Greedy algorithms

David Kauchak
cs302
Spring 2013

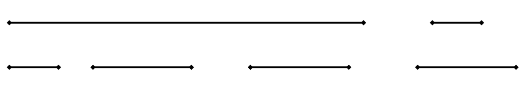
Administrative

Assignment out today (back to the normal routine)

Midterm

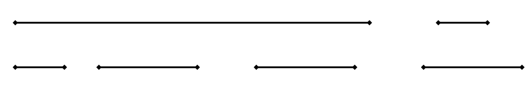
Interval scheduling

Given *n* activities A = $[a_1, a_2, .., a_n]$ 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

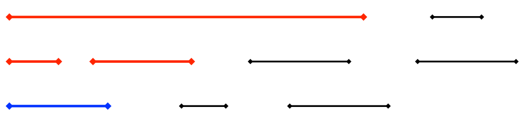
Given *n* activities A = $[a_1, a_2, .., a_n]$ where each activity has start time s_i and a finish time f_i. Schedule as many as possible of these activities such that they don't conflict.



Which activities conflict?

Interval scheduling

Given *n* activities A = [a_1,a_2, .., 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?

Simple recursive solution

Enumerate all possible solutions and find which schedules the most activities

```
IntervalSchedule-Recursive(A)
if A = {}
      return 0
else
      max = −∞
      for all a ∈ A
            A′ ← A minus a and all conflicting activites with a
            s = IntervalSchedule-Recursive(A′)
            if s > max
                  max = s
      return 1 + max
```

Simple recursive solution

Is it correct?

- max{all possible solutions}

Running time?

- O(n!)

```
IntervalSchedule-Recursive(A)
if A = {}
      return 0
else
      max = −∞
      for all a ∈ A
            A′ ← A minus a and all conflicting activites with a
            s = IntervalSchedule-Recursive(A′)
            if s > max
                  max = s
      return 1 + max
```

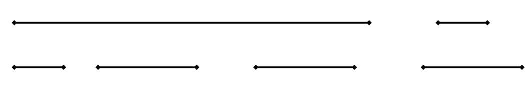
Can we do better?

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

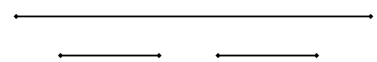
Add that activity to the answer

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. argmin$\{s_1, s_2, s_3, \ldots, s_n\}$?



Greedy options

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

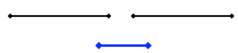
non-optimal

Greedy options

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

Greedy options

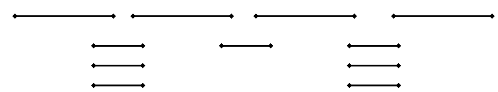
Select the shortest activity, i.e. argmin$\{f_1\text{-}s_1, f_2\text{-}s_2, f_3\text{-}s_3, \ldots, f_n\text{-}s_n\}$



non-optimal

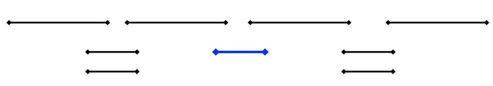
Greedy options

Select the activity with the smallest number of conflicts



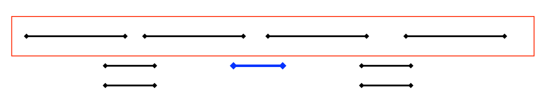
Greedy options

Select the activity with the smallest number of conflicts



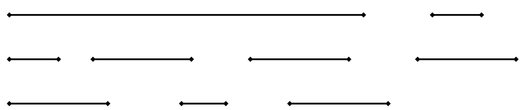
Greedy options

Select the activity with the smallest number of conflicts



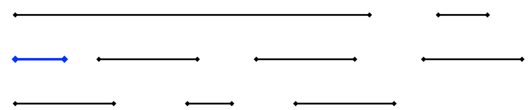
Greedy options

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



Greedy options

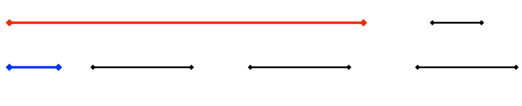
Select the activity that ends the earliest, i.e. argmin$\{f_1, f_2, f_3, \ldots, f_n\}$?



