Outline

- Definition of integer programming
- Formulating some classical problems with integer programming
- Linear programming
- Solution methods for integer programming
- (Introduction to modeling language and solver)
- Impact of modeling on problem solving
Optimization and Programming

- Mathematical programming: to find the best solution from a set of alternatives
- A generic formulation:
  \[
  \begin{align*}
  \text{max/min} & \quad f(x) \quad \text{Objective function} \\
  \text{s. t.} & \quad x \in X \quad \text{Solution set} \\
  \text{Variables} & \quad \text{The variables (their values) represent problem solution} \\
  \text{The solution set is specified by equations and inequalities} & \quad X = \{g_i(x) \sim b_i, i = 1, \ldots, m\}, \text{where } \sim \text{ can be any of } \leq, =, \text{ and } \geq
  \end{align*}
  \]
Types of Optimization Models

- Linear programming
- Integer (linear) programming
- Nonlinear programming
- Integer nonlinear programming
Definition of Integer Programming

**Generic Model of Mixed Integer (Linear) Programming (MIP)**

\[ \frac{\text{max}}{\text{min}} \sum_{j=1}^{n} c_j x_j \]

s. t. \[ \sum_{j=1}^{n} a_{ij} x_j \sim b_i, \quad i = 1, \ldots, m \]

\[ l_j \leq x_j \leq u_j, \quad j = 1, \ldots, k \]

\[ l_j \leq x_j \leq u_j, \text{integer, } j = k + 1, \ldots, n \]

\~\ can be any of \( \leq, =, \) and \( \geq \)

Special case: binary variables
Binary Knapsack

- A set of items, each with a value and weight
- A knapsack with a weight capacity limit
- Select items to maximize the total value of the knapsack, without exceeding the weight limit

\[ \begin{align*}
\text{max} \quad & \sum_{i=1}^{n} c_i x_i \\
\text{s. t.} \quad & \sum_{i=1}^{n} a_i x_i \leq b \\
& x \in \{0, 1\}, \ i = 1, \ldots, n
\end{align*} \]
Coloring

- Assign a color to each vertex (node); two adjacent vertexes must use different colors
- Minimize the total number of colors used
Modeling Some Classical Problems with Integer Programming

Coloring (cont’d)

How to define the variables and constraints?

Define color set $C$

\[
\begin{align*}
\text{min} & \quad y_{\text{red}} + y_{\text{blue}} + \ldots \\
\text{s. t.} & \quad x_{1,\text{red}} + x_{1,\text{blue}} + \ldots = 1 \\
& \quad x_{2,\text{red}} + x_{2,\text{blue}} + \ldots = 1 \\
& \quad \vdots \\
& \quad x_{1,\text{red}} \leq y_{\text{red}} \\
& \quad x_{1,\text{blue}} \leq y_{\text{blue}} \\
& \quad \vdots \\
& \quad x_{1,\text{red}} + x_{2,\text{red}} \leq 1 \\
& \quad x_{1,\text{blue}} + x_{2,\text{blue}} \leq 1 \\
& \quad \vdots \\
& \quad x \in \{0, 1\}, y \in \{0, 1\}
\end{align*}
\]

- $x_{ic} = 1$ if node $i$ has color $c$
- $y_c = 1$ if color $c$ is used (by any node)
Set Covering

- Candidate service centers and demand points
- Each service center has a deployment cost, and can serve a subset of the demand points
- Select a subset of service centers at minimum total cost, to cover all demand points
Set Covering (cont’d)

• $J$: Set of demand points
• $I$: Candidate set of service centers
• $I_j$: Subset of service centers covering point $j \in J$
• $c_i$: Deployment cost, $i \in I$

• $x_i = 1$ if $i \in I$ is deployed

$$\min \sum_{i \in I} c_i x_i$$

s. t. \( \sum_{i \in I_j} x_i \geq 1, \forall j \in J \)

\( x \in \{0, 1\} \)
Uncapacitated Facility Location

- A set of candidate facility (e.g., warehouse) locations
- A set of customers
- Opening a facility has a fixed charge
- Transportation cost between facility locations and customers
Uncapacitated Facility Location (cont’d)

