Modelling Aspects

PD Dr. Timo Berthold
Private Lecturer @ TU Berlin
Director, Mixed-Integer Optimization @ FICO

Agenda

- Building a model
- 0/1 vs general integer, assignment formulations
- TSP example
- Combinatorial Constraints
- Indicator Constraints
- How not to do it
- Concluding remarks

Getting grounded

- "Art is a lie that makes us realize truth."
  Pablo Picasso

- A mathematical model is a lie that makes us focus on key aspects
  - by selecting, ignoring, simplifying, emphasizing, weighting, removing fuzziness, …

- This is also a creative process – an art!?
- Like art, mathematical modelling is also a craft
  - Can learn tools and skills
  - Some of them in this lecture series

Optimization solvers find optimal solutions for mathematical models! Not for real-world problems.
Success of an optimization project relies on the model approximating reality as good as possible.

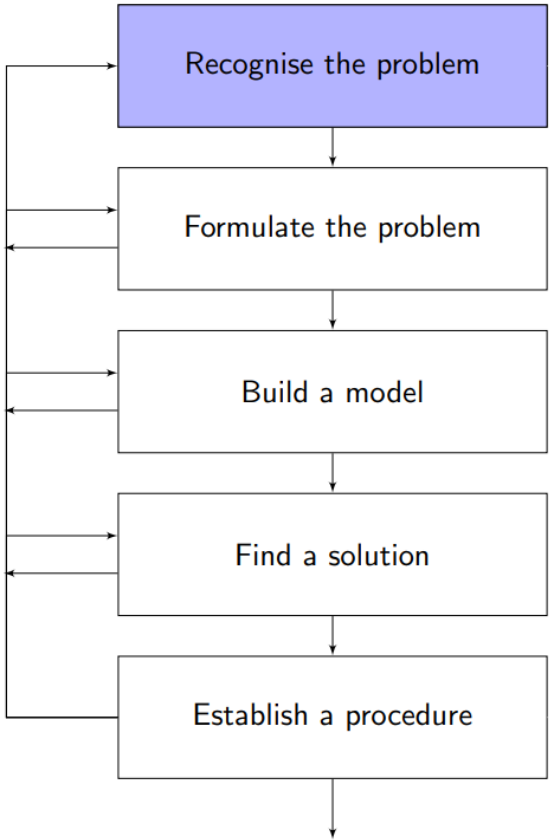
Building a Model

What Is Modelling?

- Describing a particular situation using a collection of logical and mathematical relationships
  - An objective function is used to evaluate alternative solutions
  - Constraints define the set of solutions that are feasible for the situation under consideration
- Why do we build models?
  - To capture essential aspects
  - Too many possibilities to enumerate
  - To evaluate what-if scenarios
  - Experimentation or simulation might not be possible
  - …
- Modelling is the first and recurring step in solving a real-world problem

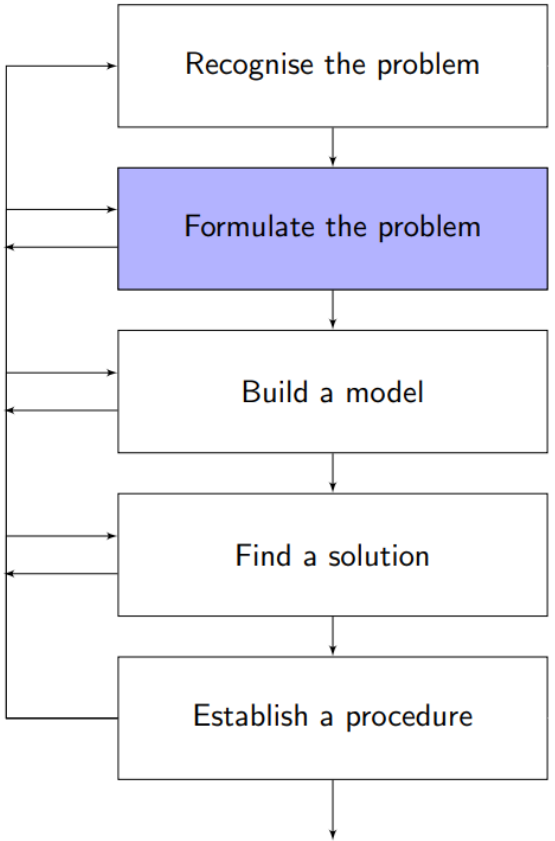
The Model Building Process

- Identify the problem
  - Can be abstract or real world
  - Single out a concrete question



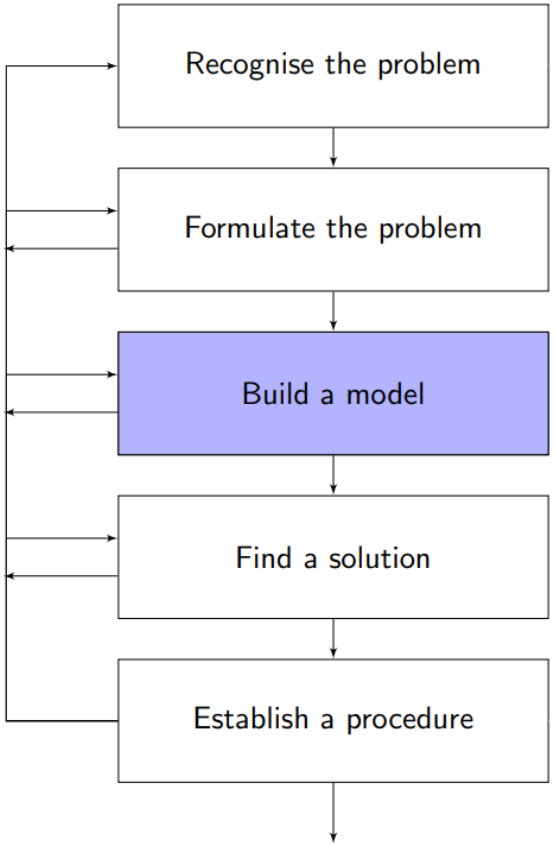
The Model Building Process

- Identify the important features
  - Define this problem in mathematical and logical notation
  - Never forget that your model is an abstraction of reality



