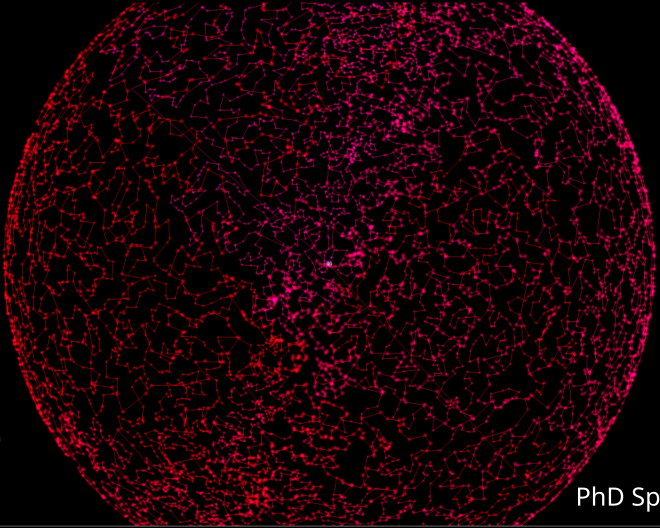
# Hardness of Metric TSP instances

## A computational study

E. Vercesi



UNIVERSITÀ DI PAVIA

Università della Svizzera italiana

PhD Spring Workshop 2022

# Outline

- Introduction: Metric Traveling Salesman Problem (TSP)
- Difficulty of a TSP instance: literature review
- Predicting hardness: preliminary results

**Main references:**


- V, E., Gualandi, S., Mastrolilli, M., & Gambardella, L. M. (2021). On the generation of Metric TSP instances with a large integrality gap by branch-and-cut. arXiv preprint *arXiv:2109.02454*. Under review at *Mathematical Programming Computation*.
- Gambardella, L. M., Gualandi, S., Mastrolilli, M., & V, E. (2022). Hardness of metric TSP instances: a computational study. To be submitted to *AIROSpringer*.

# INTRODUCTION

# Metric TSP

Given $n$ cities and a cost $c_{ij}$ to go from city $i$ to city $j$, for every city $i, j$, what is the shortest possible **tour** that visits each city exactly once and returns to the origin city?

# Metric TSP

Given $n$ cities and a cost $c_{ij}$ to go from city $i$ to city $j$, for every city $i, j$, what is the shortest possible **tour** that visits each city exactly once and returns to the origin city?

Graph $G$ with $n$ **nodes** (= cities) and $m$ **edges** (=links between cities)

# Metric TSP

Given $n$ cities and a cost $c_{ij}$ to go from city $i$ to city $j$, for every city $i, j$, what is the shortest possible **tour** that visits each city exactly once and returns to the origin city?

Graph $G$ with $n$ **nodes** (= cities) and $m$ **edges** (=links between cities)

Assumptions: metric properties of the cost vector:

- ✓ Completeness of the problem;
- ✓ No self-loop ($c_{ii} = 0$ for each $e = \{i, i\}, i \in V$)
- ✓ Symmetry ($c_{ij} = c_{ji}$ for each $e = \{i, j\}, i, j \in V$)
- ✓ Triangle inequalities ($c_{ij} \leq c_{ik} + c_{jk} \quad \forall 1 \leq i < j < k \leq n$)

$\rightarrow$ NP-hard problem! (Kannan and Monma, 1978)

# Is TSP really hard?

...It depends!

- It depends on the number of nodes of the TSP instance;
- Cover image of this presentation: Optimal tour through 119614 stars from the HYG Database ($\sim$ 130 days of computer time) [https://www.math.uwaterloo.ca/tsp/star/index.html]

# Is TSP really hard?

...It depends!

- It depends on the number of nodes of the TSP instance;
- Cover image of this presentation: Optimal tour through 119614 stars from the HYG Database ($\sim$ 130 days of computer time) [`https://www.math.uwaterloo.ca/tsp/star/index.html`]

Actually, what causes hardness it's still unclear...