- $I$: Candidate set of facility locations
- $J$: Set of customers
- $f_i$: Fixed charge, $i \in I$
- $c_{ij}$: Transportation cost, $i \in I, j \in J$
- $x_{ij} = 1$ if facility $i \in I$ serves customer $j$
- $y_i = 1$ if facility $i \in I$ is deployed

$$\min \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

s.t.

$$\sum_{i \in I} x_{ij} = 1, \forall j \in J$$

$$x_{ij} \leq y_i, \forall i \in I, \forall j \in J$$

$x, y \in \{0, 1\}$
Traveling Salesman Problem

- Find a tour visiting each node in a graph exactly once with minimum length
Traveling Salesman Problem (cont’d)

- How to formulate this problem by integer programming?

  \[ \begin{align*}
  & \min \sum_{i \in N} \sum_{j \in N: j \neq i} c_{ij} x_{ij} \\
  \text{s. t.} & \sum_{j \in N: j \neq i} x_{ij} = 1, \forall i \in N \\
  & x \in \{0, 1\}
  \end{align*} \]

  Have we overlooked anything?

  \[ \sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq 1, \forall S \subset N \]

  Potential drawback of the formulation?
Linear Programming

A Small Example

\[
\text{max } 8x_1 + 5x_2 \\
\text{s. t. } x_1 + x_2 \leq 6 \\
9x_1 + 5x_2 \leq 45 \\
x_1 \geq 0, x_2 \geq 0, \text{ integer}
\]

\[
\text{max } 8x_1 + 5x_2 \\
\text{s. t. } x_1 + x_2 \leq 6 \\
9x_1 + 5x_2 \leq 45 \\
x_1 \geq 0, x_2 \geq 0
\]
Linear Programming Relaxation

- Relaxation: “removal” of some constraints/restrictions
- In general, the linear programming relaxation is an approximation of the integer model; the solution of the former may be fractional
- Which one is easier to solve?
Solving a Linear Programming Model

- Fundamental property: Optimum is located at one of the extreme/corner points (why?)

- This is used by the Simplex Method for solving linear programs (visiting a sequence of objective-improving extreme points)

- There are other efficient, interior-point methods
Linear Programming

The Convex Hull

- Convex hull: The minimum convex set containing the solution space

- Integer programming = linear programming on the convex hull of the integer points
- Convex hull exists, but its description is hard to derive in general
Computing the Global Optimum

- General-purpose method: Linear programming relaxation +
  - Iterative improvement in approximating the convex hull, a.k.a. cutting planes
  - Divide-and-conquer, a.k.a. branch-and-bound

- Optimality gap: The (relative) difference between the objective value of the best known integer solution and that of the best ("optimistic") LP bound so far
Cutting Planes: An Example

Knapsack problem instance:

\[
\begin{align*}
\text{max} & \quad 6x_1 + 4x_2 + 6x_3 + 7x_4 + 5x_5 + 9x_6 + 8x_7 \\
\text{s. t.} & \quad 5x_1 + 6x_2 + 8x_3 + 6x_4 + 4x_5 + 6x_6 + 5x_7 \leq 21 \\
& \quad x \in \{0, 1\}
\end{align*}
\]

LP optimum: \(x_1 = x_5 = x_6 = x_7 = 1, x_4 = 0.167, x_2 = x_3 = 0\)

Can we pack items 1, 4, 6, and 7 all in the knapsack? (5+6+6+5=22)

\[
\Rightarrow x_1 + x_4 + x_6 + x_7 \leq 3
\]

The above inequality (referred as a “cover cut”) is valid for integer solutions, but violated by the LP relaxation optimum
Cutting Planes: An Example (cont’d)

Adding the cut to the linear programming relaxation:

\[
\begin{align*}
\text{max} & \quad 6x_1 + 4x_2 + 6x_3 + 7x_4 + 5x_5 + 9x_6 + 8x_7 \\
\text{s. t.} & \quad 5x_1 + 6x_2 + 8x_3 + 6x_4 + 4x_5 + 6x_6 + 5x_7 \leq 21 \\
& \quad x_1 + x_4 + x_6 + x_7 \leq 3 \\
& \quad 0 \leq x \leq 1
\end{align*}
\]

