Integer Programming II

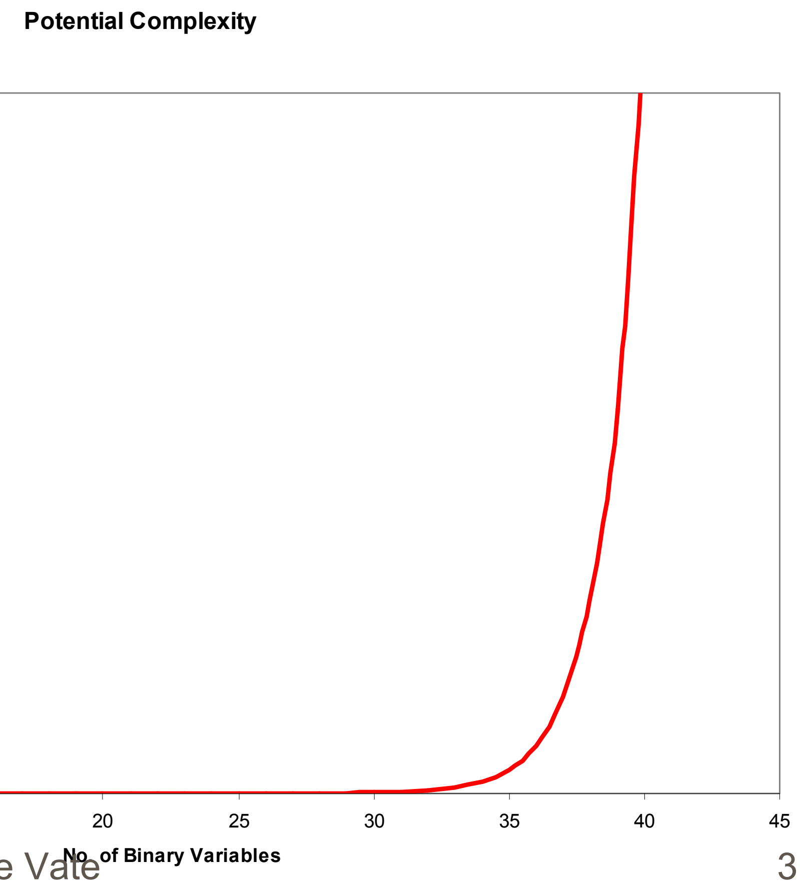
Modeling to Reduce Complexity
Capturing Economies of Scale

Better Models

- Better Formulation can distinguish solvable from not.
- Often counterintuitive what's better
- Has led to vastly improved solvers that actually improve your formulation as they solve the problem.

In Theory…

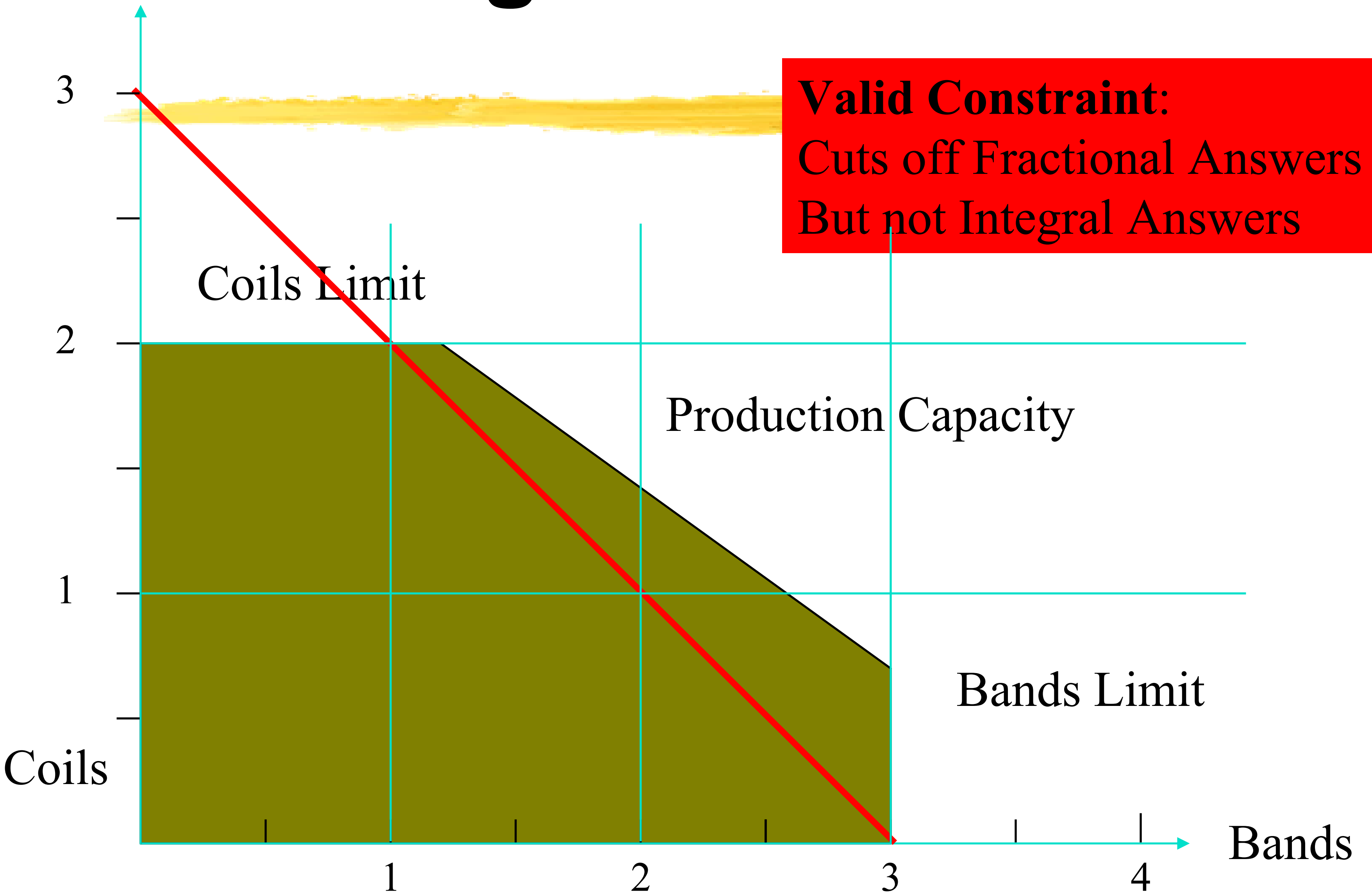
- Each new binary variable doubles the difficulty of the problem



