TSP: mathematical programming definition

- A set of n cities, with d_{ij} being the distance from city i to city j
- $x_{ij} = \begin{cases} 1 & \text{if city } j \text{ is reached from city } i \text{ (i.e., the edge } (i,j) \text{ is in the solution)} \\ 0 & \text{otherwise} \end{cases}$

$$\min \quad z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, \ldots n \qquad \text{(city } j \text{ is entered once and only once)}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, \ldots n \qquad \text{(city } i \text{ is exited once and only once)}$$

$$x_{ij} \in \{0, 1\}$$

$$\{ \textit{Solution forms an } n\textit{-city Hamiltonian tour} \}$$

- *Question:* how do we explicit the constraint $\rightarrow \{ \textit{Solution forms an } n\textit{-city Hamiltonian tour} \}$?

The *assignment* part of the formulation

- The first part of the formulation is equivalent to an *Assignment problem* (AP), which is solvable in polynomial time

$$
\begin{aligned}
\min \quad & z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \\
s.t. \quad & \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots n \\
& \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, \ldots n \\
& x_{ij} \in \{0, 1\}
\end{aligned}
$$

- *Example of an assignment problem*: *the best person for the job*! Given n workers with different skills and n jobs to be completed, a job that matches worker's skills costs less than one in which the operator is not as skillful for the characteristics of the job. Therefore, the problem consists in finding the assignment of each worker to a different job in order to define a set of pairs (worker, job) that minimize the total cost of executing the set of n jobs, where the cost of a pair (worker$_i$, job$_j$) is d_{ij}

		Jobs			
		1	**2**	...	**n**
Workers	**1**	d_{11}	d_{12}	...	d_{1n}
	2	d_{21}	d_{22}	...	d_{2n}
	⋮	⋮	⋮	⋮	⋮
	n	d_{n1}	d_{n2}	...	d_{nn}

- $x_{ij} = 1$ means that in the solution worker i has been assigned to job j, and the associated cost contribution is d_{ij}

- Constraint $\sum_{i=1}^{n} x_{ij} = 1,\ \forall j = 1, \ldots n$, guarantees that *each job is selected once and only once*

  Example: for $n = 4$ and job $j = 3$: $x_{13} + x_{23} + x_{33} + x_{43} = 1 \Longrightarrow$ since $x_{ij} \in \{0, 1\}$, one and only one of the $x_{i3}, i = 1, \ldots, 4$ must be set to 1 in the solution, all the others must be equal to 0

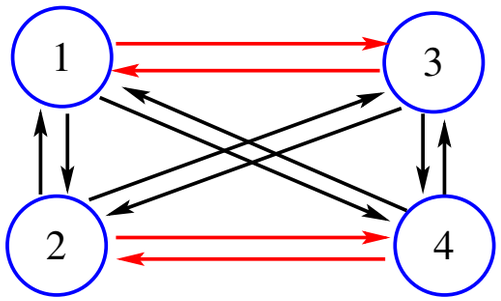
- Constraint $\sum_{j=1}^{n} x_{ij} = 1,\ \forall i = 1, \ldots n$, guarantees that *each worker is selected once and only once*

- *Examples of feasible solutions* ($n = 4$):

  - $x_{13} = x_{24} = x_{32} = x_{41} = 1$, and all other x_{ij} set to 0

  - $x_{13} = x_{24} = x_{31} = x_{42} = 1$, and all the other x_{ij} variables set to 0

    - AP-feasible but not TSP-feasible: *It contains the sub-tours (1-3-1) and (2-4-2)*



    - *The AP is a relaxation of a TSP*, since it consists of a TSP minus {Tour constraints}

- The basic AP is a special case of a minimum cost flow problem which is in turn a special case of a linear program. It is solvable in polynomial time with many specialized algorithms (e.g., the *Hungarian algorithm*) or using more general approaches such as the Simplex algorithm.

- *Hamiltonian tour constraints* can be explicited in various ways, one of the early and most used (since it's 'tight') formulation is the *DFJ formulation* due to Dantzig, Fulkerson, and Johnson (1954):

$$\sum_{i,j \in S}^{n} x_{ij} \leq |S| - 1, \quad \forall S \subset V, \quad 2 \leq |S| \leq n - 1$$

- *Example for a TSP with $n \geq 6$:* For a subset S of nodes with indices $\{2, 3, 6\}$, $|S| = 3$, the constraint equation writes as: $x_{32} + x_{26} + x_{63} + x_{23} + x_{62} + x_{36} \leq 2$. A tour among these three nodes would imply that, at least, $x_{23} = x_{36} = x_{62} = 1$, or $x_{32} = x_{26} = x_{63} = 1$. But in these cases, the sum would be at least 3. Forcing the sum to be at most 2 avoids the creation of sub-tours among the 3 nodes.

- The above constraint alone would not avoid sub-tours of 2 nodes among the nodes in $\{2, 3, 6\}$. However, this is dealt by the constraints where S is respectively equal to $\{2, 3\}, \{2, 6\}$, and $\{3, 6\}$

- With the DFJ formulation the number of constraints amounts to $O(2^n) \rightarrow$ e.g., 2^{100} = huge!

Heuristics for dealing with the size of the *DFJ formulation*

- $\rightarrow$ *Heuristic approach:* start with the AP formulation and then iteratively add subsets of DFJ constraints related to the variables causing the (smaller) sub-tours existing in the solution
- This alone will not necessarily generate the optimal solution, that can be generated according to the following procedure:
  - Once enough DFJ constraints have been added (i.e., the solution contains no sub-tours or a little number of them) add an exact tour constraint set which is more manageable than the DFJ one, and then solve to optimality with a branch-and-bound.

  - $\rightarrow$ The *MTZ, Miller, Tucker, Zemlin (1960), formulation* for tour constraints introduces substantially less constraint equations ($O(n^2)$) than the DFJ formulation and can be used for this purpose.

    - Starting directly with MTZ (without the heuristic based on the iterative addition of DFJ constraints) might be not effective. In fact, the MTZ formulation is not a tight one and it would require a large number of branch-and-bound node expansions to reach optimality. On the other hand, the DFJ formulation is tight, but too large in practice.

The *MTZ formulation*

- The idea of this formulation is to use extra variables $u_i, i = 1, \ldots, n$, to assign the indices $1l \ldots n$ to the nodes, so that the indexing corresponds to the order of the nodes in the tour. Clearly, such an indexing must be unique for each node in a tour (except for the initial one). In the case of a solution with a sub-tour (e.g., 1-5-6-2-4-3-6-1), node 6 is both in positions 3 and 7, such that the positional variable u_6 cannot be assigned consistently.

- The formulation assumes to fix at position 1 an initial node 1:

$$
\begin{aligned}
& u_1 = 1 \\
& 2 \leq u_i \leq n, && \forall i, j \neq 1 \\
& u_i - u_j + 1 \leq (n-1)(1 - x_{ij}), && \forall i, j \neq 1
\end{aligned}
$$

- Let's see through an example how a sub-tour would violate the constraints: If the sub-tour (x_{ij}, x_{jk}, x_{ki}), with $i, j, k \neq 1$, is in the solution, then, the right-hand side of the constraint inequality is zero. From the left-hand part:
$u_i \leq u_j - 1, \quad u_j \leq u_k - 1, \quad u_k \leq u_i - 1 \Longrightarrow u_i \leq u_k - 2$, that, put together with $u_k \leq u_i - 1$, would imply the infeasible inequality $u_i \leq u_i - 3$

- If $x_{ij} = 0$ (not in the solution), the constraint simply says that the positional distance between nodes i and j in the solution must be less than $n - 2$