IE418: Integer Programming

Jeff Linderoth

Department of Industrial and Systems Engineering
Lehigh University

7th February 2005

Please Don't Call On Me!

10 Minutes Only!

Any questions on the homework?

- What is strong branching?
- What are pseudocosts?

- Don't forget—1/9;05, 9AM, Room 444, Introduction to Linux and Computing in COR@L!
- Thursday 2/10—12PM. COR@L Lunchtime Seminar Series.

Solving Integer Knapsack by B&B

Integer Knapsack Problem

$$(IKP) \quad \max_{x \in \mathbb{Z}^n_+} \{c^T x \mid a^T x \leq b\}$$

- To solve the linear programming relaxation of (IKP), you need only be *greedy!*
- Sort the coefficients from largest c_j/a_j to smallest c_j/a_j: Bang/Buck ratio
- Cram 'em in, in that order.
- After you branch, be sure to obey all the restrictions in your cramming.

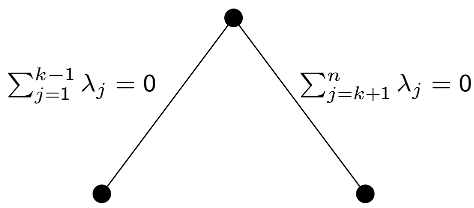
SOS2 Branching

- $\{\lambda_1, \lambda_2, \ldots \lambda_{100}\}$ is an SOS2
- Suppose:
  - $\lambda_1 = 0.2$
  - $\lambda_6 = 0.1$
  - $\lambda_8 = 0.3$
  - $\lambda_{10} = 0.1$
  - $\lambda_{17} = 0.05$
  - $\lambda_{99} = 0.25$

The $64 Question

How would you branch?

- If $\lambda_k > 0$, then feasible solutions have $\lambda_1 = \cdots = \lambda_{k-1} = 0$, or $\lambda_{k+1} = \cdots = \lambda_{100} = 0$



- Even better: Let $\lambda_k > 0, \lambda_l > 0$ with $l \geq k + 2$
- Branch on *any* variable with index $k + 1, \ldots l - 1$
- Then the infeasible point is excluded on both branches.

Choices in Branch and Bound **Node Selection**

- We've talked about one choice in branch and bound: Which variable.
- Another important choice in branch and bound is the strategy for selecting the next subproblem to be processed.
  - That said, in general, the branching variable selection method has a larger impact on solution time than the node selection method
- Node selection is often called search strategy
- In choosing a search strategy, we might consider two different goals:
  - Minimizing overall solution time.
  - Finding a good feasible solution quickly.

The Best First Approach

- One way to minimize overall solution time is to try to minimize the size of the search tree.
- We can achieve this choose the subproblem with the best bound (highest upper bound if we are maximizing).
- Can you prove this?
  - A candidate node is said to be *critical* if its bound exceeds the value of an optimal solution solution to the IP.
  - Every critical node will be processed no matter what the search order
  - Best first is guaranteed to examine only critical nodes, thereby minimizing the size of the search tree.

QUITE ENOUGH DONE

Drawbacks of Best First

1. Doesn't necessarily find feasible solutions quickly
   - Feasible solutions are "more likely" to be found deep in the tree
2. Node setup costs are high
   - The linear program being solved may change quite a bit from one node evalution to the next
3. Memory usage is high
   - It can require a lot of memory to store the candidate list, since the tree can grow "broad"

The Depth First Approach

- The depth first approach is to always choose the deepest node to process next.
  - Just dive until you prune, then back up and go the other way
- This avoids most of the problems with best first:
  - The number of candidate nodes is minimized (saving memory).
  - The node set-up costs are minimized
    - LPs change very little from one iteration to the next
  - Feasible solutions are usually found quickly
- Unfortunately, if the initial lower bound is not very good, then we may end up processing lots of non-critical nodes.
- We want to avoid this extra expense if possible.

Hybrid Strategies

- Go depth-first until you find a feasible solution, then do best-first search

A Key Insight

If you *knew* the optimal solution value, the best thing to do would be to go depth first

- Go depth-first for a while, then make a best-first move.
- What is "for a while"?
  - Estimate z_E as the optimal solution value
  - Go depth-first until $z_{LP} \leq z_E$
  - Then jump to a better node

Estimate-based Strategies

- Let's focus on a strategy for finding feasible solutions quickly.
- One approach is to try to estimate the value of the optimal solution to each subproblem and pick the best.
- For any subproblem S_i, let
  - $s^i = \sum_j \min(f_j, 1 - f_j)$ be the sum of the integer infeasibilities,
  - z_U^i be the upper bound, and
  - z_L the global lower bound.
- Also, let S_0 be the root subproblem.
- The best projection criterion is $E_i = z_U^i + \left(\frac{z_L - z_U^0}{s^0}\right) s^i$
- The best estimate criterion uses the pseudo-costs to obtain $E_i = z_U^i + \sum_j \min\left(P_j^- f_j, P_j^+(1 - f_j)\right)$

