

## Objectives

- Dynamic Programming
  - Review: Weighted Interval Scheduling
  - Wrap up: Least Segmented Squares
  - Knapsack

## Review

- What is the new algorithm design technique we're learning?

## Review Solving

### Dynamic Programming Problems

1. Determine optimal substructure of problem
  - Define the recurrence relation
2. Define algorithm to find the **value** of optimal solution
3. Optionally, change algorithm to an **iterative** rather than recursive solution
4. Define algorithm to find **optimal solution**
5. Analyze running time of algorithms

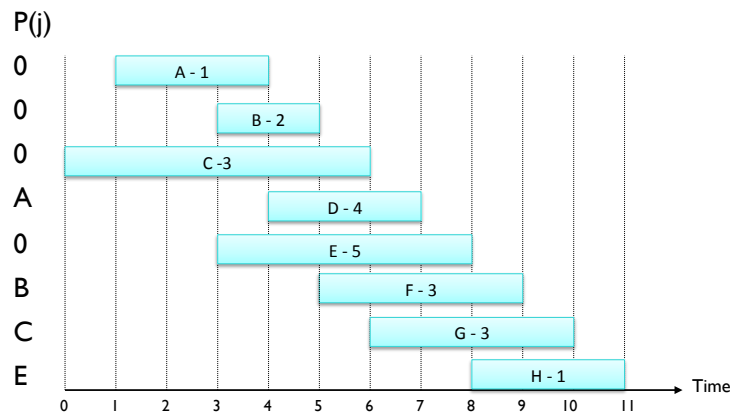
Map to weighted-interval scheduling

Mar 23, 2018

CSCI211 - Sprenkle

3

## Review: Weighted Interval Scheduling

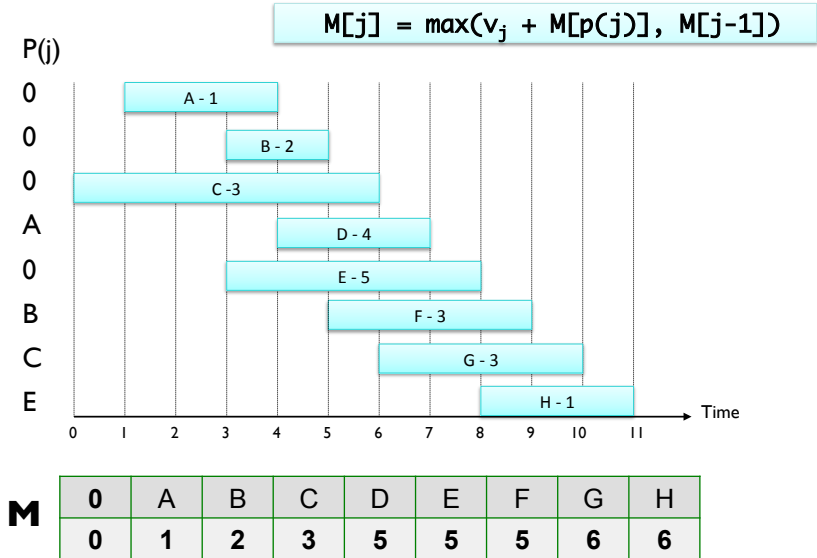


Mar 23, 2018

CSCI211 - Sprenkle

4

## Review: Weighted Interval Scheduling



Mar 23, 2018

CSCI211 - Sprenkle

5

## Review

- What is the segmented least squares problem?

Mar 23, 2018

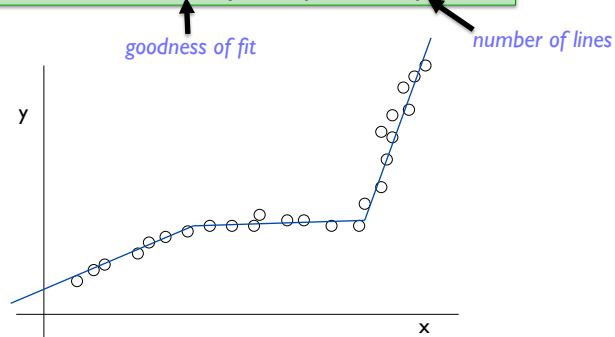
CSCI211 - Sprenkle

6

## Segmented Least Squares

- Points lie roughly on a **sequence** of line segments
- Given  $n$  points in the plane  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  with  $x_1 < x_2 < \dots < x_n$ , find a sequence of line segments that **minimizes  $f(x)$**

What's a reasonable choice for  $f(x)$  to balance *accuracy* and *parsimony*?