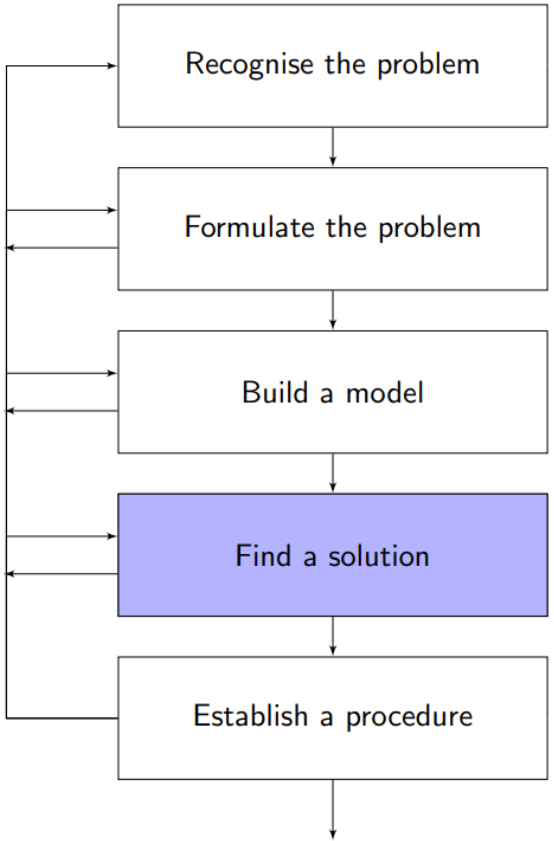
The Model Building Process

- Transfer the mathematical problem formulation to a model
  - Make use of available modelling tools
    - direct coding via API or modeling language
  - Think about alternative formulations

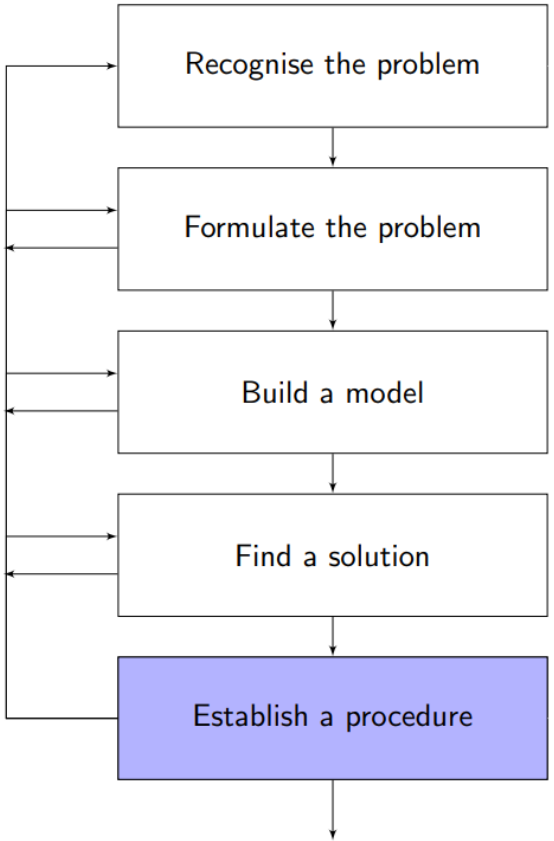


The Model Building Process

- Employ tools to solve the mathematical model
  - In our case, typically a MIP solver
  - Check validity of solution in practice



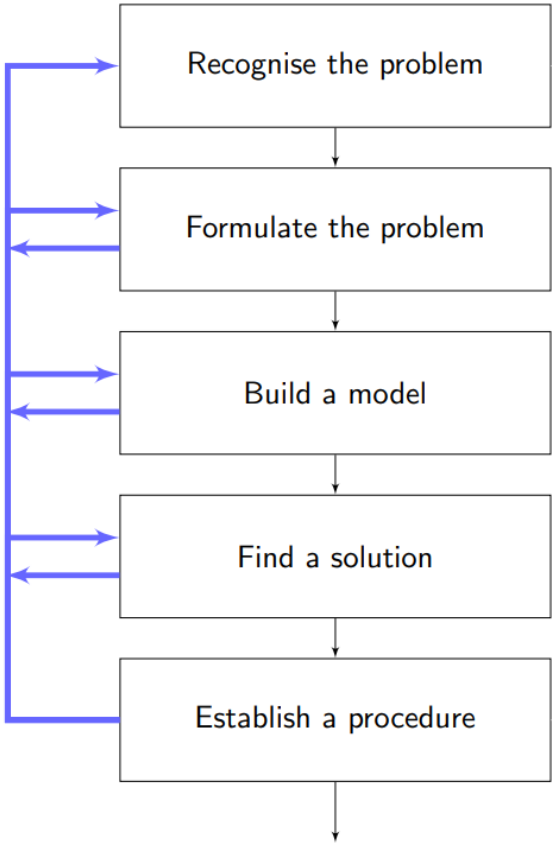
The Model Building Process



- Deployment:
  - Design a procedure to implement the solution
  - This is where a lot of OR projects fail!