# Is TSP really hard?

- It depends on the used solver. State-of-art solver: `concorde` (Applegate et al., 1998);
- We have observed that different implementation of algorithms for TSP lead to differences between runtimes on different instances.
  - Instance $A$ requires more time than instance $B$ to be solved using `concorde`
  - Instance $B$ requires more time than instance $A$ to be solved using a custom implementation of TSP solver

# Is TSP really hard?

- It depends on the used solver. State-of-art solver: `concorde` (Applegate et al., 1998);
- We have observed that different implementation of algorithms for TSP lead to differences between runtimes on different instances.
  - Instance *A* requires more time than instance *B* to be solved using `concorde`
  - Instance *B* requires more time than instance *A* to be solved using a custom implementation of TSP solver
- In the literature, we have both instances that are *easy* or *hard*;
- In the literature, there are instances with $30$ nodes that require $\sim 400$ seconds or $0.02$ seconds *in the same conditions*.

# Is TSP really hard?

- It depends on the used solver. State-of-art solver: `concorde` (Applegate et al., 1998);
- We have observed that different implementation of algorithms for TSP lead to differences between runtimes on different instances.
  - Instance *A* requires more time than instance *B* to be solved using `concorde`
  - Instance *B* requires more time than instance *A* to be solved using a custom implementation of TSP solver
- In the literature, we have both instances that are *easy* or *hard*;
- In the literature, there are instances with $30$ nodes that require $\sim 400$ seconds or $0.02$ seconds *in the same conditions*.

? What is the main cause of hardness?

# Predicting hardness: literature review

- Exact algorithms: Cheeseman et al. (1991), Gent and Walsh (1996), Osorio and Pinto (2003), Fischer et al. (2005), Schawe and Hartmann (2016).
- Heuristic methods: Hemert and Urquhart (2004), Smith-Miles et al. (2010), Cárdenas-Montes (2016).

# Predicting hardness: literature review

- Exact algorithms: Cheeseman et al. (1991), Gent and Walsh (1996), Osorio and Pinto (2003), Fischer et al. (2005), Schawe and Hartmann (2016).
- Heuristic methods: Hemert and Urquhart (2004), Smith-Miles et al. (2010), Cárdenas-Montes (2016).

**Main Limitations:**

- ✘ Predictors only work for particular metric TSP (see e.g Euclidean TSP, nodes = coordinates in $\mathbb{R}^k$ and distances computed with $||\cdot||_p$, $p = 1, 2$);
- ✘ Predictors that can only computed *a posteriori*;

# Our contribution

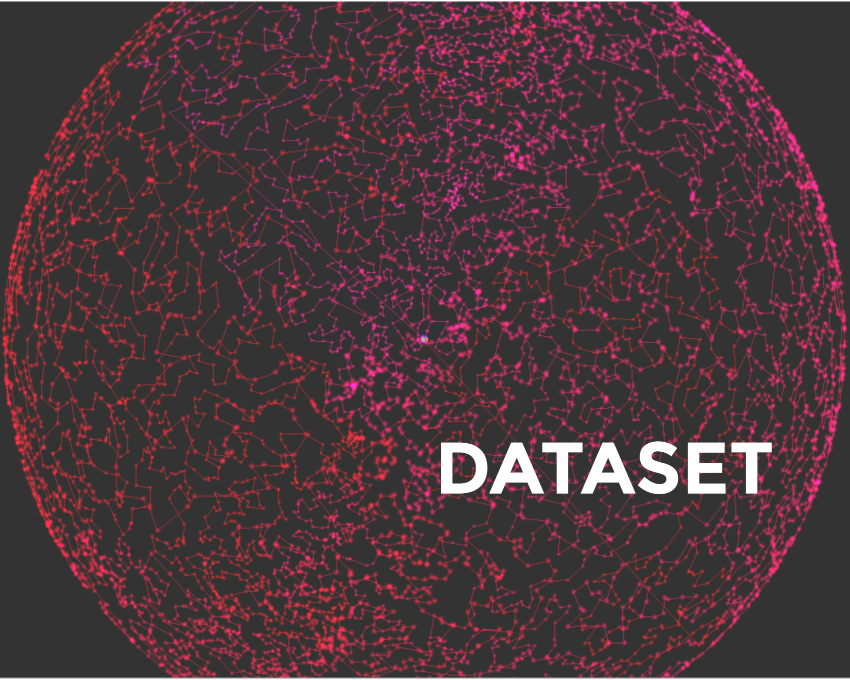
- ✓ Introduce easy-to-compute scores for predicting hardness of a metric TSP instance using only the cost vector $\boldsymbol{c}$;
- ✓ Compute such scores for $\sim 5500$ metric TSP instances;
- ✓ Train a Decision Tree (DT) aiming to predict hardness;
- ✓ Test the DT to predict hardness.

