CS/ECE/ISYE524: Introduction to Optimization – Integer Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

April 22, 2024

Quadratic assignment problems (QAP)

Two big classes of problems seen so far:

- **GAP:** Assign jobs to machines. Each assignment has a price, each machine has a fixed cost and a max capacity.
- **TSP:** Find the optimal sequence of items (or cities). Each possible transition has an associated cost. (probably the most famous integer program)

One more important class of problems:

- **QAP:** Assign facilities to fixed locations. The facilities have known communication requirements and incurred costs depend on the distances between the facilities. (second most famous integer program – harder than TSP)

QAP example: shopping mall

A small shopping mall has four shop locations. The walking distance, in feet, between all pairs of locations are shown below. Four shops, designated A, B, C, D, are to be assigned to the four locations in such a way that customers traveling between pairs of shops will not walk too far. We have data on the number of customers per week that travel between the shops, shown below.

Distance	1	2	3	4
1	0	80	150	170
2		0	130	100
3			0	120
4	d_{ij}			0

Flow	A	B	C	D
A	0	5	2	7
B		0	3	8
C			0	3
D	f_{ij}			0

- let $x_{ij} = 1$ if shop i is in location j.

QAP example: shopping mall

- Each shop can only be in one location:

$$\sum_{j=1}^{L} x_{ij} = 1 \qquad \text{for } i = 1, \ldots, S$$

- Each location can only contain one shop:

$$\sum_{i=1}^{S} x_{ij} = 1 \qquad \text{for } j = 1, \ldots, L$$

- Cost depends on pairs of facilities! If shop i is in location j and shop k is in location ℓ, then between them we have a flow of f_{ik} and a distance of $d_{j\ell}$. The total cost is:

$$\frac{1}{2} \sum_{i=1}^{S} \sum_{j=1}^{L} \sum_{k=1}^{S} \sum_{\ell=1}^{L} f_{ik} d_{j\ell} x_{ij} x_{k\ell}$$

QAP example: shopping mall

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{S}\sum_{j=1}^{L}\sum_{k=1}^{S}\sum_{\ell=1}^{L} f_{ik}d_{j\ell}x_{ij}x_{k\ell} \\
\text{subject to:} \quad & \sum_{j=1}^{L} x_{ij} = 1 \qquad \text{for } i = 1,\ldots,S \\
& \sum_{i=1}^{S} x_{ij} = 1 \qquad \text{for } j = 1,\ldots,L \\
& x_{ij} \in \{0,1\}
\end{aligned}$$

- cost is quadratic in the variables! Can we linearize?
- define new binary variable: $z_{ijk\ell} = x_{ij}x_{k\ell}$

QAP example: shopping mall

$$\begin{aligned}
\underset{x,z}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{S}\sum_{j=1}^{L}\sum_{k=1}^{S}\sum_{\ell=1}^{L} f_{ik}d_{j\ell}z_{ijk\ell} \\
\text{subject to:} \quad & \sum_{j=1}^{L} x_{ij} = 1 \qquad \text{for } i = 1,\ldots,S \\
& \sum_{i=1}^{S} x_{ij} = 1 \qquad \text{for } j = 1,\ldots,L \\
& x_{ij} \in \{0,1\} \\
& z_{ijk\ell} = x_{ij}x_{k\ell}
\end{aligned}$$

- Equivalent to: $(z_{ijk\ell} = 1) \iff (x_{ij} = 1) \wedge (x_{k\ell} = 1)$

QAP example: shopping mall

How do we model: $(z_{ijk\ell} = 1) \iff (x_{ij} = 1) \wedge (x_{k\ell} = 1)$?

- we saw this when we discussed logic constraints!
  - ($\Longrightarrow$): $x_{ij} \geq z_{ijk\ell}$ and $x_{k\ell} \geq z_{ijk\ell}$
  - ($\Longleftarrow$): $x_{ij} + x_{k\ell} \leq z_{ijk\ell} + 1$
- But in fact we (may) need only to enforce $x_{ij} + x_{k\ell} \leq z_{ijk\ell} + 1$. **Why?**
  - When $(x_{ij} = 1) \wedge (x_{k\ell} = 1)$, then $x_{ij} + x_{k\ell} \leq z_{ijk\ell} + 1$ forces $z_{ijk\ell} = 1$.
  - When not both $(x_{ij} = 1)$ and $(x_{k\ell} = 1)$ then the constraint $x_{ij} + x_{k\ell} \leq z_{ijk\ell} + 1$ *allows* $z_{ijk\ell} = 0$. Because $z_{ijk\ell}$ appears in the objective, multiplied by a positive coefficient, and the objective is minimized, we have in fact that $z_{ijk\ell} = 0$.