The Model Building CYCLE

- FEEDBACK is crucial!
  - Each stage helps refine the previous stages
  - The modelling process aids the understanding of the problem
  - The problem understanding develops and the solution approach becomes clearer



Example: Knapsack

A burglar has a knapsack with 15kg capacity and breaks into a house with the following items: 1kg worth 2000€, 1kg worth 1000€, 2kg worth 2000 €, 4kg worth 10000€, 12kg worth 4000 € They want to take as much value as possible. How?

- What are the variables? What are the constraints? What is the objective?
  - Variables: $x_i \in \{0,1\}$: Do I take item i?
  - Constraint: Must not exceed capacity: $x_1 + x_2 + 2x_3 + 4x_4 + 12x_5 \leq 15$
  - Objective: Maximize revenue: $\max 2x_1 + x_2 + 2x_3 + 10x_4 + 4x_5$
- Final model:
  - $\max 2x_1 + x_2 + 2x_3 + 10x_4 + 4x_5$
    s.t. $x_1 + x_2 + 2x_3 + 4x_4 + 12x_5 \leq 15$
    $x_i \in \{0,1\}$

FICO

General Integers and Other Variable Types

Binaries or Integer Variables?

- Rule of thumb
  - Use general integers whenever they represent actual quantities and ordering is important
  - Whenever integers represent just „some different values“, use binaries instead
- Example: Sudoku

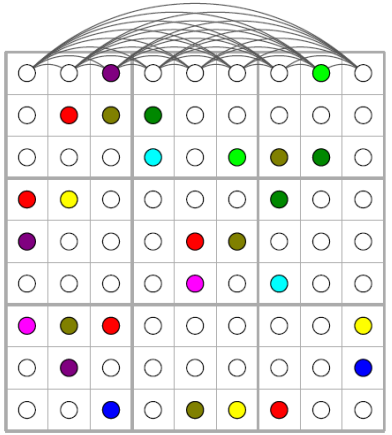
8			6			9		5
				2		3	1	
		7	3	1	8		6	
2	4						7	3
		2	7	9		1		
5				8			3	6
		3						

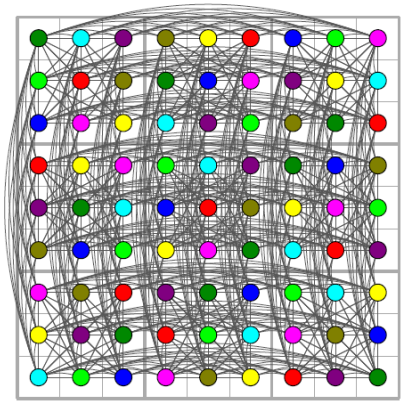
Naive approach:
Use 81 integer variables $1 \leq y_i \leq 9$
And then…?

Sudoku → Graph Coloring

- Each number corresponds to a color, each cell to a vertex
  - 9 binaries per cell: x_{ijk}
  - Exactly one color: $\sum_k x_{ijk} = 1 \ \forall i, j$
- Edges when two cells must not have the same color/number, e.g., $x_{1,1,1} + x_{1,2,1} \leq 1$
  - Can do better and add clique equations: $\sum_j x_{ijk} = 1 \ \forall i, k$
- Sudoku corresponds to the question: Is there a feasible 9-coloring of a partially colored graph with 27 9-cliques?

		7					9	
	3	8	2					
			4		9	8	2	
3	6					2		
7				3	8			
				5		4		
5	8	3						6
	7							1
		1		8	6	3		





2	4	7	8	6	3	1	9	5
9	3	8	2	1	5	7	6	4
1	5	6	4	7	9	8	2	3
3	6	5	9	4	7	2	1	8
7	2	4	1	3	8	6	5	9
8	1	9	6	5	2	4	3	7
5	8	3	7	2	1	9	4	6
6	7	2	3	9	4	5	8	1
4	9	1	5	8	6	3	7	2

Assignment Structure

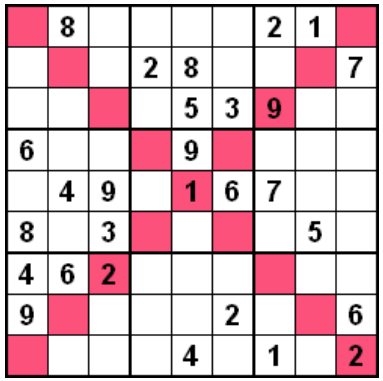
$$
\begin{aligned}
\text{maximize} \quad & 0 \\
\text{subject to} \quad & \sum_{v=1}^{9} x_{vrc} = 1 \text{ for } r, c \in [1,9] \\
& \sum_{r=1}^{9} x_{vrc} = 1 \text{ for } v, c \in [1,9] \\
& \sum_{c=1}^{9} x_{vrc} = 1 \text{ for } v, r \in [1,9] \\
& \sum_{r=3p-2}^{3p} \sum_{c=3q-2}^{3q} x_{vrc} = 1 \text{ for } v \in [1,9] \text{ and } p, q \in [1,3]
\end{aligned}
$$

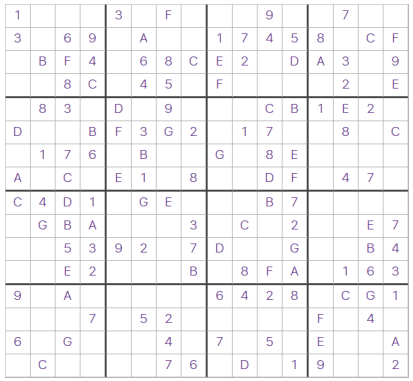
8			6			9		5
				2		3	1	
		7	3	1	8		6	
2	4						7	3
		2	7	9		1		
5				8			3	6
		3						