remove the conflicts

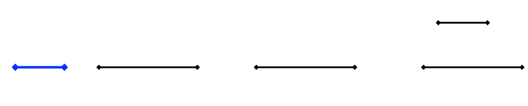
Greedy options

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



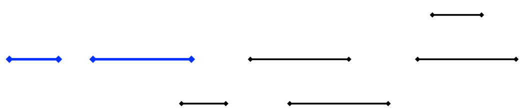
Greedy options

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



Greedy options

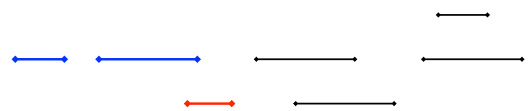
Select the activity that ends the earliest, i.e. argmin{f_1, f_2, f_3, …, f_n}?



remove the conflicts

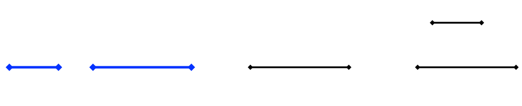
Greedy options

Select the activity that ends the earliest, i.e. argmin{f_1, f_2, f_3, …, f_n}?



Greedy options

Select the activity that ends the earliest, i.e. argmin{f_1, f_2, f_3, …, f_n}?



Greedy options

Select the activity that ends the earliest, i.e. argmin{f_1, f_2, f_3, …, f_n}?

Greedy options

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

Greedy options

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

Multiple optimal solutions

Greedy options

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

Greedy options

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

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?

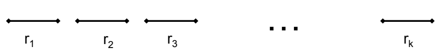
"Stays ahead" argument:

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

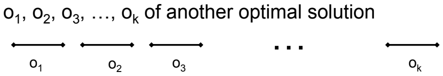
Is our greedy approach correct?

"Stays ahead" argument

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

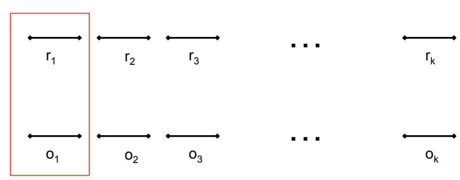


Let $o_1, o_2, o_3, \ldots, o_k$ of another optimal solution



Show our approach "stays ahead" of any other solution

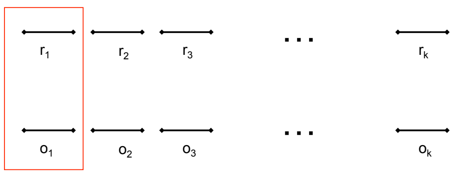
Stays ahead



Compare first activities of each solution

what do we know?

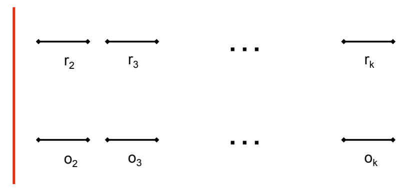
Stays ahead



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

what does this imply?

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)
sort A based on finish times f_i
for i ← 1 to n
        add a_i to R
        finish ← f_i
        while s_i < finish
                i ← i + 1
return R
```

Running time?