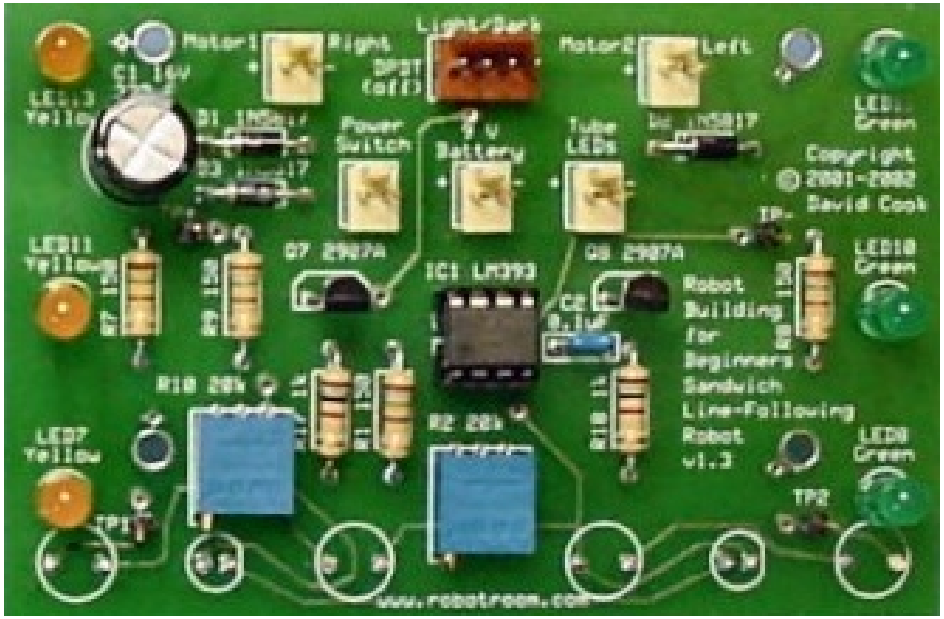
QAP example: shopping mall

$$
\begin{aligned}
\underset{x,z}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{S}\sum_{j=1}^{L}\sum_{k=1}^{S}\sum_{\ell=1}^{L} f_{ik}d_{j\ell}z_{ijk\ell} & \\
\text{subject to:} \quad & \sum_{j=1}^{L} x_{ij} = 1 & \forall i \\
& \sum_{i=1}^{S} x_{ij} = 1 & \forall j \\
& x_{ij} \geq z_{ijk\ell} \quad \text{and} \quad x_{k\ell} \geq z_{ijk\ell} & \forall i,j,k,\ell \\
& x_{ij} + x_{k\ell} \leq z_{ijk\ell} + 1 & \forall i,j,k,\ell \\
& x_{ij}, z_{ijk\ell} \in \{0,1\} & \forall i,j,k,\ell
\end{aligned}
$$

- Julia code: QAP.ipynb

QAP example: circuit layout

- Place n electronic modules in n predetermined positions on a backplate.
- We are given a wiring specification that tells us how the various modules must be connected.



- Identical to the facility location problem:
  - f_{ij} is the number of wires between module i and module j
  - d_{ij} distance between positions i and j on the backplate.
  - Minimize total length of wire used.