A Simple LP

- The WorldLight Company produces two types of light fixtures (products 1 and 2) that require both metal frame parts and electrical components.
- For each unit of product 1, 1 unit of frame parts and 2 units of electrical components are required.
- For each unit of product 2, 3 units of frame parts and 2 units of electrical components are required.
- The company has 200 units of frame parts and 300 units of electrical components.
- Each unit of product 1 gives a net profit of $1, and each unit of product 2, up to 60 units, gives a profit of $2.
- Any excess over 60 units of product 2 brings no profit, so such an excess has been rules out explicity.

LP Instance

$$\max x_1 + 2x_2$$

subject to

$$\begin{aligned} x_1 + 3x_2 &\leq 200 \\ 2x_1 + 2x_2 &\leq 300 \\ x_2 &\leq 60 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Communicating Instances to a Solver

1. Formulate the model
2. Gather all the data
3. Generate the constraint matrix for your instance and data. (A, b, c, etc)
4. Type the entire constraint matrix into a file using a "standard format"
5. Pass the file to a solver
6. Get the answer and interpret it in terms of the original model

Problems with this approach

- The constraint matrices can be huge!!!
  - Maybe write a "matrix generation" program to create the constraint matrix file.
- If you want to modify the model parameters or data, you have to retype the entire matrix.
- The "standard" file format, called *MPS Format* is...
  - Old.
  - So very, very ugly.

How Ugly Is It?

```
NAME
ROWS
 N   obj
 L   c1
 L   c2
 L   c3
COLUMNS
    x1         obj                 -1    c1                        1
    x1         c2                   2
    x2         obj                 -2    c1                        3
    x2         c2                   2    c3                        1
RHS
    rhs        c1                 200    c2                      300
    rhs        c3                  60
ENDATA
```


Recognize this problem?

- It's your old friend WorldLight!

maximize

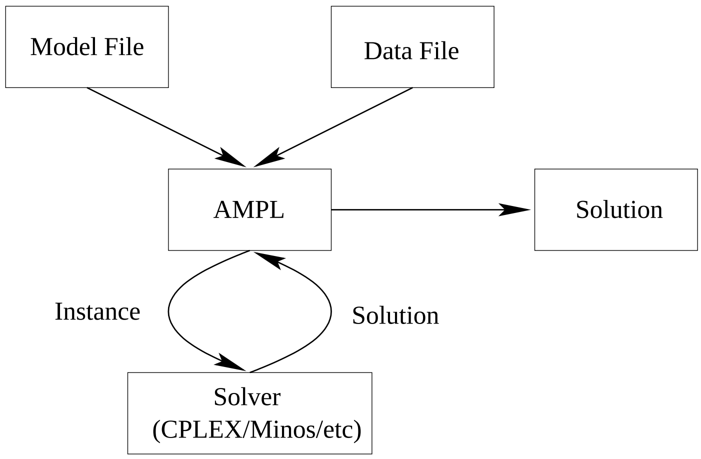
$$x_1 + 2x_2$$

subject to

$$\begin{array}{rcll} x_1 + 3x_2 & \leq & 200 & \text{Frame Part Units} \\ 2x_1 + 2x_2 & \leq & 300 & \text{Electrical Components} \\ x_2 & \leq & 60 & \text{Rule out production over 60 units} \\ x_1 & \geq & 0 & \text{The immutable laws of physics} \\ x_2 & \geq & 0 & \text{The immutable laws of physics} \end{array}$$

AMPL Concepts

- AMPL is an Algebraic Modeling Language
- In many ways, AMPL is like any other programming language.
- It just has special syntax that helps us create an optimization instance and interact with optimization solvers.



Modeling and Solving in AMPL

```
ampl: option solver cplexamp;
ampl: var x1;
ampl: var x2;
ampl: maximize profit: x1 + 2 * x2;
ampl: subject to frame_parts: x1 + 3 * x2 <= 200;
ampl: subject to electrial_components: 2 * x1 + 2 * x2 <= 300;
ampl: subject to x2_prod_limit: x2 <= 60;
ampl: subject to x2_lb: x2 >= 0;
ampl: solve;
CPLEX 7.1.0: optimal solution; objective 175
3 simplex iterations (0 in phase I)
ampl: display x1;
x1 = 125
ampl: display x2;
x2 = 25
ampl: quit;
```

Generalizing the Model

- Suppose we want to generalize the model to more than two products
  - AMPL (and all “real” modeling environments) allow the model to be separated from the data
  - This is *IMPORTANT!!!*
- Data
  - Sets: lists of products, materials, etc
  - Parameters: numerical inputs such as costs, etc
- Model
  - Variables: The values to be decided upon
  - Objective Function
  - Constraints

Fickle Management

- Management now has decided that it wants to build *five* new products.