- Important concept: Assignment structure
  - Assignment problem: Given costs c_{ij} for assigning object i to person j
  - $\min \sum_{i,j} c_{ij} x_{ij}$
    s.t. $\sum_i x_{ij} = 1$
    $\sum_j x_{ij} = 1$
  - Easy, but a common substructure in other problems

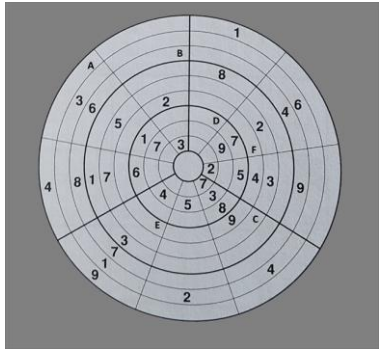
Many Variants, Similar Models

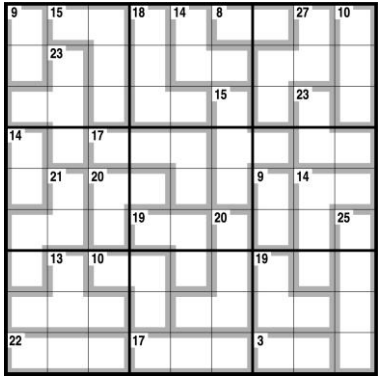
- X-Sudoku
- 16x16-Sudoku
- 3D-Sudoku

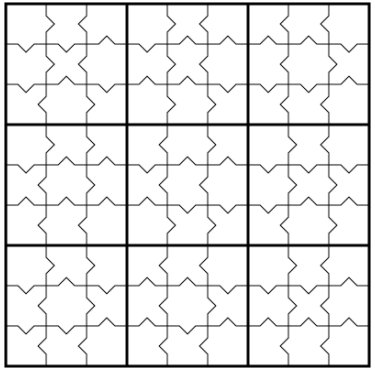




- Ensaimada
- Killer-Sudoku
- Comparison-Sudoku



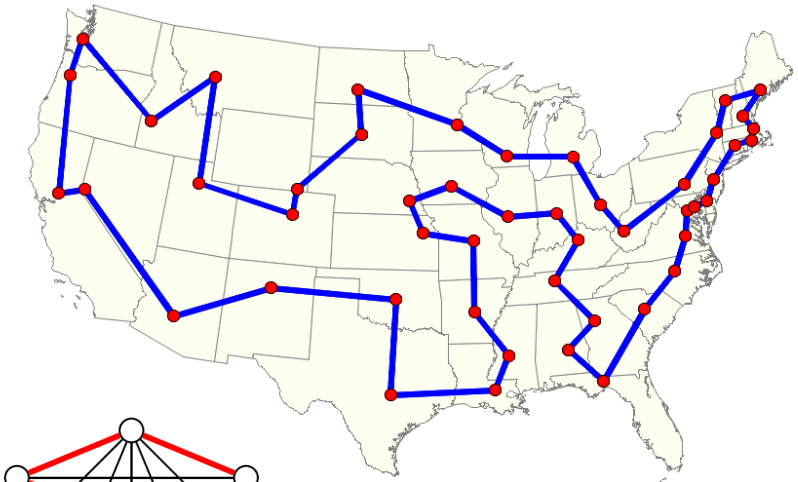


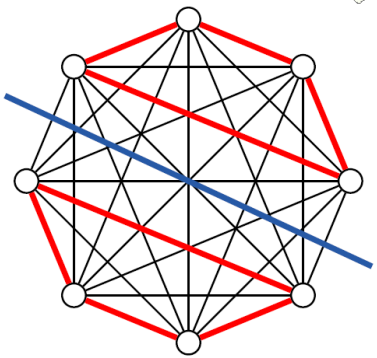


Many Choices in Modeling

TSP – the Most Famous Optimization Problem?

- TSP: Given a complete graph $G = (V, E)$ and distances c_{ij} for all $(i, j) \in E$:
  - Find a Hamiltonian cycle (tour) of minimum length
- Classical MIP formulation:
  - $\min \sum c_e y_e$
    s.t. $\sum_{e \in \delta(v)} y_e = 2$ for all v $\in V$
    $\sum_{e \in \delta(S)} y_e \geq 2$ for all $S \subset V, S \neq \emptyset$
    $y_e \in \{0,1\}$
- Highly efficient special purpose codes
  - Concorde
- And more tractable MIP formulations
  - MTZ, Vyve-Wolsey





Many Different Variations (Achterberg et al 2008)

- Alternative MIP formulation for TSP: Vyve-Wolsey (combines Miller-Tucker-Zemlin with MCF)
- Achterberg et al considered redundant changes, that do not change the LP bound (or the set of integer optimal solutions)
  - Yet, they can have a huge effect on solver performance

- Relaxations (that remove redundant constraints):
  - Remove upper bounds on u_i
  - Equality in the flow conservation constraints can be ommited:

$$\sum_{(i,l)\in\delta^-(l)} w_{il}^l - \sum_{(i,l)\in\delta^+(l)} w_{li}^l \geq 1$$

$$\sum_{(i,j)\in\delta^-(j)} w_{ij}^l - \sum_{(i,j)\in\delta^+(j)} w_{ji}^l \geq 0$$