```
IntervalSchedule-Greedy(A)
sort A based on finish times f_i      Θ(n log n)
for i ← 1 to n
        add a_i to R
        finish ← f_i                  Θ(n)
        while s_i < finish
                i ← i + 1
return R
```

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

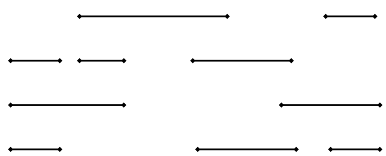
Better than:

$O(n!)$

$O(n^2)$

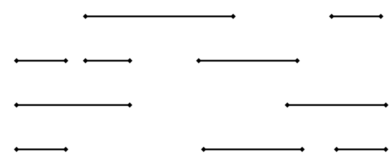
Scheduling *all* intervals

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

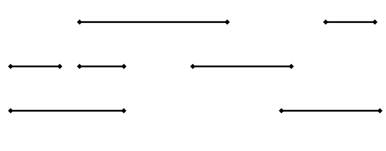


Greedy approach?

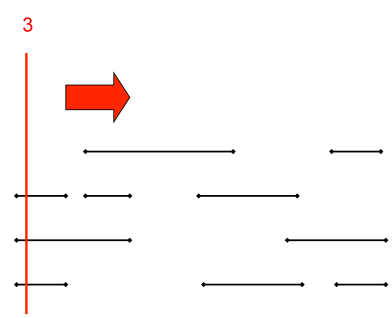
The best we could ever do is the maximum number of conflicts for any time period



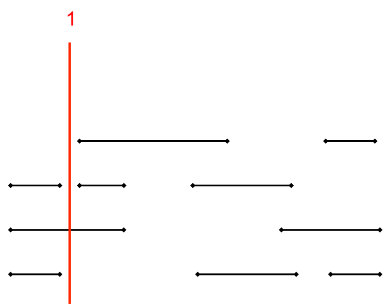
Calculating max conflicts efficiently



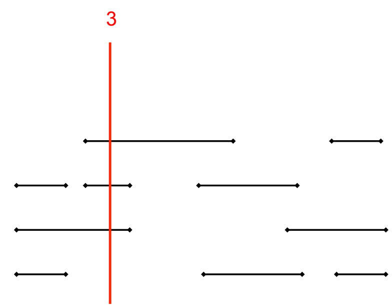
Calculating max conflicts efficiently



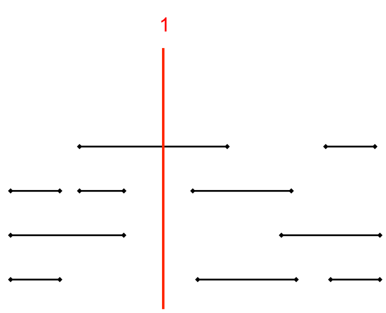
Calculating max conflicts efficiently



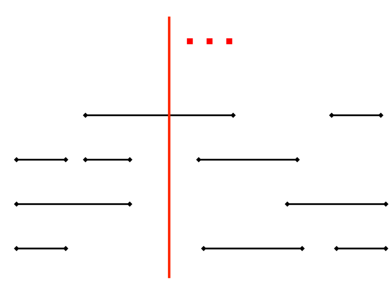
Calculating max conflicts efficiently



Calculating max conflicts efficiently



Calculating max conflicts efficiently



Calculating max conflicts

```
AllIntervalScheduleCount(A)
Sort the start and end times, call this X
current ← 0
max ← 0
for i ← 1 to length[X]
        if x_i is a start node
                current++
        else
                current--
        if current > max
                max ← current
return max
```

Correctness?

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


```
AllIntervalScheduleCount(A)
Sort the start and end times, call this X
current ← 0
max ← 0
for i ← 1 to length[X]
        if x_i is a start node
                current++
        else
                current--
        if current > max
                max ← current
return max
```


Runtime?

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


```
AllIntervalScheduleCount(A)
Sort the start and end times, call this X
current ← 0
max ← 0
for i ← 1 to length[X]
        if x_i is a start node
                current++
        else
                current--
        if current > max
                max ← current
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

$$z \wedge y \Rightarrow x$$

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

What does this implication mean in English?

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

$$\Rightarrow x$$

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

What does this implication mean in English?

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

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

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

What does this clause mean in English?

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

Horn formula

A horn formula is a set of implications and negative clauses:

$$\Rightarrow x \qquad x \wedge u \Rightarrow z$$

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

Goal

Given a horn formula, determine if the formula is satisfiable, i.e. an assignment of true/false to the variables that is consistent with all of the implications/causes

$$\Rightarrow x \qquad x \wedge u \Rightarrow z$$

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

u	x	y	z
0	1	1	0

Goal

Given a horn formula, determine if the formula is satisfiable, i.e. an assignment of true/false to the variables that is consistent with all of the implications/causes

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

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

u x y z

not satifiable

Goal

Given a horn formula, determine if the formula is satisfiable, i.e. an assignment of true/false to the variables that is consistent with all of the implications/causes

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

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

?

Goal

Given a horn formula, determine if the formula is satisfiable, i.e. an assignment of true/false to the variables that is consistent with all of the implications/causes

$x \wedge u \Rightarrow z$

what do each of these encourage in the solution?

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

Goal

Given a horn formula, determine if the formula is satisfiable, i.e. an assignment of true/false to the variables that is consistent with all of the implications/causes

$x \wedge u \Rightarrow z$ implications tell us to set some variables to true

$\bar{x} \vee \bar{y} \vee \bar{z}$ 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?

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

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

w	0
x	0
y	0
z	0

A greedy solution?

$\boxed{\Rightarrow x}$ $\quad x \wedge z \Rightarrow w$ $\quad w \wedge y \wedge z \Rightarrow x$

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

w	0
x	1
y	0
z	0

A greedy solution?