Eliminate Excess Variables

- Assign each customer to a DC
- s.t. AssignCustomers{cust in CUSTOMERS}:
  - sum{dc in DCS} Assign[cust, dc] <= 1;
- What improvement?

Add Stronger Constraints



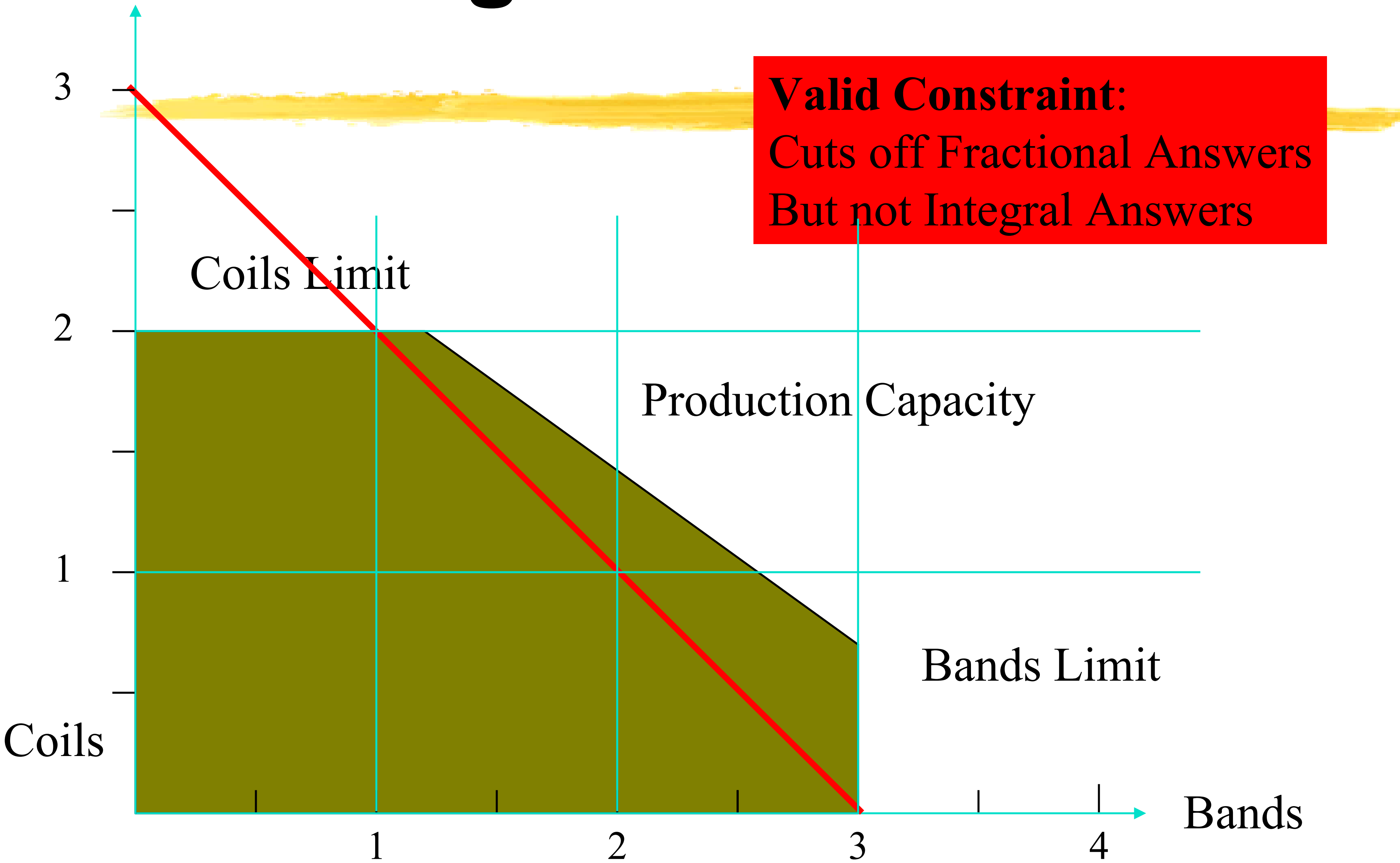
Adding Stronger Constraints

- Formulating Current Constraints Better
  - More constraints are generally better
  - Use parameters carefully
- Creating new constraints that help
  - Some examples