Many Different Variations II

- Additional restrictions:
  - Ordering variables have to be integer: $u_i \in \mathbb{Z}$
  - Explicit bounds on flow variables: $w_{ij}^l \leq 1$
  - Flow variables have to be integer: $w_{ij}^l \in \mathbb{Z}$
  - Fix variables w_{ij}^l with $(i,j) \in \delta^+(V_l)$ to zero
- Some of those restrictions, the solver can figure out itself (E.g., fixing w_{ij}^l with $(i,j) \in \delta^+(V_l)$ to zero)
- Others lead to huge reductions (e.g., changing flow constraints to inequality)
- 64 cases, which fall into three clusters w.r.t. problem size (minor variations still occur)
  - Some solve in seconds, others not in a day
    - Note: Solvers of 2008, significantly weaker presolving

Quiz Time

- Modelling a 9x9 Sudoku as integer programs is best done with
  a) 81 general integer variables
  b) 81 binary variables
  c) 729 binary variables
- Where do most OR projects fail?
  a) Model formulation
  b) Deployment
  c) Solution finding
- Adding redundant information to a model:
  a) Will not change the performance
  b) Will always slow down the solver
  c) Might change performance dramatically, in either direction

Quiz Time

- Modelling a 9x9 Sudoku as integer programs is best done with
  a) 81 general integer variables
  b) 81 binary variables
  c) 729 binary variables
- Where do most OR projects fail?
  a) Model formulation
  b) Deployment
  c) Solution finding
- Adding redundant information to a model:
  a) Will not change the performance
  b) Will always slow down the solver
  c) Might change performance dramatically, in either direction

Quiz Time

- Modelling a 9x9 Sudoku as integer programs is best done with
  - a) 81 general integer variables
  - b) 81 binary variables
  - **c) 729 binary variables**
- Where do most OR projects fail?
  - a) Model formulation
  - **b) Deployment**
  - c) Solution finding
- Adding redundant information to a model:
  - a) Will not change the performance
  - b) Will always slow down the solver
  - c) Might change performance dramatically, in either direction

Quiz Time

- Modelling a 9x9 Sudoku as integer programs is best done with
  - a) 81 general integer variables
  - b) 81 binary variables
  - **c) 729 binary variables**
- Where do most OR projects fail?
  - a) Model formulation
  - **b) Deployment**
  - c) Solution finding
- Adding redundant information to a model:
  - a) Will not change the performance
  - b) Will always slow down the solver
  - **c) Might change performance dramatically, in either direction**

Auxiliary Variables and Constraints

Auxiliary Variables

- For modeling/readibility purposes:
  E.g., an artificial „cost“ variable $\min z;\ z = \sum x_i$
- For expressions that appear at many places in the model
  - Might help the LP solver (sparsification!)
  - Undoing „replacements“ is easier than finding suitable replacements
- For linearization/higher modeling constructs
  - Example: Converting an unbounded variable into two bounded

Example: Absolute Values

- Imagine, you want to use an absolute value $|x|$
- Can be modeled by using two auxiliary variables $x^+, x^- \geq 0$ and a binary $y \in \{0,1\}$
  - $x = x^+ - x^-$ and $|x| = x^+ + x^-, 0 \leq x^+ \leq Uy, 0 \leq x^- \leq |L|(1-y)$
  - Binary can be ommited when minimizing $|x|$
- Can be extended to model various min-max formulations

Auxiliary Constraints

- To link auxiliary variables to problem variables
- Stating connections between variables that are also implicitly given
  - Might give alternatives to the solver which constraints to use
  - Might be used for bound tightening (see Wednesday)

Time Discretization

- Continuous time may not be always necessary/possible to model
- Discretizing time can be helpful in many applications
  - Can inflate model size
  - Maybe not the same granularity is necessary for the whole model
- Commonly applied technique: Rolling horizon heuristics
  - Overlapping time periods
  - Fix decisions from previous periods
  - Compute fixing values for following period
  - Problem: Think about the final state!
- Requires care to not make model numerically infeasible!

Logical Constraints

Logical Constraints

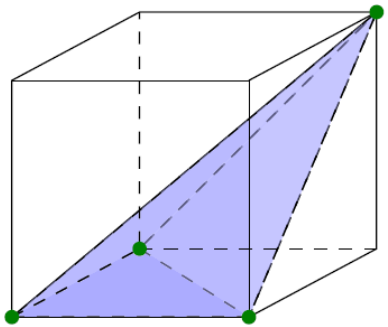
- For binary resultant r and operands x_i
  - $r = \text{AND}(x_1, \dots, x_n) : r = 1 \Leftrightarrow x_1 = \cdots = x_n = 1$
  - $r = \text{OR}(x_1, \dots, x_n) \quad : r = 1 \Leftrightarrow \exists i : x_i = 1$
  - $r = \text{XOR}(x_1, \dots, x_n) : r = 1 \Leftrightarrow |i : x_i = 1|$ is odd

- Relatively common constructs in modelling
- Easy to linearize
  - Or are they?