# Our contribution

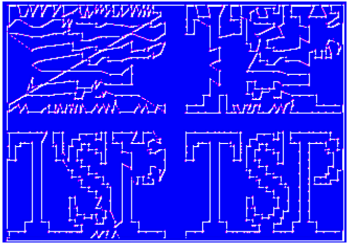
- ✓ Introduce easy-to-compute scores for predicting hardness of a metric TSP instance using only the cost vector $\boldsymbol{c}$;
- ✓ Compute such scores for $\sim 5500$ metric TSP instances;
- ✓ Train a Decision Tree (DT) aiming to predict hardness;
- ✓ Test the DT to predict hardness.
- Quick look at the instances that generate the datasets

# Our contribution

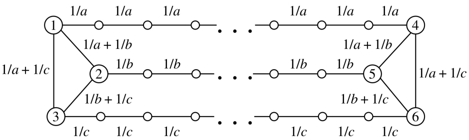
- ✓ Introduce easy-to-compute scores for predicting hardness of a metric TSP instance using only the cost vector $\boldsymbol{c}$;
- ✓ Compute such scores for $\sim 5500$ metric TSP instances;
- ✓ Train a Decision Tree (DT) aiming to predict hardness;
- ✓ Test the DT to predict hardness.
- Quick look at the instances that generate the datasets
- → More focused look at the scores.



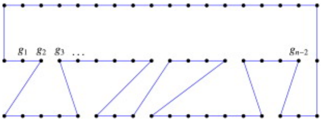
# DATASET



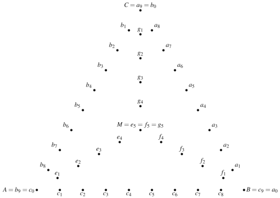
TSPLIB (1995)



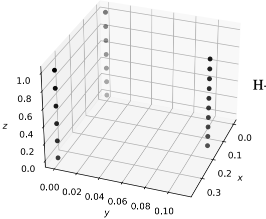

Benoit and Boyd (2008)



Hougardy (2014)



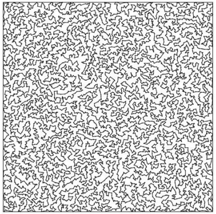
Hougardy and Zhong (2020)



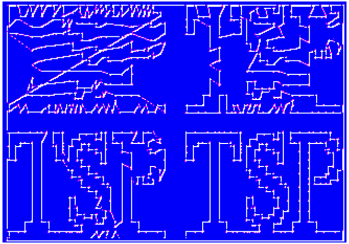

Zhong (2021)

$$
\begin{aligned}
\text{H-OPT}(\bar{\boldsymbol{x}}^{(h)}) := \min \quad & \sum_{\{i,j\}\in E} \bar{x}_{ij}^{(h)} c_{ij} \\
\text{s.t.} \quad & \sum_{\{i,j\}\in E} \bar{z}_{ij} c_{ij} \geq 1 && \forall \bar{\boldsymbol{z}} \in \mathcal{T}_n \\
& c_{ij} \leq c_{ik} + c_{jk} && \forall i,j,k \in V \\
& c_{ij} \geq 0 && \forall \{i,j\} \in E.
\end{aligned}
$$