	Product 1	Product 2	Product 3	Product 4	Product 5
Frame Parts	1	3	2	3	1
Elec. Comp.	2	2	2	1	3
Profit	1	2	1.4	1.8	1.7
Prod. Limit	∞	60	80	50	66

The Generalized WorldLight Problem

```
set PROD;
param profit {PROD};
param frame_req {PROD};
param elec_req {PROD};
param max_production {PROD};

var x{PROD} >= 0;

maximize total_profit:
sum {i in PROD} profit[i] * x[i];
```


GWP, Cont.

```
subject to frame_parts:
sum {i in PROD} frame_req[i] * x[i] <= 200;

subject to electrial_components:
sum {i in PROD} elec_req[i] * x[i] <= 300;

subject to production_limits {i in PROD}:
x[i] <= max_production[i];
```

New World Light Data File

```
set PROD := p1 p2 p3 p4 p5;

param: profit frame_req elec_req max_production :=
p1 1 2 1 Infinity
p2 3  2 2 60
p3 2 2 1.4 80
p4 3 1 1.8 50
p5 1 3 1.7 66 ;
```


Solving the Big WorldLight Problem

```
ampl: option solver cplexamp;
ampl: model wl.mod;
ampl: data wl-1.dat;
ampl: data wl-1.dat;
ampl: solve;
CPLEX 7.1.0: optimal solution; objective 360
3 simplex iterations (0 in phase I)
ampl: display x;
x [*] :=
p1   0
p2  60
p3  15
p4  50
p5   0   ;
```

Important AMPL Notes

- The # character starts a comment
- Variables are declared using the **var** keyword.
- All statements must end in a semi-colon;
- Names must be unique!
  - A variable and a constraint cannot have the same name
- AMPL is case sensitive. Keywords must be in lower case.

Getting AMPL

- AMPL is available in COR@L (`/usr/local/bin/ampl`)
- Student versions at `http://www.ampl.com`
  - Limited to 300 variables and 300 constraints.
  - You will also want to get the AMPL/CPLEX Solver
- There are "full fledged" versions of solvers you can use with AMPL on NEOS.
  - `http://www.mcs.anl.gov/neos`

Fun, Interactive Portion of Class

- Let's solve a TSP!
- How to deal with those pesky "subtour eliminations?"
- Let's solve the problem without them first...

The Separaration Problem

Given $\hat{x} \in \mathbb{R}^{|E|}$, does $\exists S \subseteq V$ such that

$$\sum_{e \in \delta(S)} x_e < 1?$$

- $\delta(S) = \{e = (i,j) \in E \mid i \in S, j \notin S\}$
- Does this problem look familiar?
  - min $s - t$ cut!
- Is the problem easier if $x \in \mathbb{B}^{|E|}$?

Our TSP

- Through 10 cities in the United States.

```
param c : Atlanta Chicago Denver Houston LosAngeles Miami NewYork SanFrancisco Seattle W
ashingtonDC :=
Atlanta         0  587  1212  701  1936  604  748  2139  2182  543
Chicago       587     0  920  940  1745  1188  713  1858  1737  597
Denver       1212   920     0   879  831  1726  1631  949  1021  1494
Houston       701   940   879     0   1372  968  1420  1645  1891  1220
LosAngeles   1936  1745   831  1374    0  2339  2451  347  959  2300
Miami         604  1188  1726   968  2339     0  1092  2594  2734  923
NewYork       748   713  1631  1420  2451  1092     0   2571  2408  205
SanFrancisco 2139  1858   949  1645   347  2594  2571     0  678  2442
Seattle      2182  1737  1021  1891   959  2734  2408   678     0   2329
WashingtonDC  543   597  1494  1220  2300   923   205  2442  2329     0
;
```

Mosel

- A modeling language (and environment) from Dash Optimization that uses the Xpress-MP optimizer
- On shark
  - In `/usr/local/shark`
  - `file:///usr/local/xpress/docs/mosel/mosel_ug/dhtml/moselug.html`
  - Software: `/home/jeff/IP-Class`

Next Time

- Ugh – Homework #1 Due!
- Ugh – Pass out homework #2?
- Lots more stuff on IP Software
- Don't forget—