Special ordered sets

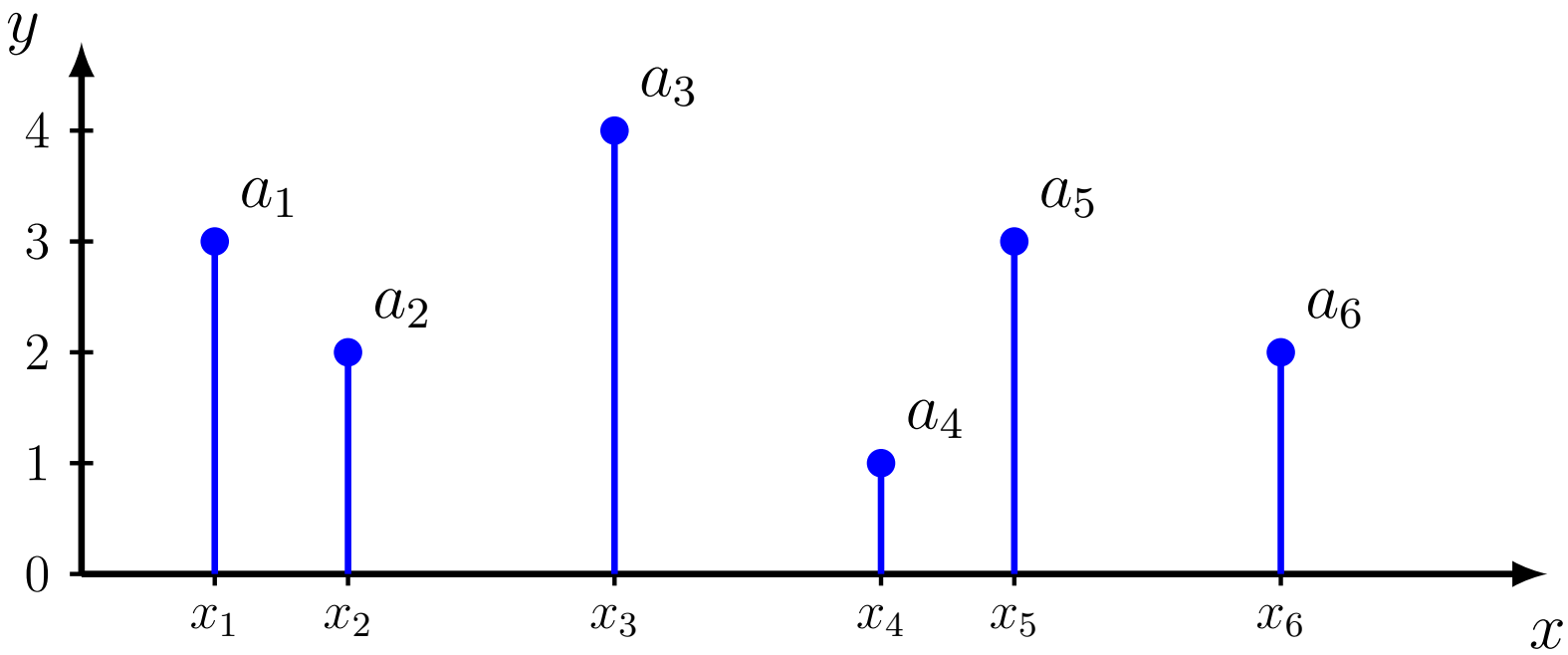
- Another type of constraint that is standard among many solvers is called the *special ordered set* (SOS).
- We saw one type of SOS constraint in the context of a variable belonging to a discrete set of values. There are other types of SOS constraints (we will see them next!)
- All SOS constraints can be implemented using logic tricks but some solvers also allow you to specify them explicitly.
- However, we still need to know how to implement them in logic.

SOS1 (type 1) constraint

SOS1 constraint: The variables $\{x_1, \ldots, x_m\}$ satisfy an SOS1 constraint if *at most one of them is nonzero*.

- An SOS1 constraint represents a multiple-choice notion.
- Standard use: representing a discrete-valued function

SOS1 (type 1) constraint



- Let (x_i, a_i), $i = 1, \ldots, m$ be the admissible (x, y) pairs.
- write: $x = \sum_{i=1}^{m} x_i \lambda_i$ and $y = \sum_{i=1}^{m} a_i \lambda_i$.
- constraint: exactly one of the λ_i is equal to 1, the rest are equal to zero.