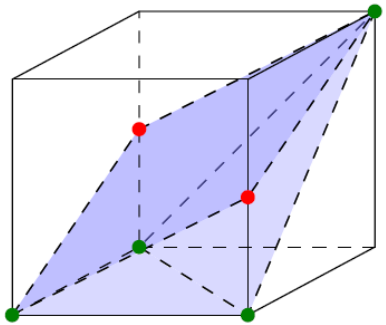
Advantages

- Convenient for modelling
  - Supported by many languages and solver interfaces
  - Together with MIN, MAX, ABS, PWL constraints
- Solver might decide dynamically how to linearize them
- Solver might use constraint specific presolving techniques
  - $r = \mathrm{AND}(x, y, z),\ z = \mathrm{AND}(a, b) \Rightarrow r = \mathrm{AND}(x, y, a, b)$
- Higher-level formulation might give additional structural insights that can be exploited

Linearizing AND Constraints



- $\sum_{i=1}^{n} x_i - r \leq n - 1$
- $x_i - r \geq 0$ for all $i \in \{1, \ldots, n\}$
- Strong relaxation
  - n+1 linear constraints
  - Only integral vertices (green)



- $\sum_{i=1}^{n} x_i - r \leq n - 1$
- $\sum_{i=1}^{n} x_i - nr \geq 0$
- Weak relaxation
  - 2 linear constraints
  - Contains fractional vertices (red)

Domain Propagation

- Procedure of iterating over constraints, for given (local) bounds of variables and deducing new bounds
  - Typically (at least for finite domain) until some form of consistency is reached
- Four rules for an AND constraint $r = \text{AND}(x_1, \dots, x_n)$
  1. $r = 1 \rightarrow x_i = 1$ for all i
  2. $x_i = 1$ for all $i \rightarrow r = 1$
  3. $\exists i\colon\ x_i = 0 \rightarrow r = 0$
  4. $r = 0$ and $x_i = 1 \forall i \in \{1, \dots, n\} \backslash \{j\} \rightarrow x_j = 0$

Domain Propagation

- There are four propagation rules for AND constraints
  1. $r = 1 \rightarrow x_i = 1$ for all i
  2. $x_i = 1$ for all $i \rightarrow r = 1$
  3. $\exists i\colon\ x_i = 0 \rightarrow r = 0$
  4. $r = 0$ and $x_i = 1 \forall i \in \{1, \dots, n\} \backslash \{j\} \rightarrow x_j = 0$
- Do we lose information by using a linearization?
  - Rule 1 is enforced by $\sum_{i=1}^{n} x_i - nr \geq 0$ or by all $x_i - r \geq 0$
  - Rule 2 is enforced by $\sum_{i=1}^{n} x_i - r \leq n - 1$
  - Rule 3 is enforced by $\sum_{i=1}^{n} x_i - nr \geq 0$ or by $x_i - r \geq 0$
  - Rule 4 is enforced by $\sum_{i=1}^{n} x_i - r \leq n - 1$
- Problem-specific propagation might still be more efficient to implement

Products of Variables

- AND constraints show how products of binary variables can be linearized
- Similar for product of one binary x_b and one bounded continuous $x_c \in [0, u]$
  - Introduce auxiliary continuous variable $y = x_b x_c$
  - $y \leq u x_b$
    $y \leq x_c$
    $y \geq x_c - u(1 - x_b)$
    $y \geq 0$
- The first constraint implies $x_b = 0$ (hence $x_b x_c = 0$) $\rightarrow y = 0$
  - and x_c can take an arbitrary value in $[0, u]$ (by the second and third constraint)
- The second and third constraint imply $x_b = 1 \rightarrow y = x_c$
  - and $y = x_c \in [0, u]$ by the first and last constraint
- For values $\mathrm{x_b} = \beta \in (0,1), y = x_b x_c$ does not hold, but it doesn‘t need to!

Indicator Constraints

Indicator Constraints

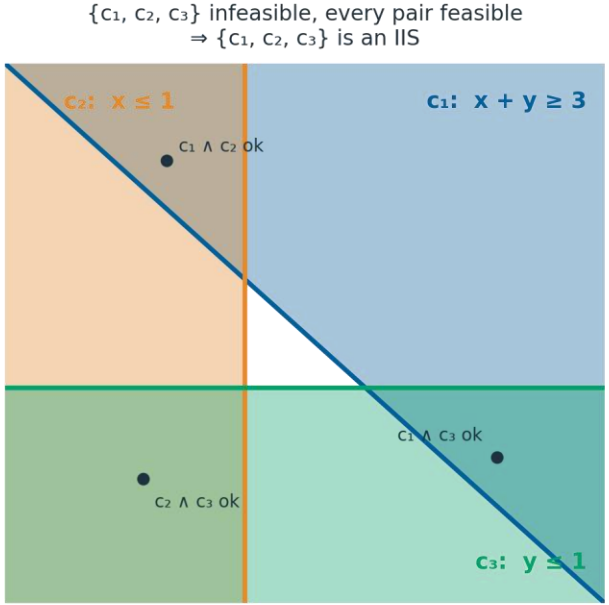
- Model If-then relations
- Most simple form: If $x = 1$ then $y = 0$ with x binary, y continuous
  - Often written as „x -> y=0“
  - x is called the indicator (variable)
- General form: Indicator for constraints $x_0 \rightarrow a^T x \leq b$
  - Constraint is enforced when $x_0 = 1$, relaxed otherwise
- Used to model that subsets of constraints have to hold
- Or for adding penalty terms when certain constraints do not hold
  - indicator variable appears in objective
- The same indicator variable can be used in different indicator constraints to model different scenarios

Indicator Constraints: Big-M Formulation

- Take indicator constraints $\mathrm{x}_0 \rightarrow a^T x \leq b$
- Linearize as $a^T x \leq b + (1 - x_0)\mathrm{M}$
  - Requires careful choice of M
  - E.g. $\mathrm{M} = \max(a^T x - b)$, but with user knowledge, much smaller values might be feasible
  - Too small M might lead to solutions being cut off
- Propagation, theoretically the same, but numerically it might be different…
  - Big-M formulations are known to be numerically cumbersome
  - For $y \leq 10000000x,\ x \in \{0,1\}$, the solution $x = 0.0000001, y = 1$ is feasible (and ε-„integer“)
    - See Thursday lecture
- Indicator formulation often solve slower because information is not present in the LP relaxation
- Choose your big-Ms wisely!!! Try both variants, indicators and big-Ms