LP optimum: \( x_4 = x_5 = x_6 = x_7 = 1, x_1 = x_2 = x_3 = 0 \)

Challenge: To time-efficiently find valid and useful cuts
Solution Methods for Integer Programming

Branch-and-Bound: An Example

Same knapsack problem instance:

\[
\begin{align*}
\text{max} \quad & 6x_1 + 4x_2 + 6x_3 + 7x_4 + 5x_5 + 9x_6 + 8x_7 \\
\text{s. t.} \quad & 5x_1 + 6x_2 + 8x_3 + 6x_4 + 4x_5 + 6x_6 + 5x_7 \leq 21 \\
& x \in \{0, 1\}
\end{align*}
\]

LP optimum: \( x_1 = x_5 = x_6 = x_7 = 1, x_4 = 0.167, x_2 = x_3 = 0 \)

Rounding down \( x_4 \) to zero gives an integer solution of value 28

\( \Rightarrow \) Integer optimum is at least 28
Solution Methods for Integer Programming

Branch-and-Bound: An Example (cont’d)

Branching generates a search tree

We can stop branching here if integer solution of value 29 is known (why?)

Can we stop branching here because of integer solution of value 28?
Optimization Solver

- Solver: software implementing methods for solving optimization models (here: integer programming models)

- Interface + optimization engine

```
CPLEX> read knapsack.lp
Problem 'knapsack.lp' read.
Read time = 0.00 sec.
CPLEX> optimize
MIP emphasis: balance optimality and feasibility.
Root relaxation solution time = 0.00 sec.

Knapsack.lp

maximize
12x1 + 10x2 + 25x3 + 11x4 + 30x5 + 24x6
subject to
2x1 + x2 + 5x3 + 2.5x4 + 4x5 + 3x6 <= 14
binaries
x1 x2 x3 x4 x5 x6
end

nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>Left</th>
<th>Objective</th>
<th>IIInf</th>
<th>Best Integer</th>
<th>Cuts/Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>96.0000</td>
<td>1</td>
<td>96.0000</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>0</td>
<td>87.0000</td>
<td>96.0000</td>
<td>1</td>
<td>10.34%</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>0</td>
<td>91.0000</td>
<td>96.0000</td>
<td>1</td>
<td>5.49%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>cutoff</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Cover cuts applied: 1
Implied bound cuts applied: 1

MIP - Integer optimal solution: Objective = 9.10000000000e+01
Solution time = 0.00 sec. Iterations = 2 Nodes = 0

CPLEX> display solution var -
Variable Name    Solution Value
x1               1.000000
x3               1.000000
x5               1.000000
x6               1.000000
All other variables in the range 1-6 are 0.
CPLEX> 
```
Another Sample of Solver Log for Integer Programming

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Objective</th>
<th>Best Integer</th>
<th>Best Bound</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>227006.7258</td>
<td>227006.7258</td>
<td>2417</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0+</td>
<td>232258.1349</td>
<td>227006.7258</td>
<td></td>
<td>2.26%</td>
<td></td>
</tr>
<tr>
<td>0+</td>
<td>228124.0562</td>
<td>227006.7258</td>
<td></td>
<td>0.49%</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>227020.2075</td>
<td>228124.0562</td>
<td>Cuts: 34</td>
<td>2475</td>
<td>0.48%</td>
</tr>
<tr>
<td>0</td>
<td>227020.4963</td>
<td>228124.0562</td>
<td>Cuts: 31</td>
<td>2500</td>
<td>0.48%</td>
</tr>
<tr>
<td>0</td>
<td>227020.5204</td>
<td>228124.0562</td>
<td>Cuts: 11</td>
<td>2505</td>
<td>0.48%</td>
</tr>
<tr>
<td>0</td>
<td>227020.5204</td>
<td>228124.0562</td>
<td>2505</td>
<td>0.48%</td>
<td></td>
</tr>
</tbody>
</table>