SOS1 constraint

Using SOS1 constraint

$$y = \sum_{i=1}^{m} a_i \lambda_i$$

$$\sum_{i=1}^{m} \lambda_i = 1, \quad \lambda_i \geq 0$$

$\{\lambda_1, \ldots, \lambda_m\}$ is SOS1

Using algebraic forumlation

$$y = \sum_{i=1}^{m} a_i \lambda_i$$

$$\sum_{i=1}^{m} \lambda_i = 1$$

$$\lambda_i \in \{0, 1\}$$

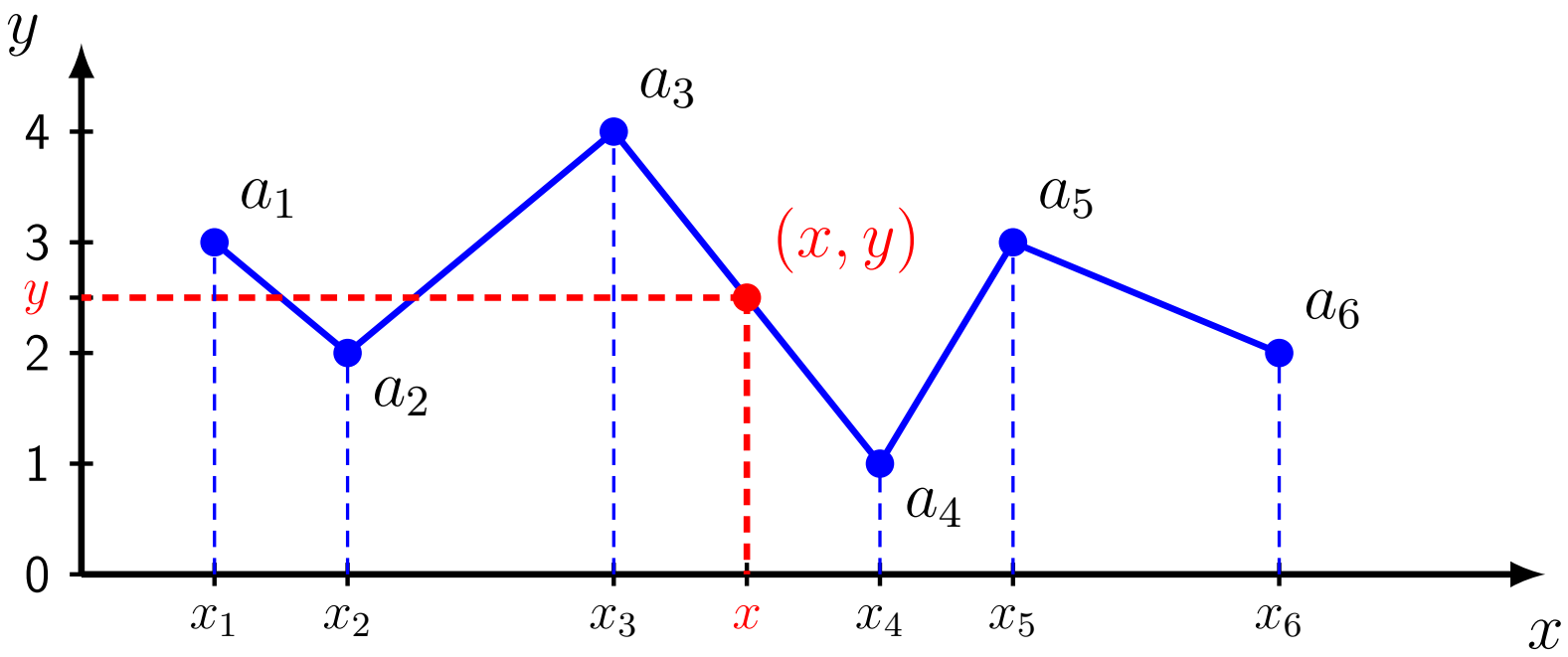
- The λ_i are real (not binary) in the SOS1 formulation
- Solver will still use binary variables internally, of course
- IJulia example: SOS.ipynb

SOS2 (type 2) constraint

SOS2 constraint: The variables $\{x_1, \ldots, x_m\}$ satisfy an SOS2 constraint if *at most two of them are nonzero*. Also, nonzero elements must be consecutive.

- SOS2 constraints are typically used to represent piecewise-linear functions.