More is Better

- X, Y, Z binary
- Which is better?
- Formulation #1
  - $X + Y \le 2Z$
- Formulation #2
  - $X \le Z$
  - $Y \le Z$

Add Stronger Constraints



Lockbox Example

Lockbox Model

Days to Mail from Each Area to Each City

City	Sea.	Chi.	NY	LA	Daily Payments
NW	2	5	5	4	$ 325,000
N	4	2	4	6	$ 475,000
NE	5	5	2	8	$ 300,000
SW	4	6	8	2	$ 275,000
S	6	6	6	4	$ 385,000
SE	8	8	5	5	$ 350,000
Oper.Cost	$ 55,000	$ 50,000	$ 60,000	$ 53,000	

Int. Rate 6.0%

City	Sea.	Chi.	NY	LA	Total	Total Float
NW	0	0	0	0	0	$ -
N	0	0	0	0	0	$ -
NE	0	0	0	0	0	$ -
SW	0	0	0	0	0	$ -
S	0	0	0	0	0	$ -
SE	0	0	0	0	0	$ -
Total	0	0	0	0	Total Float	$ -
Open?	0	0	0	0	Total Cost to Operate	
Cost	$ -	$ -	$ -	$ -	$ -	
Eff. Cap.	0	0	0	0		
	Total Cost	-				

Challenge

- Improve the formulation

Lockbox Model

Days to Mail from Each Area to Each City

City	Sea.	Chi.	NY	LA	Daily Payments
NW	2	5	5	4	$ 325,000
N	4	2	4	6	$ 475,000
NE	5	5	2	8	$ 300,000
SW	4	6	8	2	$ 275,000
S	6	6	6	4	$ 385,000
SE	8	8	5	5	$ 350,000
Oper.Cost	$ 55,000	$ 50,000	$ 60,000	$ 53,000	

Int. Rate	6.0%					
City	Sea.	Chi.	NY	LA	Total	Total Float
NW	0	0	0	0	0	$ -
N	0	0	0	0	0	$ -
NE	0	0	0	0	0	$ -
SW	0	0	0	0	0	$ -
S	0	0	0	0	0	$ -
SE	0	0	0	0	0	$ -
Total	0	0	0	0	Total Float	$ -
Open?	0	0	0	0	Total Cost to Operate	
Cost	$ -	$ -	$ -	$ -	$ -	
Eff. Cap.	0	0	0	0		
	Total Cost	-				

Conclusion

- Formulation #1
  - Assign[NW, b] +Assign[N, b] + Assign[NE, b] +
  - Assign[SW, b] +Assign[S, b] + Assign[SE, b]
  - $\leq$ 6*Open[b]
- Formulation #2
  - Assign[NW,b] $\leq$ Open[b]
  - Assign[N, b] $\leq$ Open[b]
  - …
- Don't aggregate or sum constraints

One Step Further

- Impose Constraints at Lowest Level
- Some Compromise between
  - Number of Constraints: How hard to solve LPs
  - Number of LPs: How many LPs we must solve.
- Generally, better to solve fewer LPs.

Steco Revisited

Steco's Warehouse Location Model

Unit Costs	Lease	Unit Cost/Truck to Sales District			
Warehouse	($)	1	2	3	4
A	$ 7,750	$170	$ 40	$ 70	$160
B	$ 4,000	$150	$195	$100	$ 10
C	$ 5,500	$100	$240	$140	$ 60

Monthly Trucks From/To

Decisions	Yes/No	1	2	3	4	Total	Eff. Cap.	Cap.
Lease A	0	0	0	0	0	0	0	200
Lease B	0	0	0	0	0	0	0	250
Lease C	0	0	0	0	0	0	0	300
Total TrucksTo		0	0	0	0			
Demand (Trucks/Mo)		100	90	110	60			

	Lease Cost	To 1	To 2	To 3	To 4	Truck $	Total Cost
A	$ -	$ -	$ -	$ -	$ -	$ -	$ -
B	$ -	$ -	$ -	$ -	$ (0)	$ (0)	$ (0)
C	$ -	$ -	$ -	$ 0	$ -	$ 0	$ 0
Totals	$ -	$ -	$ -	$ 0	$ (0)	$ 0	$ 0

Challenge

- Improve the formulation

Steco's Warehouse Location Model

Unit Costs Warehouse	Lease ($)	Unit Cost/Truck to Sales District 1	2	3	4
A	$ 7,750	$170	$ 40	$ 70	$160
B	$ 4,000	$150	$195	$100	$ 10
C	$ 5,500	$100	$240	$140	$ 60

Monthly Trucks From/To

Decisions	Yes/No	1	2	3	4	Total	Eff. Cap.	Cap.
Lease A	0	0	0	0	0	0	0	200
Lease B	0	0	0	0	0	0	0	250
Lease C	0	0	0	0	0	0	0	300
Total TrucksTo		0	0	0	0			
Demand (Trucks/Mo)		100	90	110	60			

	Lease Cost	To 1	To 2	To 3	To 4	Truck $	Total Cost
A	$ -	$ -	$ -	$ -	$ -	$ -	$ -
B	$ -	$ -	$ -	$ -	$ (0)	$ (0)	$ (0)
C	$ -	$ -	$ -	$ 0	$ -	$ 0	$ 0
Totals	$ -	$ -	$ -	$ 0	$ (0)	$ 0	$ 0

More Detailed Constraints

```
s.t. ShutWarehouse{w in WAREHOUSES}:
   sum{d in DISTRICTS} Ship[w,d] <= Capacity[w]*Open[w];

s.t. ShutLanes{w in WAREHOUSES, d in DISTRICTS}:
    Ship[w,d] <= Demand[d]*Open[w];
```

- Trade off between work to solve each LP and number of LPs we have to solve
- This makes each one harder, but we solve fewer.

