

# Delayed Column Generation (aka variable generation)

Optimization - 10725  
Carlos Guestrin  
Carnegie Mellon University

February 13<sup>th</sup>, 2008

©2008 Carlos Guestrin

1

## Why constraint generation converges

- LP with many many constraints:
- Solve with subset of constraints:  
□ (also called "cutting planes")
- Relaxed problem, bound on objective:

$$\begin{aligned} \min_x c'x \\ a_i'x \geq b_i \quad \forall i \in I \end{aligned} \quad \xrightarrow{x_{opt}} \quad \min_x c'x \\ a_i'x \geq b_i \quad \forall i \in \mathcal{L} \quad \xleftarrow{x_{rel}} \quad \min_x c'x \\ a_i'x \geq b_i \quad \forall i \in \mathcal{L} \quad \text{optimal for relaxed problem}$$

- If solution  $x_{\mathcal{L}}$  is feasible wrt all constraints:  $\Rightarrow$

$$x_{\mathcal{L}} \text{ is optimal} \Leftrightarrow c'x_{\mathcal{L}} \leq c'x_{opt} \leq c'x_{\mathcal{L}} \quad \leftarrow \text{optimal} \quad \text{feasible} \quad \leftarrow \text{optimal} \quad \text{feasible}$$

- If solution  $x_{\mathcal{L}}$  is infeasible wrt all constraints:

$\exists$  a violated constraint, algorithm will add one and continue (if  $I \neq \emptyset$ )

©2008 Carlos Guestrin

2

# Constraint generation and duality

- Primal problem with many constraints:

$$\max_x \sum_j b_j x_j$$

$$s.t. \sum_j a_{ij} x_j \leq c_i, \forall i \in I$$

$$\max_x b'x$$

$$Ax \leq c$$

- Constraint generation: find most important constraints

- What's the dual equivalent?

most variables are zero  
incrementally build set of  
non-zero variables

- Dual:

$$\min_y \sum_i c_i y_i$$

$$\sum_i a_{ij} y_i = b_j \quad \forall j \in 1, \dots, n$$

$$y_i \geq 0 \quad \forall i \in I$$

at solution to dual  $y_{opt}$   
 $|I| - n$  variables must be zero  
 $n$  variables may or may not  
be zero

©2008 Carlos Guestrin

3

# Column generation (aka variable generation)

- Dual problem:

$$\min_y \sum_{i \in I} c_i y_i$$

$$s.t. \sum_{i \in I} a_{ij} y_i = b_j, \quad \forall j \in 1, \dots, m$$

$$y_i \geq 0, \quad \forall i \in I$$

- Many many variables!!

- At optimal basic feasible solution

- Most variables are zero  $|I| - n$  must be zero

- Idea:

- Set most variables to zero  $|I| - n$  must be zero

- Solve problem with other variables:

set  $n$  of  
vars that  
may be non zero

- Incrementally increase sets of non-zero variables

$$\min_{y_i \in \mathcal{N}} \sum_{i \in \mathcal{N}} c_i y_i$$

$$\sum_{i \in \mathcal{N}} a_{ij} y_i = b_j \quad \forall j$$

$$y_i \geq 0 \quad \forall i \in \mathcal{N}$$

©2008 Carlos Guestrin

4

## Solving problem with subset of variables

■ Solve problem with subset of variables

$$\begin{aligned} \min_y \quad & \sum_{i \in \Omega} c_i y_i \\ \text{s.t.} \quad & \sum_{i \in \Omega} a_{ij} y_i = b_j, \quad \forall j \in 1, \dots, m \\ & y_i \geq 0, \quad \forall i \in \Omega \end{aligned}$$

- Rest of variables set to zero

■ Questions:

- How do we decide what variables to use?
- How do we decide when we are done?

## What variables should we add?

- Same as simplex

□ you did this in your HW

- Solve problem with variables  $\Omega$

- At optimal basic feasible solution set of basic variables  $B$

- Find submatrix corresponding to basic variables  $A_B$

- Cost of these variables  $c_B$

$$c_B = \begin{pmatrix} c_{B_1} \\ \vdots \\ c_{B_n} \end{pmatrix}$$

$$\begin{aligned} \min_y \quad & \sum_{i \in \Omega} c_i y_i \\ \text{s.t.} \quad & \sum_{i \in \Omega} a_{ij} y_i = b_j, \quad \forall j \in 1, \dots, m \\ & y_i \geq 0, \quad \forall i \in \Omega \end{aligned}$$

$$A_B = \begin{pmatrix} a_{1B_1} & \dots & a_{1B_n} \\ a_{2B_1} & & \vdots \\ \vdots & & \vdots \\ a_{mB_1} & \dots & a_{mB_n} \end{pmatrix}$$

- Reduced cost for each potential new variable  $y_i$ , for  $i \in I$ :

- If all are positive? if  $\forall i \bar{c}_i \geq 0 \Rightarrow$  Solution is optimal

- Otherwise:

add some variable  $y_i$  such that  $\bar{c}_i < 0$  (best is one smallest  $\bar{c}_i$ )

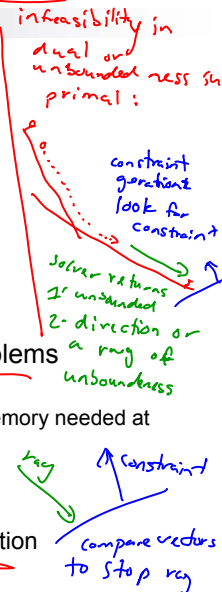
$$\bar{c}_i = c_i - c_B^T A_B^{-1} A_i$$

$A_i$  the  $i$ th column of  $A$

- Guaranteed to converge to optimal solution

# Column generation summary

- Dual of constraint generation
- Also useful for problems with infinitely many variables
- Some problems
  - Have efficient separation oracles
    - In these, constraint generation is useful
  - Have efficient variable generation oracles
    - In these, column generation is useful
- Both methods can be useful in polynomially large problems
  - E.g., when constraint matrix is too large to fit in memory
    - By incrementally solving the problem, bound amount of memory needed at each iteration
- If you have many many variables and constraints
  - Can use a combination of constraint and column generation



©2008 Carlos Guestrin