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

$\boxed{x \Rightarrow y}$ $\quad x \wedge y \Rightarrow w$ $\quad \overline{w} \vee \overline{x} \vee \overline{y}$

w	0
x	1
y	1
z	0

A greedy solution?

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

$$x \Rightarrow y \qquad \boxed{x \wedge y \Rightarrow w} \qquad \overline{w} \vee \overline{x} \vee \overline{y}$$

w	1
x	1
y	1
z	0

A greedy solution?

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

$$x \Rightarrow y \qquad x \wedge y \Rightarrow w \qquad \boxed{\overline{w} \vee \overline{x} \vee \overline{y}}$$

w	1
x	1
y	1
z	0

not satisfiable

A greedy solution

```
HORN(H)
set all variables to false
for all implications i
        if EMPTY(LHS(i))
                RHS(i) ← true
changed ← true
while changed
        changed ← false
        for all implications i
                if LHS(i) = true and !RHS(i) = true
                        RHS(i) ← true
                        changed = true
for all negative clauses c
        if c = false
                return false
return true
```

A greedy solution

```
HORN(H)
set all variables to false
for all implications i
        if EMPTY(LHS(i))
                RHS(i) ← true
changed ← true
while changed
        changed ← false
        for all implications i
                if LHS(i) = true and !RHS(i) = true
                        RHS(i) ← true
                        changed = true
for all negative clauses c
        if c = false
                return false
return true
```

set all variables of the implications of the form "⇒x" to true

A greedy solution

```
HORN(H)
set all variables to false
for all implications i
        if EMPTY(LHS(i))
                RHS(i) ← true
changed ← true
while changed
        changed ← false
        for all implications i
                if LHS(i) = true and !RHS(i) = true
                        RHS(i) ← true
                        changed = true
for all negative clauses c
        if c = false
                return false
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)
set all variables to false
for all implications i
        if EMPTY(LHS(i))
                RHS(i) ← true
changed ← true
while changed
        changed ← false
        for all implications i
                if LHS(i) = true and !RHS(i) = true
                        RHS(i) ← true
                        changed = true
for all negative clauses c
        if c = false
                return false
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)
set all variables to false
for all implications i
        if EMPTY(LHS(i))
                RHS(i) ← true
changed ← true
while changed
        changed ← false
        for all implications i
                if LHS(i) = true and !RHS(i) = true
                        RHS(i) ← true
                        changed = true
for all negative clauses c
        if c = false
                return false
return true
```

Correctness of greedy solution

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

```
Horn(H)
set all variables to false
for all implications i
    if Empty(LHS(i))
        RHS(i) ← true
changed ← true
while changed
    changed ← false
    for all implications i
        if LHS(i) = true and !RHS(i) = true
            RHS(i) ← true
            changed = true
for all negative clauses c
    if c = false
        return false
return true
```

explicitly check all negative clauses

Correctness of greedy solution

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

```
Horn(H)
set all variables to false
for all implications i
    if Empty(LHS(i))
        RHS(i) ← true
changed ← true
while changed
    changed ← false
    for all implications i
        if LHS(i) = true and !RHS(i) = true
            RHS(i) ← true
            changed = true
for all negative clauses c
    if c = false
        return false
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)
set all variables to false
for all implications i
    if Empty(LHS(i))
        RHS(i) ← true
changed ← true
while changed
    changed ← false
    for all implications i
        if LHS(i) = true and !RHS(i) = true
            RHS(i) ← true
            changed = true
for all negative clauses c
    if c = false
        return false
return true
```

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

Running time?

```
Horn(H)
set all variables to false
for all implications i
    if Empty(LHS(i))
        RHS(i) ← true
changed ← true
while changed
    changed ← false
    for all implications i
        if LHS(i) = true and !RHS(i) = true
            RHS(i) ← true
            changed = true
for all negative clauses c
    if c = false
        return false
return true
```

?

n = number of variables

m = number of formulas

Running time?

```
HORN(H)
set all variables to false
for all implications i
    if EMPTY(LHS(i))
        RHS(i) ← true
changed ← true
while changed
    changed ← false
    for all implications i
        if LHS(i) = true and !RHS(i) = true
            RHS(i) ← true
            changed = true
for all negative clauses c
    if c = false
        return false
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, .., v_n dollars and weight w_1, w_2, …, 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 $0.2v_i$.