Tighten Bounds

- Function of Continuous Variables <= Limit*Binary Variable
- Make the Limit as small as possible
- But not too small
- Don't eliminate feasible solutions
- We will see an Example with Ford Finished Vehicle Dist.

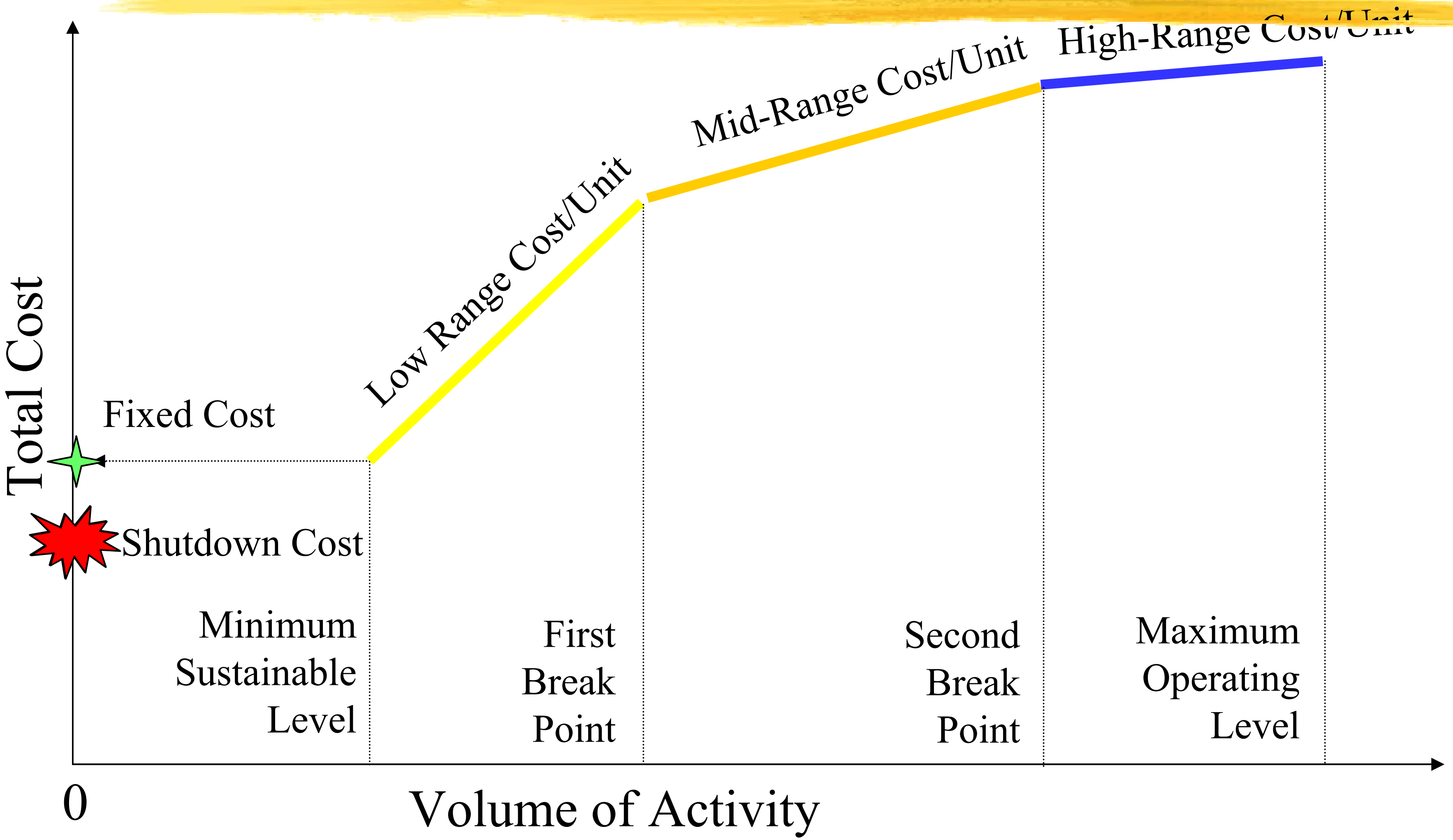
New Constraints

- Recall the Single Sourcing Problem

Constraints

- s.t. ObserveCapacity{dc in DCS}:
  - sum{cust in CUSTOMERS} Demand[cust]*Assign[dc,cust] <= Capacity[dc];
- Example: $x_1, x_2, x_3, x_4, x_5, x_6$ binary
- $5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$
- **What constraints can we add?**
  - $x_1 + x_2 + x_3 \leq 2$
  - $x_1 + x_2 + x_6 \leq 2$
  - …

Non-Linear Costs



Modeling Economies of Scale

- Linear Programming
  - Greedy
  - Takes the High-Range Unit Cost first!
- Integer Programming
  - Add constraints to ensure first things first
  - Several Strategies

Good News!