Elapsed time = 2.62 sec. (969.11 ticks, tree = 0.01 MB)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Objective</th>
<th>Best Integer</th>
<th>Best Bound</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>12+</td>
<td>227702.7631</td>
<td>227702.7507</td>
<td></td>
<td>0.30%</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>227702.7631</td>
<td>227702.7507</td>
<td>2710</td>
<td>0.30%</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>cutoff</td>
<td>227702.7631</td>
<td>227034.4173</td>
<td>6190</td>
<td>0.29%</td>
</tr>
<tr>
<td>148</td>
<td>83</td>
<td>227244.3466</td>
<td>227702.7631</td>
<td>9156</td>
<td>0.29%</td>
</tr>
<tr>
<td>206</td>
<td>100</td>
<td>cutoff</td>
<td>227702.7631</td>
<td>11789</td>
<td>0.27%</td>
</tr>
<tr>
<td>281</td>
<td>135</td>
<td>227702.7631</td>
<td>227179.3228</td>
<td>14822</td>
<td>0.23%</td>
</tr>
<tr>
<td>349</td>
<td>166</td>
<td>227409.3745</td>
<td>227702.7631</td>
<td>17941</td>
<td>0.22%</td>
</tr>
<tr>
<td>411</td>
<td>184</td>
<td>227634.5232</td>
<td>227702.7631</td>
<td>19289</td>
<td>0.20%</td>
</tr>
<tr>
<td>467</td>
<td>195</td>
<td>227560.6384</td>
<td>227702.7631</td>
<td>21664</td>
<td>0.19%</td>
</tr>
<tr>
<td>534</td>
<td>223</td>
<td>227494.7655</td>
<td>227702.7631</td>
<td>24768</td>
<td>0.19%</td>
</tr>
<tr>
<td>789</td>
<td>260</td>
<td>cutoff</td>
<td>227702.7631</td>
<td>36761</td>
<td>0.16%</td>
</tr>
</tbody>
</table>

Elapsed time = 6.20 sec. (4102.59 ticks, tree = 2.69 MB)

Cover cuts applied: 14
Implied bound cuts applied: 291
Flow cuts applied: 4
Mixed integer rounding cuts applied: 17
Lift and project cuts applied: 1
Introduction to Modeling Language and Solver

Modeling Language: Separation between Model and Data

knapsack.mod

```plaintext
# Number of items
param NumItem > 0;

# Set of items
set ITEMS := 1..NumItem; # Creates set {1, ..., NumItem}

# Other parameters
param Limit > 0;
param Value {ITEMS} >= 0;
param Weight {ITEMS} >= 0;

# Variable definition
var x {ITEMS} binary;

# Objective function
maximize TotalValue: sum {j in ITEMS} Value[j] * x[j];

# Weight limit constraint
subject to WeightLimit:
  sum {j in ITEMS} Weight[j] * x[j] <= Limit;
```

knapsack6.dat

```plaintext
param NumItem := 6;
param Limit := 14;

param: Value Weight :=
  1  12  2
  2  10  1
  3  25  5
  4  11  2.5
  5  30  4
  6  24  3;
```
Command Script: An Example

# Reset AMPL
reset;

# Load AMPL model and data
model knapsack.mod;
data knapsack500.dat;

# Set solver parameters
option relax_integality 0;
option cplex_options 'mipdisplay=2 integrality=1e-9 optimality=1e-9 timelimit=3600';

# Solve the problem
solve;

# Display optimum and solution
display TotalValue;
display {j in ITEMS: x[j]=1} x[j];

quit;
Introduction to Modeling Language and Solver

Modeling System + Solver

Model - Sets - Parameters - Variables - Objective function - Constraints

Modeling system

Solver

Command script - Model and data specification - Solver options - Display options - etc.

