Modeling the TSP as a standard optimization problem, I

- A key observation is that every tour of the TSP on $n$ cities can be viewed as a subgraph $(V, E')$ of the complete graph $K_n = (V, E)$ on $n$ nodes. *(We disregard orientation and starting point of the tour by doing so.)*
- Remember that the edges of the complete graph are the 2-element subsets of $n$:
  \[ E = \left\{ \{i,j\} : i, j = 1, \ldots, n \right\} \]

  Note that the order of $i$ and $j$ does not play any role:
  \[ \{i,j\} = \{j,i\} \]

  We can “encode” any subgraph $(V, E')$ with a “set” of 0/1 variables, one for each edge of the complete graph:
  \[
  x_{\{i,j\}} = \begin{cases} 
  1 & \text{if edge } \{i,j\} \text{ is present, i.e., } \{i,j\} \in E' \\
  0 & \text{if edge } \{i,j\} \text{ is not present}
  \end{cases}
  \]

  Thus, each vector $(x_{\{i,j\}})_{\{i,j\} \in E} \in \{0,1\}^E$ “encodes” a subgraph $(V, E')$. One way of writing this vector is as the upper triangle of a square $n \times n$ matrix – but how we write it, is not essential.

  Important is that this is a one-to-one correspondence between the combinatorial objects (“subgraphs”) and 0-1-vectors.
Next we wish to express the objective function.

- We wish to minimize the total length of the tour $T$, which we view as the edge set of a subgraph $(V, T)$ of the complete graph $K_n = (V, E)$.
- Using the notation $d(i, j)$ for the length of way from city $i$ to $j$ (or reversely – remember we deal with the symmetric case!), the total length is:

$$\text{length}(T) = \sum_{\{i,j\} \in T} d(i, j)$$

This summation is not “nice” – its domain of summation depends on the solution $T$. We prefer to sum over fixed domains of summation!

- Now we remember that we have 0/1 variables $x_{\{i,j\}}$ that are 1 if $\{i,j\} \in T$, and 0 otherwise. So it does not change anything if we multiply $d(i, j)$ by $x_{\{i,j\}}$:

$$\text{length}(T) = \sum_{\{i,j\} \in T} x_{\{i,j\}} d(i, j) = \sum_{\{i,j\} \in E} x_{\{i,j\}} d(i, j)$$

In the last step, we extended the domain of summation to all edges in $E$ – again, nothing happens, since all the added summands are 0.

- So now we have expressed the length of a tour as a linear function in our $x_{\{i,j\}}$ variables; note the domain of summation is independent of the tour!
Since all tours are subgraphs, but not all subgraphs are tours, we need to add constraints on our variables, to make sure that only tours are feasible solutions.

Remember that we call a vertex \( v \) and an edge \( e \) **incident** if \( v \in e \), i.e., \( v \) is one of the endpoints of the edge \( e \).

The **degree** of a vertex \( v \) is the number of edges incident with it.

A key observation is that in a tour, viewed as a subgraph of \( K_n \), every vertex has degree 2 (if we oriented the tour, one edge would go in, one edge would go out).

So let's write down this insight as a constraint, for every vertex \( i \):

\[
\sum_{j : \{i,j\} \in E} x_{\{i,j\}} = 2.
\]

Again, the domain of summation is independent of the tour, so this equation is a linear constraint in our variables \( x_{\{i,j\}} \).
Putting the TSP model in the computer

- In the computer, it is convenient to represent edges (2-element sets) \( \{i, j\} \) as (ordered) pairs \((i, j)\) with \( i < j \).

- Thus, for a 6-city TSP, we would be using variables named \( x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{23}, x_{24}, x_{25}, x_{26}, x_{34}, x_{35}, x_{36}, x_{45}, x_{46}, x_{56} \)

- When we write down the constraint

\[
\sum_{j: \{i, j\} \in E} x_{\{i,j\}} = 2,
\]

by using the ordered-pair representation, we actually write

\[
\sum_{j < i} x_{(j,i)} + \sum_{j > i} x_{(i,j)} = 2.
\]

- The resulting optimization model in ZIMPL is found in the file \( \text{tsp6-1.zpl} \).
Using parameters in ZIMPL

- In the example, I have used certain (made-up) distances \( d(i,j) \).
- Often we are interested in running the same optimization problem for different “data” – in our case with a different set of distances.
- For this purpose, it is useful to use parameters (named constants) in ZIMPL. This allows to decouple specific data of a problem instance from the modeling that is valid for a whole class of problems.

  Syntax:

  ```
  param NAME := VALUE;
  ```

- See tsp6-2.zpl
Using SCIP on tsp6-1.zpl or tsp6-2.zpl, we obtain an optimal solution of

\[ x_{12} = x_{23} = x_{13} = x_{45} = x_{46} = x_{56} = 1, \quad \text{all other } x_{ij} = 0 \]

This does not look like a tour! What are we missing?