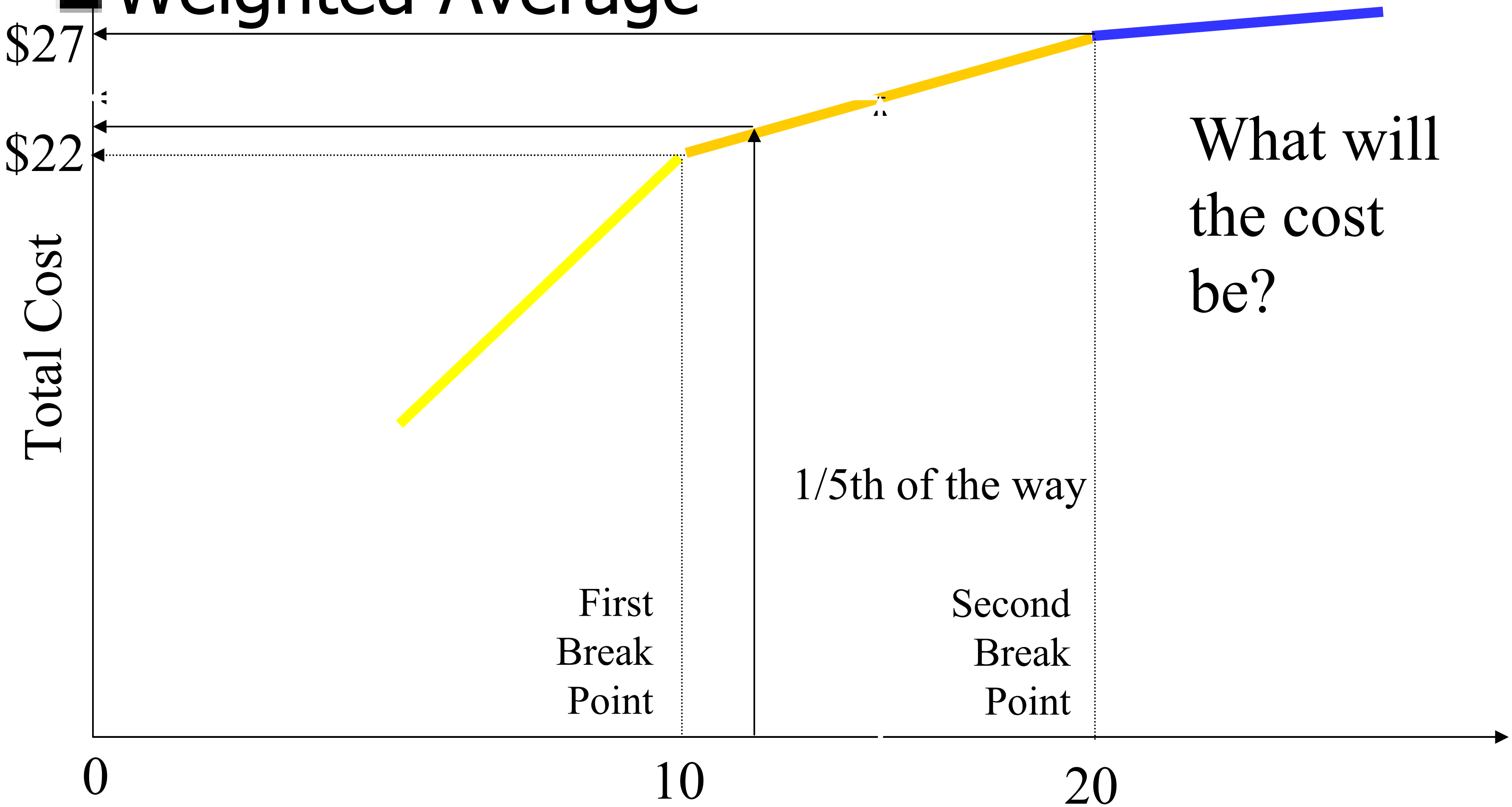
- AMPL offers syntax to “automate” this
- Read Chapter 14 of Fourer for details
- <<BreakPoint[1], BreakPoint[2]; Slope[1], Slope[2], Slope[3]>> Variable;
  - Slope[1] before BreakPoint[1]
  - Slope[2] from BreakPoint[1] to BreakPoint[2]
  - Slope[3] after BreakPoint[2]
  - Has 0 cost at activity 0

Summary

- To control complexity and get solutions
  - Eliminate unnecessary binary variables
  - Don't aggregate constraints
  - Add strong valid constraints
  - Tighten bounds
- Integer Programming Models can approximate non-linear objectives

Convex Combination

- Weighted Average



Conclusion

- If the Volume of Activity is a fraction λ of the way from one breakpoint to the next, the cost will be that same fraction of the way from the cost at the first breakpoint to the cost at the next
- If Volume = $10\lambda + 20(1-\lambda)$
- Then Cost = $22\lambda + 27(1-\lambda)$