Data - Set elements - Parameter values

Result analysis
- Feasible solution?
- Optimal solution?
- Optimum and variable value
- Solution time
- etc.
AMPL Basics: Sets

- Simple sets (numbers or symbols)
  ```plaintext
  set ITEMS := 1, 2, 3, 4, 5, 6;
  set ITEMS := 1..6;
  set DAYS := Mon, Tues, Wed, Thurs, Fri, Sat, Sun;
  ```

- Indexed collection of sets
  ```plaintext
  # Declaration of base stations, test points, and coverage relation
  set BASESTATIONS;
  set TESTPOINTS;
  set COVERAGE {TESTPOINTS} within BASESTATIONS;
  
  # Numerical values in a data file
  set BASESTATIONS := 1..100;
  set TESTPOINTS := 1..10000;
  set COVERAGE[1] := 1 3 10 15;
  ```
Introduction to Modeling Language and Solver

AMPL Basics: Parameters

- Scalar parameter and parameters for set elements
  ```
  param Limit;
  param Capacity {Links};
  ```

- Bounds and default value
  ```
  param Limit >0;
  
  param MaxCapacity;
  param Capacity {LINKS} >=0, <=MaxCapacity;
  
  param Cost {BASESTATIONS} >=0 default 1000;
  param Traffic {TESTPOINTS} >=0 default 0;
  ```

- Symbolic parameters
  ```
  set DAYS;
  param FirstDay symbolic in DAYS;
  param LastDay symbolic in DAYS;
  
  # Values in a data file
  set DAYS := Mon, Tues, Wed, Thurs, Fri, Sat, Sun;
  param FirstDay := Mon;
  param LastDay := Sun;
  ```
Introduction to Modeling Language and Solver

AMPL Basics: Variables

- Similar to declaration of numerical parameters
- May have value and/or type restrictions

```plaintext
var x {ITEMS} binary;

var production {DAYS} >=0, integer;

var flow {(i,j) in LINKS} >=0, <=Capcity[i,j];

var location {BASESTATIONS} binary;

var serve {TESTPOINTS, BASESTATIONS} binary;

# A more efficient declaration using set COVERAGE
var serve {j in TESTPOINTS, i in COVERAGE[j]} binary;
```
Introduction to Modeling Language and Solver

AMPL Basics: Objective Function and Constraints

- Integer linear programming: The objective is a linear expression of the variables
  
  ```
  maximize TotalValue: sum {j in ITEMS} Value[j] * x[j];
  minimize TotalCost: sum {i in BASESTATIONS} Cost[i] * location[i];
  ```

- Single constraint
  
  ```
  subject to WeightLimit:
  sum {j in ITEMS} Weight[j] * x[j] <= Limit;
  ```

- Indexed collections of constraints (with condition)
  
  ```
  subject to ServiceCoverage {j in TESTPOINTS: traffic[j]>0}:
  sum {i in COVERAGE[j]} serve[j,i] >= 1;
  ```
Introduction to Modeling Language and Solver

**AMPL Basics: A Complete Model for Set Covering**

\[
\begin{align*}
\text{min} & \quad \sum_{i \in I} c_i x_i \\
\text{s. t.} & \quad \sum_{i \in I_j} x_i \geq 1, \forall j \in J \\
& \quad x \in \{0, 1\}
\end{align*}
\]

```plaintext
set CENTERS;
set POINTS;
set COVERAGE {POINTS} within CENTERS;
param Cost {CENTERS} >=0;
var x {CENTERS} binary;
minimize TotalCost: sum {i in CENTERS} Cost[i] * x[i];
subject to Covering {j in POINTS}:
- sum {i in COVERAGE[j]} x[i] >= 1;
```
Impact of Modeling on Problem Solving

Uncapacitated Facility Location

Can we reduce the model size?

\[
\begin{align*}
\min & \quad \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\
\text{s. t.} & \quad \sum_{i \in I} x_{ij} = 1, \forall j \in J \\
& \quad x_{ij} \leq y_i, \forall i \in I, \forall j \in J \\
& \quad x, y \in \{0, 1\}
\end{align*}
\]

\[
\begin{align*}
\min & \quad \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\
\text{s. t.} & \quad \sum_{i \in I} x_{ij} = 1, \forall j \in J \\
& \quad \sum_{j \in J} x_{ij} \leq |J| y_i, \forall i \in I \\
& \quad x, y \in \{0, 1\}
\end{align*}
\]
### Uncapacitated Facility Location (cont’d)

