
5 changed ← true
6 while changed
7   changed ← false
8   for all implications i
9     if LHS(i) = true and !RHS(i) = true
10      RHS(i) ← true
11      changed = true
12 for all negative clauses c
13   if c = false
14     return false
15 return true

```

Our algorithm is “stingy”. It only sets those variables that have to be true. All others remain false.

Running time?

```

HORN(H)
1 set all variables to false
2 for all implications i
3   if EMPTY(LHS(i))
4     RHS(i) ← true
5 changed ← true
6 while changed
7   changed ← false
8   for all implications i
9     if LHS(i) = true and !RHS(i) = true
10      RHS(i) ← true
11      changed = true
12 for all negative clauses c
13   if c = false
14     return false
15 return true

```

?

Running time?

```

HORN(H)
1 set all variables to false
2 for all implications i
3   if EMPTY(LHS(i))
4     RHS(i) ← true
5 changed ← true
6 while changed
7   changed ← false
8   for all implications i
9     if LHS(i) = true and !RHS(i) = true
10      RHS(i) ← true
11      changed = true
12 for all negative clauses c
13   if c = false
14     return false
15 return true

```

$O(nm)$

n = number of variables
 m = number of formulas

Knapsack problems: Greedy or not?

- 0-1 Knapsack** – A thief robbing a store finds n items worth v_1, v_2, \dots, v_n dollars and weight w_1, w_2, \dots, w_n pounds, where v_i and w_i are integers. The thief can carry at most W pounds in the knapsack. Which items should the thief take if he wants to maximize value.
- Fractional knapsack problem** – Same as above, but the thief happens to be at the bulk section of the store and can carry fractional portions of the items. For example, the thief could take 20% of item i for a weight of $0.2w_i$ and a value of $0.2v_i$.