Idea

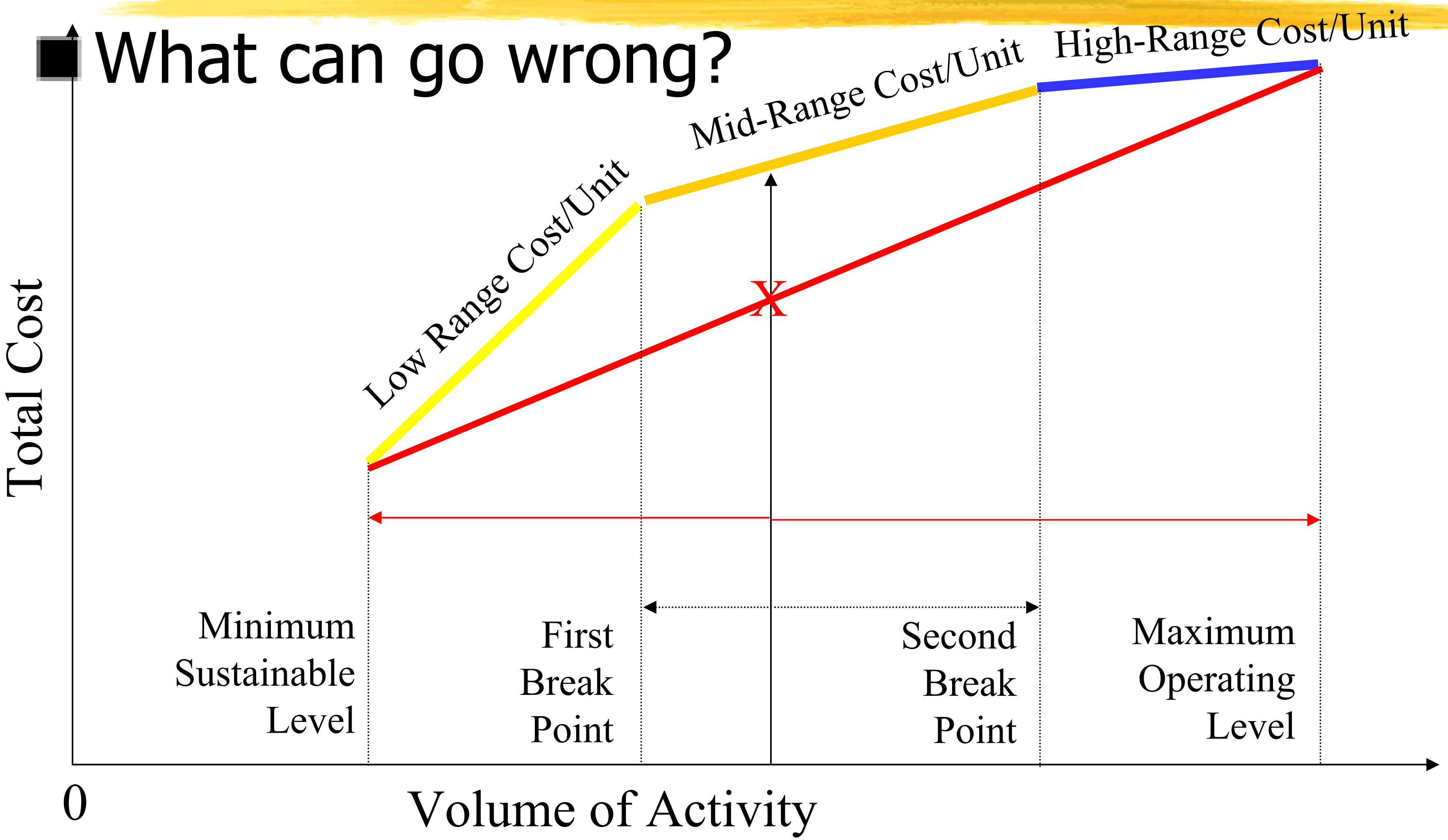
- Express Volume of Activity as a Weighted Average of Breakpoints
- Express Cost as the same Weighted Average of Costs at the Breaks
- Activity = Min Level λ_0 + Break 1 λ_1 +
  Break 2 λ_2 + Max Level λ_3
- Cost = Cost at Min Level λ_0 + Cost at Break 1 λ_1 +
  Cost at Break 2 λ_2 + Cost at Max Level λ_3
- $1 = \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3$

In AMPL Speak

- param NBreaks;
- param BreakPoint{0..NBreaks};
- param CostAtBreak{0..NBreaks};
- var Lambda{0..NBreaks} >= 0;
- var Activity;
- var Cost;
- s.t. DefineCost:
- Cost = sum{b in 0..NBreaks} CostAtBreak[b]*Lambda[b];
- s.t. DefineActivity:
- Activity = sum{b in 0..NBreaks} BreakPoint[b]*Lambda[b];
- s.t. ConvexCombination:
- 1 = sum{b in 0..NBreaks}Lambda[b];