Irreducible Infeasible Subsystem (IIS)

- Consider an infeasible constraint matrix $Ax \leq b$
- An IIS is a subset of constraints that is infeasible, but every proper subset of it is feasible
  - Identifies a small reason for infeasibility
- Related to Maximum Feasible Subset:
  - Modeled via indicators: $y_j \rightarrow a_j^T x \leq b_j$ for all constraints
  - $max \sum_{j=1}^{m} y_j$
    s.t. $y_j \rightarrow a_j^T x \leq b_j$
  - MaxFS is the complement of min. IIS-Cover



How NOT to Do It

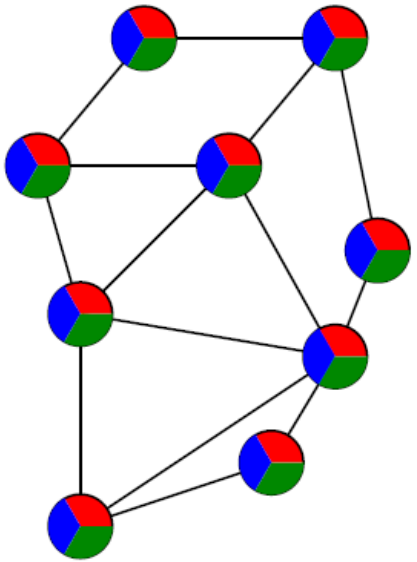
A few practical examples

Making Assumptions What Structure a MIP Solver Can Recognize

- Given a flow model with per-node flow conservation constraints
  - Capacity constraints on some subsets
  - A layered structure
  - S.t. the capacity constraints create cuts between two layers
  - And no bounds on the flow variables
  - „because this should follow from the structure“
- MIP solvers are great at making deductions from single constraints or pairs of constraints
  - Less so from a specific combinatorial structure that is itly captured in hundreds of constraints
  - In the present case, the solver will detect that this is a network flow, but that‘s about it
- You as a modeler are the „structure“ expert, try passing information to the solver

Destroying Structure That Is There

- Often, hard to prove optimality for symmetric models
  - If possible, choose a non-symmetric formulation
- MIP solvers employ sophisticated methods to handle symmetries
- Breaking them by hand („I fixed a few of the decisions“) might do more harm than good*
  - Also, it increases the risk of making a non-obvious error



- *= If your solver supports symmetry detection and exploitation
  - Most modern MIP solvers do
  - Most modern MINLP solvers don‘t
  - As often…try both?

Arbitrary Big-Ms

- Model with piecewise linear functions, modelled via SOS2
  - Last point in the PWL being essentially $[\infty, \infty^2]$
  - Corresponding big-Ms also made it into the objective and other constraints
- First reaction: Oh yes, we can easily use M=1000 instead of M=1000000000
- Second thought: There was only one breakpoint, so one could model this with a SOS1
  - Even with several breakpoints, one could use the SOS2 formulation for the regular PWL and a SOS1 for „going infinity“

Concluding Considerations

What Does a Good Model Look Like?

- Compact is not always better
  - There are huge models (>1M variables) that solve in seconds and small (<50 variables) that do not solve in days
- Ideally, the LP optimum should be close to the integer optimum (tight formulation)
- Small number of fractionals in the LP solution is a plus
- Fixing variables should have an impact on other variables (not too many degrees of freedom)
- Keep numerics under control: not too large span of coefficients
  - Not more than six orders of magnitude for single row/column
  - Not more than nine over the whole model
- Try to avoid big-M formulations

Things to Keep in Mind

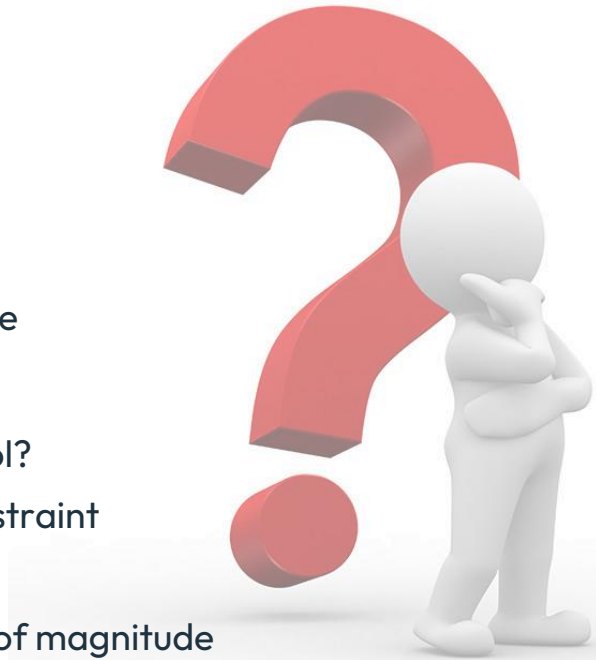
- The first modelling attempt often is infeasible or unbounded
  - MIP solvers are typically super fast in detecting those „trivial“ errors
  - IIS for the rescue!
- The „second“ attempt often produces unsatisfying solutions
  - Might violate some it constraints that were forgotten in the model
- MIP solvers prefer extremal solutions
  - Customers often do not
  - Most often, there are alternative optima
    - Or solutions almost as good that might fulfill some robustness considerations

