Introduction to mathematical programming

Julien Meier

April 6, 2021

Damn, it is about mathematics?!

Nope, it is not - at least not really if you're not delving into the theory.

This will be more about modelling of problems from practice and the theoretical basics.

Why should we care?

Because...

1. we can easily **solve difficult problems**
2. it serves as a **first approach**
3. it might help **approximating** problems
4. **Braun** loves it! (awesome for your PA)
5. it is widely used in **practice**

Agenda

1. Example problems
2. Basics of modelling
3. Applying the techniques
4. Staying realistic

What can we model?

A lot!

Let's look at some examples...

Example: Maximum flow (“easy“ problem)

Problem: Maximum flow

Goal: How much flow can you push from node s to node t while complying with capacity constraints.

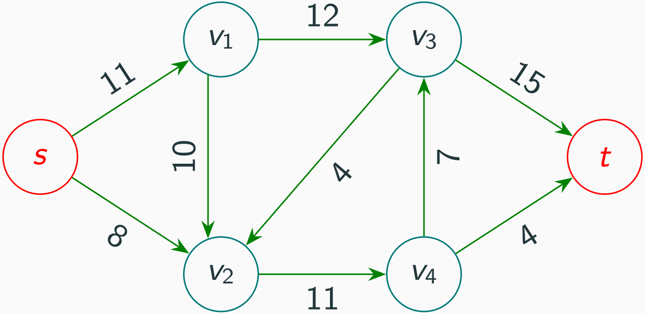


Figure 1: Example for maximum flow. Upper bound is 19 units due to the capacity constraints of the edges going into t.

Example: Minimum Vertex Cover ($\mathcal{NP}$-hard)

Problem: Minimum Vertex Cover

Goal: Find a subset of vertices that cover each edge while minimizing costs.

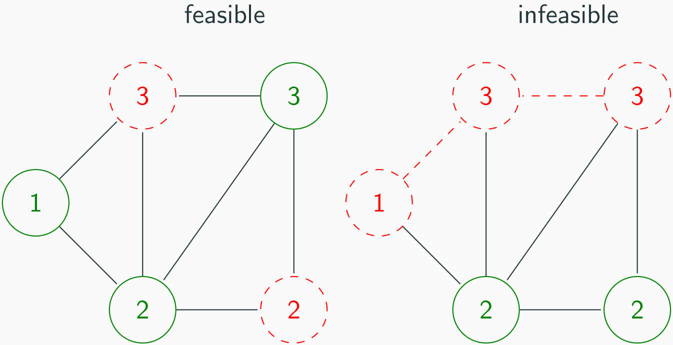


Figure 2: Example for minimum vertex cover. Numbers in nodes are associated costs. Dashed edges are not covered which renders the instance infeasible. All dashed nodes are not included in the subset.

Example: Graph coloring ($\mathcal{NP}$-hard)

Problem: Graph coloring

Goal: Color each node with adjacent nodes colored distinctly.

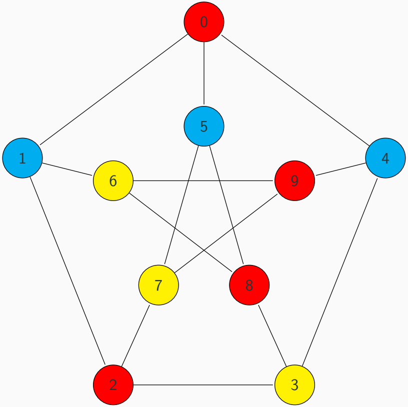


Figure 3: Example for graph coloring. Adjacent nodes are colored distinctly.

Example: Maximum Independent Set ($\mathcal{NP}$-hard)

Problem: Maximum Independent Set

Goal: Find a maximum subset of vertices that are not adjacent.

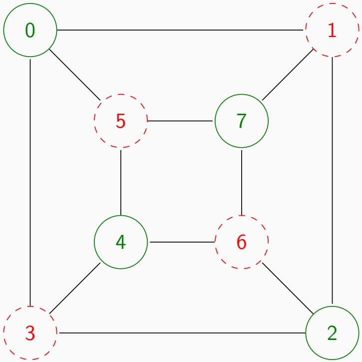


Figure 4: Example for maximum independent set. Dashed edges are excluded from the set. Note that none of the vertices included are adjacent.

Short “reminder“ on what's $\mathcal{NP}$-hard

A problem being $\mathcal{NP}$-hard means

“probably not solvable deterministically in sub-exponential time“

$\rightarrow$ difficult problem...

More precisely, a problem B is $\mathcal{NP}$-hard if you can reduce another $\mathcal{NP}$-hard problem A to it. Basic idea: Solve A by solving B.

Reduction: $A \rightarrow B$ (formulate problem A as problem B, solve it, then derive knowledge from that)

How do we model these problems?