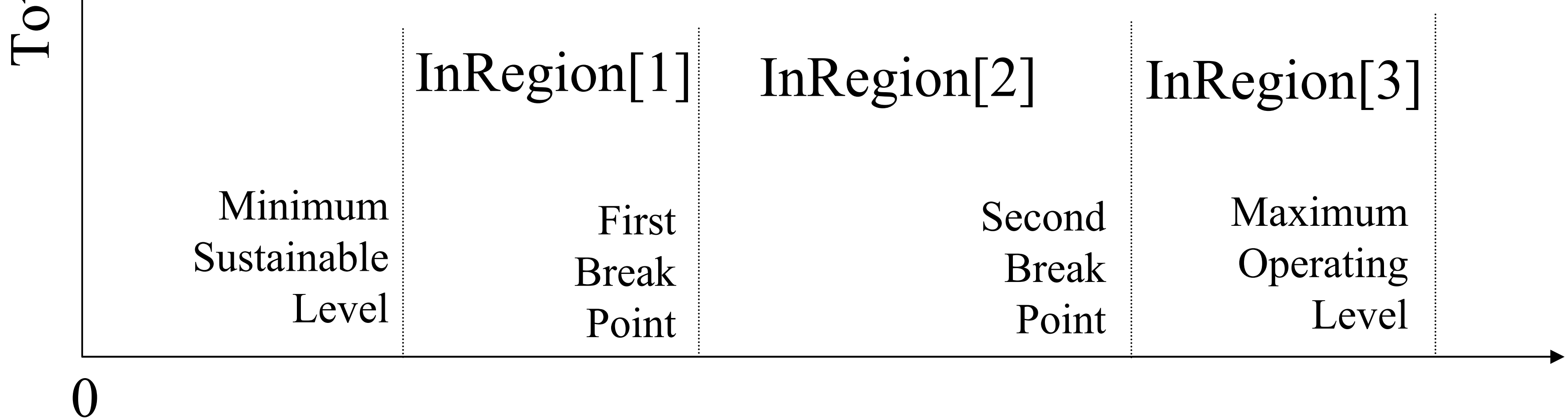
Does that Do It?

- What can go wrong?



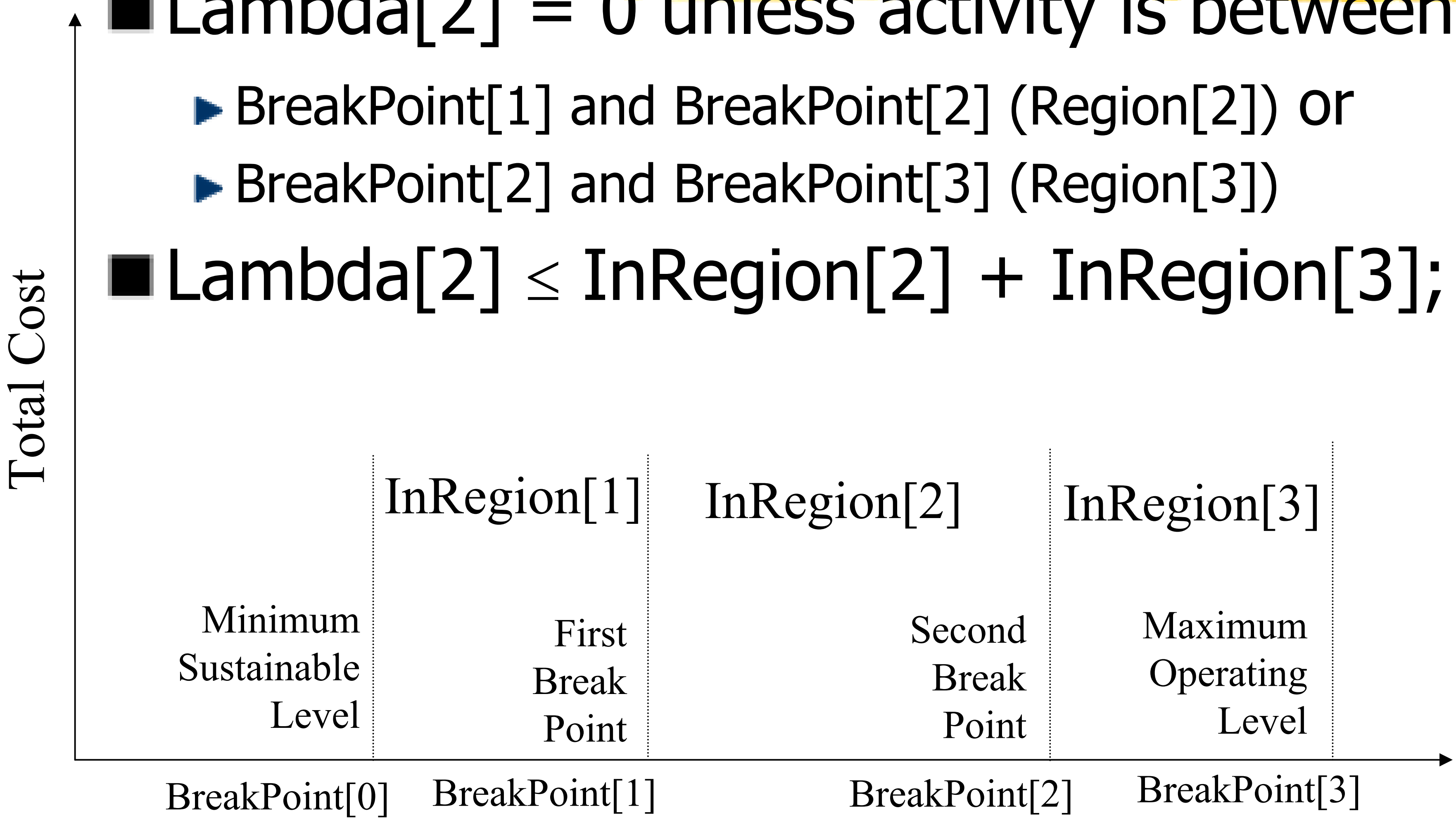
Role of Integer Variables

- Ensure we express Activity as a combination of two consecutive breakpoints
- var InRegion{1..NBreaks} binary;