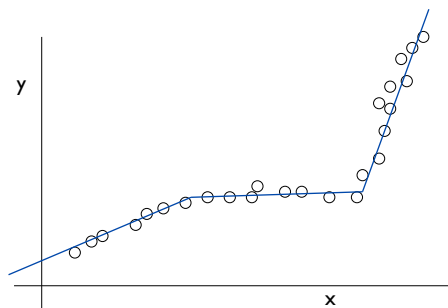
Mar 23, 2018

CSCI211 - Sprenkle

7

## Segmented Least Squares

- Points lie roughly on a **sequence** of several line segments.
- Given  $n$  points in the plane  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  with  $x_1 < x_2 < \dots < x_n$ , find a sequence of line segments that minimizes:
  - $E$ : sum of the sums of the squared errors in each segment
  - $L$ : the number of lines
- **Tradeoff function:**  $E + cL$ , for some constant  $c > 0$ .



How should we define an optimal solution?

Mar 23, 2018

CSCI211 - Sprenkle

8

## Recall:

### Properties of Problems for DP

- Polynomial number of subproblems
- Solution to original problem can be easily computed from solutions to subproblems
- Natural ordering of subproblems, easy to compute recurrence

We need to:

- Figure out how to break the problem into subproblems
- Figure out how to compute solution from subproblems
- Define the recurrence relation between the problems

Mar 23, 2018

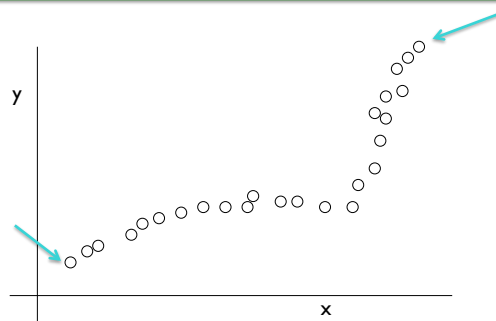
CSCI211 - Sprenkle

9

## Toward a Solution

- Consider just the first or last point

What do we know about those points?  
their segments? cost of a segment?



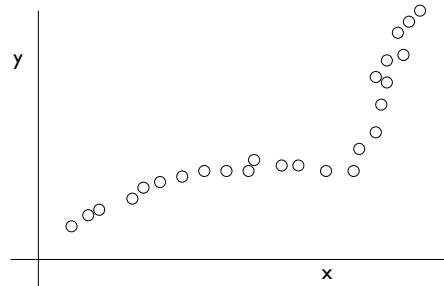
Mar 23, 2018

CSCI211 - Sprenkle

10

## Toward a Solution

- $p_n$  can only belong to one segment
  - Segment:  $p_i, \dots, p_n$
  - Cost:  $c$  (cost for segment) + error of segment
- What is the remaining problem?



Mar 23, 2018

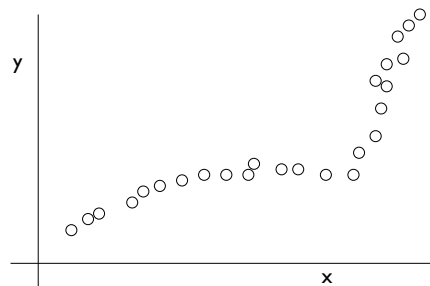
CSCI211 - Sprenkle

11

## Toward a Solution

- $p_n$  can only belong to one segment
  - Segment:  $p_i, \dots, p_n$
  - Cost:  $c$  (cost for segment) + error of segment
- What is the remaining problem?
  - Solve for  $p_1, \dots, p_{i-1}$

**Next:** Formulate as a recurrence



Mar 23, 2018

CSCI211 - Sprenkle

12

## Dynamic Programming: Multiway Choice

- **Notation.**
  - **OPT(j)** = minimum cost for points  $p_1, p_{i+1}, \dots, p_j$ .
  - **e(i, j)** = minimum sum of squares for points  $p_i, p_{i+1}, \dots, p_j$ .
- How do we compute OPT(j)?
  - Last problem: binary decision (include job or not)
  - This time: **multiway** decision
    - Which option do we choose?