<table>
<thead>
<tr>
<th>Facilities</th>
<th>Customers</th>
<th>Aggregated Model (seconds)</th>
<th>Disaggregate Model (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>200</td>
<td>1.19</td>
<td>0.39</td>
</tr>
<tr>
<td>30</td>
<td>300</td>
<td>5.55</td>
<td>1.6</td>
</tr>
<tr>
<td>50</td>
<td>500</td>
<td>69.67</td>
<td>19.42</td>
</tr>
<tr>
<td>70</td>
<td>700</td>
<td>2362.67</td>
<td>764.30</td>
</tr>
</tbody>
</table>

---

**Impact of Modeling on Problem Solving**

<table>
<thead>
<tr>
<th>Facilities</th>
<th>Customers</th>
<th>Aggregated Model (seconds)</th>
<th>Disaggregate Model (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>200</td>
<td>1.19</td>
<td>0.39</td>
</tr>
<tr>
<td>30</td>
<td>300</td>
<td>5.55</td>
<td>1.6</td>
</tr>
<tr>
<td>50</td>
<td>500</td>
<td>69.67</td>
<td>19.42</td>
</tr>
<tr>
<td>70</td>
<td>700</td>
<td>2362.67</td>
<td>764.30</td>
</tr>
</tbody>
</table>

**AMPL Version 20060626 (Linux 2.6.9-5.EL)**

Node log . . .
Best integer = 1.433808e+05 Node = 0 Best node = 1.687869e+04
Best integer = 2.293498e+04 Node = 0 Best node = 1.687869e+04
Heuristic still looking.
Heuristic still looking.
Heuristic complete.
Best integer = 2.276127e+04 Node = 828 Best node = 1.831542e+04
Best integer = 2.267411e+04 Node = 1000 Best node = 1.838685e+04
Implied bound cuts applied: 2292
Flow cuts applied: 18
Times (seconds):
Solve = 2362.67
CPLEX 10.1.0: optimal integer solution within mipgap or absmipgap; objective 22674.11
1048766 MIP simplex iterations
16865 branch-and-bound nodes

**AMPL Version 20060626 (Linux 2.6.9-5.EL)**

Node log . . .
Best integer = 4.704366e+04 Node = 0 Best node = 2.203318e+04
Best integer = 2.295632e+04 Node = 0 Best node = 2.203318e+04
Heuristic still looking.
Best integer = 2.267411e+04 Node = 0 Best node = 2.203529e+04
Heuristic complete.
Gomory fractional cuts applied: 3
Using devex.
Times (seconds):
Solve = 764.304
CPLEX 10.1.0: optimal integer solution; objective 22674.11
157501 MIP simplex iterations
36 branch-and-bound nodes
Impact of Modeling on Problem Solving

**Coloring**

Define color set $C$

$$\min \quad y_{red} + y_{blue} + \ldots$$

s. t.  

$$x_{1,red} + x_{1,blue} + \ldots = 1$$

$$x_{2,red} + x_{2,blue} + \ldots = 1$$

$$\ldots$$

$$x_{1,red} \leq y_{red}$$

$$x_{1,blue} \leq y_{blue}$$

$$\ldots$$

$$x_{1,red} + x_{2,red} \leq 1$$

$$x_{1,blue} + x_{2,blue} \leq 1$$

$$\ldots$$

$$x \in \{0, 1\}, \ y \in \{0, 1\}$$

- $x_{ic} = 1$ if node $i$ has color $c$
- $y_c = 1$ if color $c$ is used (by any node)

- Symmetry: Solution search becomes inefficient
- Alternative model of set-covering type

Is this a good formulation? Other formulations?
### Impact of Modeling on Problem Solving

#### Coloring (cont’d)

Sample results for an extension of graph coloring:

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Model I (previous slide)</th>
<th>Model II (not shown)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best Solution</td>
<td>Time</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.1s</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>1s</td>
</tr>
<tr>
<td>30</td>
<td>21</td>
<td>10h</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
<td>10h</td>
</tr>
<tr>
<td>50</td>
<td>28</td>
<td>10h</td>
</tr>
<tr>
<td>60</td>
<td>31</td>
<td>10h</td>
</tr>
</tbody>
</table>

It may matter (a lot) which model you use