Well, that's easy!

Using:

1. A bunch of **variables**
2. An **objective function** we want to *minimize*/*maximize*
3. A lot of **constraints** in the form of **inequalities**

Component 1: Variables

Variables $x_1, ..., x_n$ can be defined to be constrained to:

1. $x_i \in \mathbb{R}$
2. $x_i \in \mathbb{Z}$ (makes it difficult)

Using a suited solver and the "program" we define, we can get the variables' optimal values.

Component 2: Objective function (1)

The solver is given an objective function that should be *minimized*/*maximized*.
Thus, the optimal value is a *variable assignment* that has minimal/maximal value.

Component 2: Objective function (2)

How does it look like?

$$max\{\ 2x_1 + 3x_2 - 4x_3\ \} \tag{1}$$

Trivial! Why not set everything to either ∞ or $-\infty$?
In the real world, everything is constrained.

The objective function **should** be linear.

Component 2: Objective function (3)

What could we minimize/maximize?

Examples:

- Minimize the colors used in the coloring problem.
- Maximize the flow in the max flow problem.
- Minimize the cost in Minimum Vertex Cover.
- Maximize the cardinality of the Maximum Independent Set problem.

Component 3: Constraints (1)

Constraints are linear inequalities of the form:

$$3x_2 - 4x_3 \leq 5 \tag{2}$$

In general with m inequalities and n variables

$$\sum_{i=1}^{n} a_{j,i} \cdot x_i \leq b_j, \qquad j \in \{ 1, ..., m \} \tag{3}$$

with constants $a_{j,i} \in \mathbb{R}$, $b_j \in \mathbb{R}$ and decision variables x_i.

Component 3: Constraints (2)

What can we model using $\leq$-constraints?

Well, in the “easy“ case (none of the variables is integral)

1. $\geq$ by multiplying with -1.
2. $a = b$ by splitting into $a \leq b$ and $a \geq b$.

Component 3: Constraints (3)

We can define “binary“ values as follows:

$$0 \leq x \leq 1, \qquad x \in \mathbb{Z} \tag{4}$$

Using binary values we can create constraints of the form $a! = b$.

We define $x \in \mathcal{B} = \{\ 0, 1\ \}$ to indicate that $x \in \mathbb{Z}$ and $0 \leq x \leq 1$.

Putting it all together

Let's solve the Maximum Independent Set problem using a MILP:

Variables: $x_0, x_1, ..., x_7 \in \mathcal{B}$

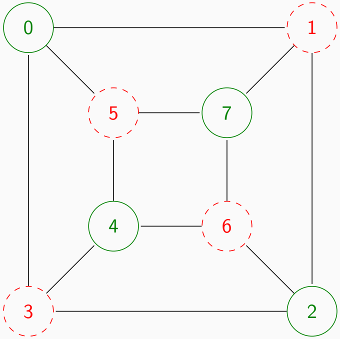
Objective: $max\{ \sum_{i=0}^{7} x_i \}$

Constraints:

For every edge $e = \{ i, j \}$,

we create a constraint:

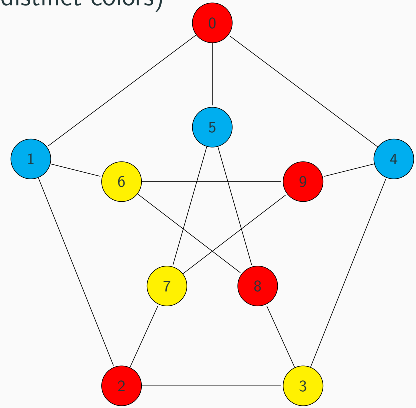
$x_i + x_j \leq 1$



Demo (1): Maximum Independent Set

One Ring to rule them all [...]

Let's do graph coloring (minimizing distinct colors)



Variables: $x_0, ..., x_9 \in \mathbb{Z}$, $x_\lambda \in \mathbb{R}$
Objective: $min\{\ x_\lambda\ \}$
Constraints:
For every node i:

$x_i \geq 0$

$x_i \leq x_\lambda$

For every edge $e = \{\ i, j\ \}$:

$x_i \neq x_j$

Demo (2):
Graph coloring

Invincible

Well...

Nope.

Counterexample

Problem: Circle packing into a rectangle.

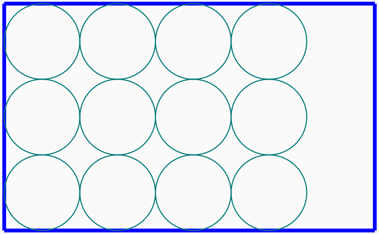


Figure 5: As many circles as possible should be packed into the rectangle without the circles overlapping. Well, for that we need quadratic constraints...

We will not solve that.

Thank you for staying.