SOS2 (type 2) constraint

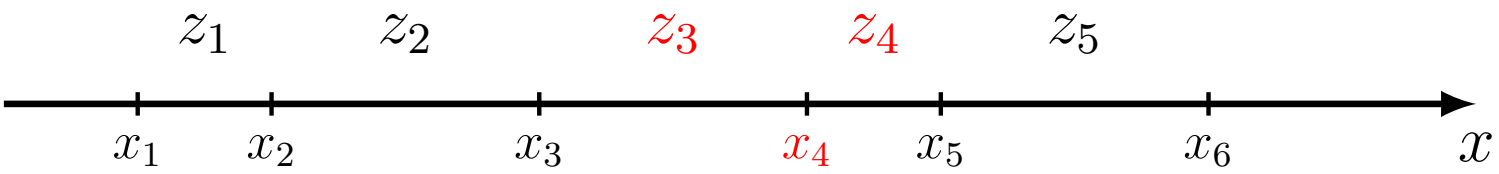


- Let (x_i, a_i), $i = 1, \ldots, m$ be the transition points.
- $x = \sum_{i=1}^{m} x_i \lambda_i$ and $y = \sum_{i=1}^{m} a_i \lambda_i$ with $\sum_{i=1}^{m} \lambda_i = 1$.
- If $x_i < x < x_{i+1}$ then $\lambda_i + \lambda_{i+1} = 1$ and the other λ_j are zero. Then, $x = \lambda_i x_i + \lambda_{i+1} x_{i+1}$ and $y = \lambda_i a_i + \lambda_{i+1} a_{i+1}$

SOS2 (type 2) constraint

How do we represent the constraint that at most two of the variables $\{\lambda_1, \ldots, \lambda_m\}$ are nonzero, and nonzero variables must be consecutive?

- Let $\{z_1, \ldots, z_{m-1}\}$ be binary with $\sum_{i=1}^{m-1} z_i = 1$.
- If z_{i-1} and z_i are zero, then weight $\lambda_i = 0$.
- Just take care for the endpoints.



SOS2 constraint

$$y = \sum_{i=1}^{m} a_i \lambda_i$$

$$\sum_{i=1}^{m} \lambda_i = 1, \quad \lambda_i \geq 0$$

$\{\lambda_1, \ldots, \lambda_m\}$ is SOS2

- Extra binary variables needed in algebraic formulation.
- Solver still uses binary variables for SOS2.

(One) Algebraic forumlation

$$y = \sum_{i=1}^{m} a_i \lambda_i$$

$$\sum_{i=1}^{m} \lambda_i = 1, \qquad \lambda_i \geq 0$$

$$\lambda_1 \leq z_1$$

$$\lambda_i \leq z_{i-1} + z_i, \quad i = 2, \ldots, m-1$$

$$\lambda_m \leq z_{m-1}$$

$$\sum_{i=1}^{m-1} z_i = 1, \qquad z_i \in \{0, 1\}$$

Piecewise Linear Modeling

- There are lots of other ways to model piecewise linear functions
- The one above is called the convex combination model