Mar 23, 2018

CSCI211 - Sprenkle

13

## Dynamic Programming: Multiway Choice

- **Notation.**
  - **OPT(j)** = minimum cost for points  $p_1, p_{i+1}, \dots, p_j$ .
  - **e(i, j)** = minimum sum of squares for points  $p_i, p_{i+1}, \dots, p_j$ .
- To compute OPT(j):
  - Last segment contains points  $p_i, p_{i+1}, \dots, p_j$  for some  $i$
  - Cost =  $e(i, j) + c + \text{OPT}(i-1)$ .

$$OPT(j) = \begin{cases} 0 & \text{if } j=0 \\ \min_{1 \leq i \leq j} \{ e(i, j) + c + OPT(i-1) \} & \text{otherwise} \end{cases}$$

Mar 23, 2018

CSCI211 - Sprenkle

14

## Segmented Least Squares: Algorithm Analysis

How do we find the solution?

INPUT:  $n, p_1, \dots, p_n, c$

Segmented-Least-Squares()

$M[0] = 0$

$e[0][0] = 0$

for  $j = 1$  to  $n$

  for  $i = 1$  to  $j$

$e[i][j] =$  least square error for the  
    segment  $p_i, \dots, p_j$

$O(n^2)$  for  $j = 1$  to  $n$   
   $M[j] = \min_{1 \leq i \leq j} (e[i][j] + c + M[i-1])$

return  $M[n]$

can be improved to  $O(n^2)$  by  
pre-computing various statistics

$O(n^3)$

- Bottleneck: computing  $e(i, j)$  for  $O(n^2)$  pairs,  $O(n)$  per pair using previous formula

Mar 23, 2018

CSCI211 - Sprenkle

15

## Post-Processing: Finding the Solution

FindSegments( $j$ ):

  if  $j = 0$ :

    output nothing

  else:

    Find an  $i$  that minimizes  $e_{i,j} + c + M[i-1]$

    Output the segment  $\{p_i, \dots, p_j\}$

    FindSegments( $i-1$ )

Cost?

$O(n^2)$

Call as: FindSegments( $n$ )

Mar 23, 2018

CSCI211 - Sprenkle

16



# KNAPSACK

Mar 23, 2018

CSCI211 - Sprenkle

17

## Knapsack Problem

- Given  $n$  objects and a “knapsack”
- Item  $i$  weighs  $w_i > 0$  kilograms and has value  $v_i > 0$ 
  - Example: jobs require  $w_i$  time
- Knapsack has capacity of  $W$  kilograms
  - Example:  $W$  is time interval that resource is available

**Goal:** fill knapsack so as to maximize total value

$W = 11$

Item	Value	Weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

Mar 23, 2018

CSCI211 - Sprenkle

## Towards a Recurrence...

- What do we know about the knapsack with respect to item  $i$ ?

## Towards a Recurrence...

- What do we know about the knapsack with respect to item  $i$ ?
  - Either select item  $i$  or not
  - If don't select
    - Pick optimum solution of remaining items
  - Otherwise

What happens?  
How does problem change?  
Formulate the recurrence

## Dynamic Programming: False Start

- **Def.**  $OPT(i)$  = max profit subset of items 1, ...,  $i$ 
  - Case 1: OPT does not select item  $i$ 
    - OPT selects best of { 1, 2, ...,  $i-1$  }
  - Case 2: OPT selects item  $i$ 
    - Accepting item  $i$  does not immediately imply that we will have to reject other items
      - No known conflicts
    - Without knowing what other items were selected before  $i$ , we don't know if we have enough room for  $i$

➔ Need more sub-problems!

Mar 23, 2018

CSCI211 - Sprenkle

21

## Dynamic Programming: Adding a New Variable

- **Def.**  $OPT(i, \mathbf{w})$  = max profit subset of items 1, ...,  $i$  with weight limit  $\mathbf{w}$ 
  - Case 1: OPT does not select item  $i$ 
    - OPT selects best of { 1, 2, ...,  $i-1$  } using weight limit  $\mathbf{w}$
  - Case 2: OPT selects item  $i$ 
    - new weight limit =  $w - w_i$
    - OPT selects best of { 1, 2, ...,  $i-1$  } using new weight limit,  $\mathbf{w} - \mathbf{w}_i$

$$OPT(i, w) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1, w) & \text{if } w_i > w \\ \max\{OPT(i-1, w), v_i + OPT(i-1, w - w_i)\} & \text{otherwise} \end{cases}$$

Mar 23, 2018

22

## Looking Ahead

- Wiki due Monday
  - Chap 6: 6.1-6.3
- PS8 due Friday