Constraints

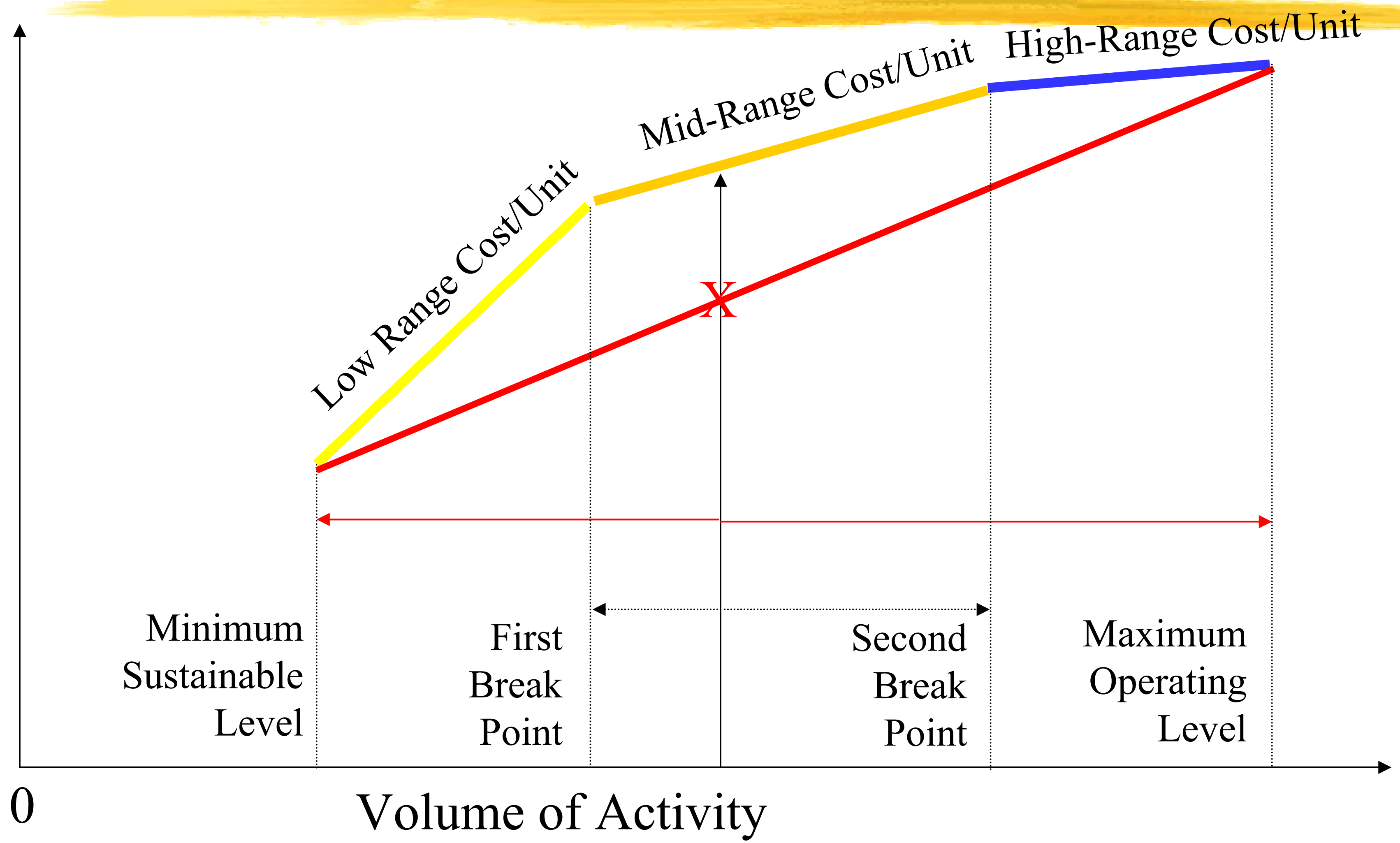
- Lambda[2] = 0 unless activity is between
  - BreakPoint[1] and BreakPoint[2] (Region[2]) or
  - BreakPoint[2] and BreakPoint[3] (Region[3])
- Lambda[2] ≤ InRegion[2] + InRegion[3];



And Activity in One Region

- $InRegion[1] + InRegion[2] + InRegion[3] \leq 1$
- Why ≤ 1?
- If it is in Region[2]:
  - $Lambda[1] \leq InRegion[1] + InRegion[2] = 1$
  - $Lambda[2] \leq InRegion[2] + InRegion[3] = 1$
  - Other Lambda's are 0

We can’t go wrong



AMPL Speak

```
param NBreaks;
param BreakPoint{0..NBreaks};
param CostAtBreak{0..NBreaks};
var Lambda{0..NBreaks} >= 0;
var Activity;
var Cost;
s.t. DefineCost:
Cost = sum{b in 0..NBreaks} CostAtBreak[b]*Lambda[b];
s.t. DefineActivity:
Activity = sum{b in 0..NBreaks} BreakPoint[b]*Lambda[b];
s.t. ConvexCombination:
1 = sum{b in 0..NBreaks}Lambda[b];
```

What we Added

- var InRegion{1..NBreaks} binary;
- s.t. InOneRegion:
-   sum{b in 1..NBreaks} InRegion[b] <= 1;
- s.t. EnforceConsecutive{b in 0..NBreaks-1}:
-   Lambda[b] <= InRegion[b] + InRegion[b+1];
- s.t. LastLambda:
  - Lambda[NBreaks] <= InRegion[NBreaks];