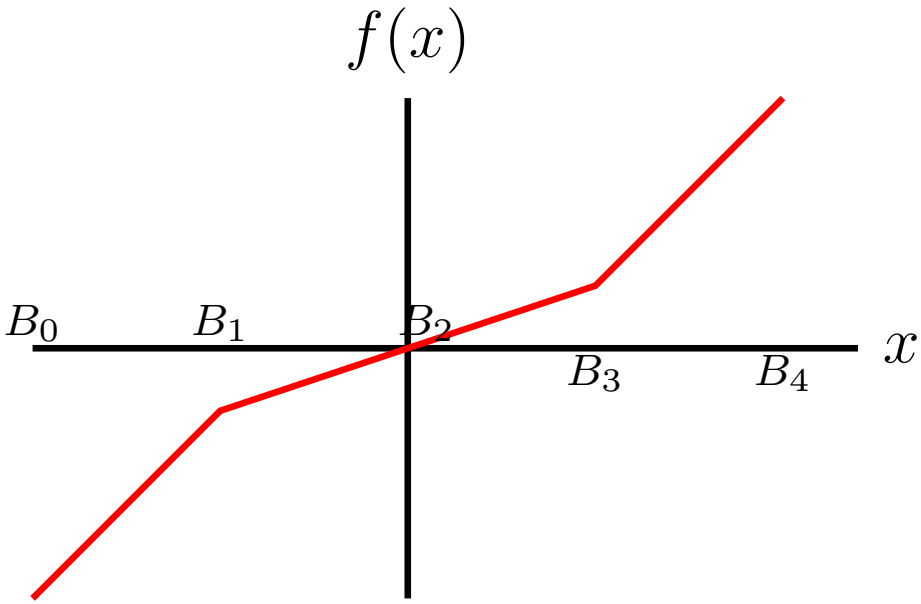
Other Methods

- Incremental model
- Multiple Choice Model
- Disaggregated Convex Combination Model
- Logarithmic Model

Multiple Choice Model

- Want to model $(\forall i)$

$$f(x) = m_i x + c_i, \quad x \in [B_{i-1}, B_i]$$



Introduce New Variables

- $w_i := x$ if $x \in [B_{i-1}, B_i]$
- $z_i := 1$ if $x \in [B_{i-1}, B_i]$
- $t := f(x)$

Multiple Choice Model

- $w_i := x$ if $x \in [B_{i-1}, B_i]$
- $b_i := 1$ if $x \in [B_{i-1}, B_i]$
- $t := f(x)$

$$x = \sum_{i=1}^{n} w_i,$$

$$t = \sum_{i=1}^{n} (m_i w_i + c_i z_i)$$

$$B_{i-1} z_i \leq w_i \leq B_i z_i \qquad \forall i = 1, \ldots, n$$

$$1 = \sum_{i=1}^{n} z_i$$

PW-Linear Functions

- You are a beer producer who needs to buy 800 pounds of barley.
- There are three suppliers who are willing to provide barley, quoting prices for various quantities of the product.
- Each supplier offers decreasing prices for increased lot size, in the form of incremental discounts.
- The quoted prices for each quantity are given in the table below.

Supplier 1		**Supplier 2**		**Supplier 3**	
#	**Price**	**#**	**Price**	**#**	**Price**
[0,100]	9.2	[0,50]	9	[0,100]	11
[100,200]	9	[50,250]	8.5	[100,300]	8.5
[200,800]	7	[250,800]	8.3	[300,800]	7.5

Piecewise Linear

- The costs are incremental. For example, if you wish to purchase 300 pounds of barley from supplier 1, you would pay

$$100(9.2) + (200 - 100)(9) + (300 - 200)7 = 2520.$$

- You would like to buy your barley at the least total cost, yet not buy too much from any one supplier.
- You can buy at most 40% of the total required from any one supplier.

Modeling Costs

- Let $f_1(x_1)$ be the cost of the amount ordered from supplier 1.

$$\begin{aligned} f_1(x_1) &= 9.2x_1 && x_1 \in [0, 100] \\ f_1(x_1) &= 100(9.2) + (x_1 - 100) * 9 = 9x + 20, && x_1 \in [100, 200] \\ f_1(x_1) &= 100(9.2) + 200(9) + 7(x_1 - 200) = 7x_1 + 420 && x_1 \in [200, 800] \end{aligned}$$

- Introduce a variable t_1 to represent the cost, and binary variables $y_{1,k}$ for $k = 1, 2, 3$ and continuous variables $w_{1,k}$ for $k = 1, 2, 3$

PW Linear Model: First Function

$$
\begin{aligned}
w_{11} + w_{12} + w_{13} &= x_1 \\
y_{11} + y_{12} + y_{13} &= 1 \\
9.2w_{11} + 9w_{12} + 7w_{13} + 20y_{12} + 420y_{13} &= t_1 \\
0 \leq w_{11} &\leq 100y_{11} \\
100y_{12} \leq w_{12} &\leq 200y_{12} \\
200y_{13} \leq w_{13} &\leq 800y_{13}.
\end{aligned}
$$

- Do the same for other functions and put it all together.

CS/ECE/ISYE524: Introduction to Optimization – Integer Optimization Models

Jeff Linderoth

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

April 22, 2024