We are not missing anything; to the contrary, we have **too much**!
Our integer program has many feasible solutions that do not correspond to tours. (The corresponding subgraphs do satisfy the degree-2 conditions.)
In the example, we obtained a feasible solution that corresponds to 2 cycles of length 3.

Dually speaking, we are missing something: We need to add more inequalities that “forbid” short cycles.

**Lemma**

Let \( T \subseteq E \) be any TSP tour on \( K_n \).
Let \( S \subseteq V \) be a vertex subset of size \( 3 \leq |S| \leq n - 3 \). Then

\[ |\{ \{i,j\} \in T : i,j \in S \}| \leq |S| - 1. \]
Theorem (Complete TSP Formulation)

The 0/1 solutions of the system

\[
\sum_{j: \{i,j\} \in E} x_{\{i,j\}} = 2
\]

for all vertices \(i = 1, \ldots, n\)

\[
\sum_{\{i,j\} \in E: i,j \in S} x_{\{i,j\}} \leq |S| - 1
\]

for all \(S\) in \(K_n\) with \(3 \leq |S| \leq n-3\)

are in one-to-one-correspondence with the TSP tours on \(K_n\).

- How many short-cycle inequalities?

\[
2^n - 2 \binom{n}{0} - 2 \binom{n}{1} - 2 \binom{n}{2}
\]

For \(n = 15\): about 32000.

- Shall we continue with this formulation?
  Yes, but (at least) we don’t want to write the constraints down manually.
So far, we have used “made-up” variable names like $x_{23}$. It is more useful to use **indexed** variables (and parameters).

The ZIMPL syntax is `VARIABLE[INDEX]`, but we first have to declare the indexed variables.

We first need an **index set**. Sets are defined like this in ZIMPL (**section 4.2 in the manual**):

- \[ 	ext{set } A := \{ 1, 2, 3 \}; \]

In the TSP model, we will certainly need the set $V$ of vertices:

- \[ 	ext{param } n := 6; \]
- \[ 	ext{set } V := \{ 1..n \}; \]

Additionally, we need the set $E$ of edges, which we represent by (ordered) pairs $(i,j)$ with $i < j$. Pairs or, more generally, vectors are called **tuples** in ZIMPL and have the notation $\langle i, j \rangle$ (angle brackets).

Using these index sets, we can declare indexed variables and parameters.

- \[ 	ext{var } x[E] \text{ binary;} \]

There is a special syntax for defining parameters, entry by entry.

See `tsp6–3.zpl`
The summation operator, to be used in objective functions or the left-hand or right-hand side of constraints. The general syntax is:

\[
\text{sum TUPLE-TEMPLATE in SET : EXPRESSION}
\]

This makes it possible to write down the expression for the objective function in a compact way:

\[
\text{minimize tour length:} \\
\text{sum \langle i,j \rangle in E : d[i,j] * x[i,j]; (Operator precedence: sum binds stronger than +, but weaker than \ast.)}
\]

The iteration statement, to be used in constraints:
The general syntax is:

\[
\text{forall TUPLE-TEMPLATE in SET do}
\]

This allows to generate multiple constraints at once:

\[
\text{subto degree:} \\
\text{forall <v> in V do} \\
\text{sum <v,j> in E : x[v,j] + sum <i,v> in E : x[i,v] == 2;}
\]
We next construct the set $E$ within ZIMPL using the with operator (section 4.2). General syntax:

$$\text{set NAME := \{ TUPLE-TEMPLATE in SET with CONDITION \}}$$

For the set $E$:

$$\text{set E := \{ <i,j> in (V cross V) with i < j \};}$$

Finally, we can define **indexed sets** (section 4.2):

$$\text{set S[] := powerset(V);}$$

This defines sets $S[1],\ldots,S[2^{|V|}]$ as all the subsets of $V$. An easy way to get the index set $1,\ldots,2^{|V|}$ that allows to access these sets is by using the **indexset** operator:

$$\text{set S_Indices := indexset(S);}$$

Now we can express the complete TSP formulation (**tsp6-5.zpl**).
Case Study: Line Drawings on Pen Plotters

Optimizing the operation of a pen plotter

Pen plotters are used instead of printers for very large-scale line drawings, such as for drawings in architecture, or charts of logic circuits in electronics. (Nowadays pen plotters are gradually being replaced by large-format inkjet printers.)

- The plotter can move a pen horizontally
- At the same time it can roll the paper (either a large sheet or paper from a roll) up and down
- These movements can be done in pen-up (not drawing) or pen-down (drawing) mode

Problem: Given a drawing to be produced, minimize the total drawing time.

Key questions:
- How is the drawing time determined?
- There are two parts of the total drawing time – one part is independent of our decisions, one does depend on our decisions.
- Can we draw every drawing in pen-down mode only?
- What are useful variables for modeling?
- What constraints do we need?
Case Study: The Shortest Path Problem in GPS Navigation Systems

The fundamental problem to be solved is to find the “shortest” path from A to B through the network of streets and roads.

Questions:

- How are distances defined?
- Mathematical abstraction of the network?
- Integer linear optimization model?