Things to Keep in Mind

- Try to stress-test your model
  - Do the solutions for corner cases make sense?
  - Always ensure your solutions are at least two-optimal / locally optimal / not trivially improvable
- A solution is only always optimal (or [in]feasible) w.r.t. your model
  - ...and the data that was fed into your model
  - Slight violations might still be tolerable in practice
  - Often enough solving to exact optimality is not required (e.g., due to inaccurate data)
- Be prepared for a pushback

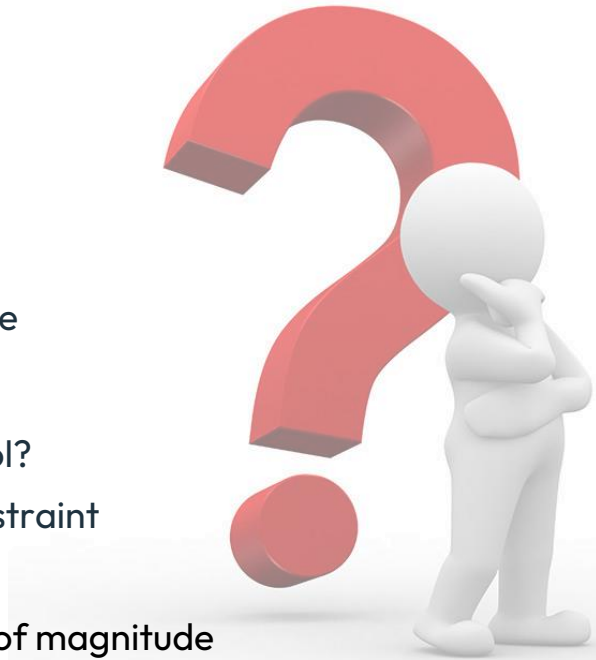
Quiz Time

- A product of binary variables
  a) Is a nonlinear structure and requires an MINLP solver
  b) Needs to be linearly approximated by the modeller
  c) Is a structure that many MIP solvers support
- Which of the following can NOT be easily linearized?
  a) Product of two bounded continuous variables
  b) Product of a binary and a bounded continuous variable
  c) Product of two binaries
- What is a good rule of thumb to keep numerics under control?
  a) Don‘t mix continuous and integer variables in one constraint
  b) Don‘t mix inequalities and equations in one model
  c) Don‘t use coefficients that span more than six orders of magnitude



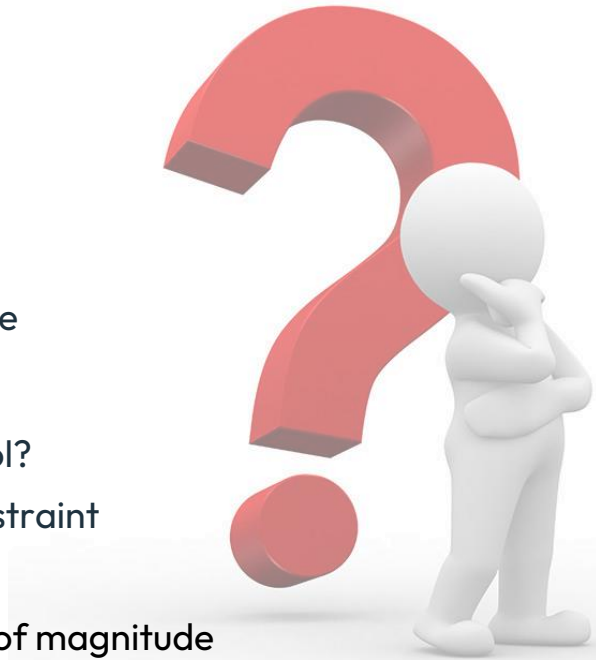
Quiz Time

- A product of binary variables
  a) Is a nonlinear structure and requires an MINLP solver
  b) Needs to be linearly approximated by the modeller
  c) Is a structure that many MIP solvers support
- Which of the following can NOT be easily linearized?
  a) Product of two bounded continuous variables
  b) Product of a binary and a bounded continuous variable
  c) Product of two binaries
- What is a good rule of thumb to keep numerics under control?
  a) Don‘t mix continuous and integer variables in one constraint
  b) Don‘t mix inequalities and equations in one model
  c) Don‘t use coefficients that span more than six orders of magnitude



Quiz Time

- A product of binary variables
  a) Is a nonlinear structure and requires an MINLP solver
  b) Needs to be linearly approximated by the modeller
  c) Is a structure that many MIP solvers support
- Which of the following can NOT be easily linearized?
  a) Product of two bounded continuous variables
  b) Product of a binary and a bounded continuous variable
  c) Product of two binaries
- What is a good rule of thumb to keep numerics under control?
  a) Don‘t mix continuous and integer variables in one constraint
  b) Don‘t mix inequalities and equations in one model
  c) Don‘t use coefficients that span more than six orders of magnitude



Quiz Time

- A product of binary variables
  a) Is a nonlinear structure and requires an MINLP solver
  b) Needs to be linearly approximated by the modeller
  c) Is a structure that many MIP solvers support
- Which of the following can NOT be easily linearized?
  a) Product of two bounded continuous variables
  b) Product of a binary and a bounded continuous variable
  c) Product of two binaries
- What is a good rule of thumb to keep numerics under control?
  a) Don‘t mix continuous and integer variables in one constraint
  b) Don‘t mix inequalities and equations in one model
  c) Don‘t use coefficients that span more than six orders of magnitude

Thank You!

PD Dr. Timo Berthold
Private Lecturer @ TU Berlin
Director, Mixed-Integer Optimization @ FICO