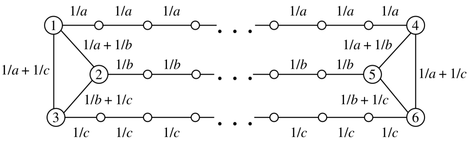
V. , Gualandi,
Mastrolilli, Gambardella (2021)



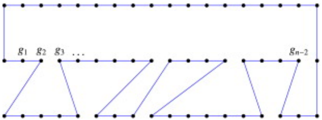
Random 2D - 3D nodes



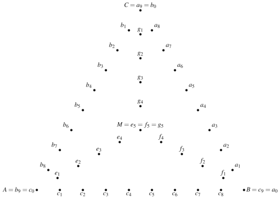
TSPLIB (1995)



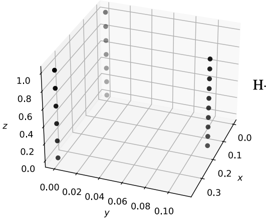

Benoit and Boyd (2008)




Hougardy (2014)



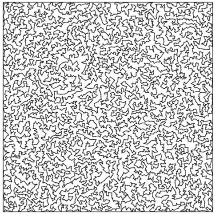

Hougardy and Zhong (2020)



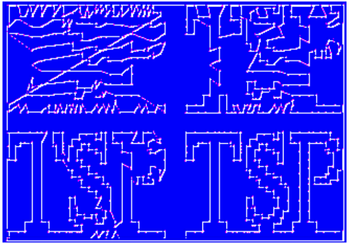

Zhong (2021)

$$
\begin{aligned}
\text{H-OPT}(\bar{\boldsymbol{x}}^{(h)}) := \min \quad & \sum_{\{i,j\}\in E} \bar{x}_{ij}^{(h)} c_{ij} & \\
\text{s.t.} \quad & \sum_{\{i,j\}\in E} \bar{z}_{ij} c_{ij} \geq 1 & \forall \bar{\boldsymbol{z}} \in \mathcal{T}_n \\
& c_{ij} \leq c_{ik} + c_{jk} & \forall i,j,k \in V \\
& c_{ij} \geq 0 & \forall \{i,j\} \in E.
\end{aligned}
$$

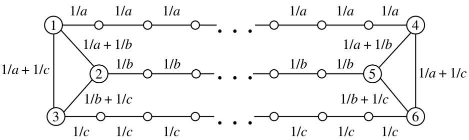
V. , Gualandi,
Mastrolilli, Gambardella (2021)



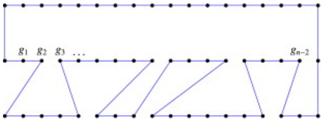
Random 2D - 3D nodes



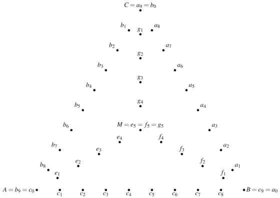
TSPLIB (1995)



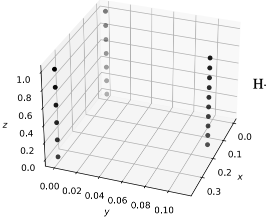

Benoit and Boyd (2008)



Hougardy (2014)



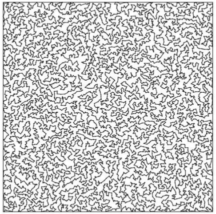

Hougardy and Zhong (2020)



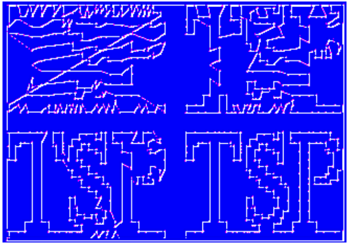

Zhong (2021)

$$
\begin{aligned}
\text{H-OPT}(\bar{\boldsymbol{x}}^{(h)}) := \min \quad & \sum_{\{i,j\}\in E} \bar{x}_{ij}^{(h)} c_{ij} \\
\text{s.t.} \quad & \sum_{\{i,j\}\in E} \bar{z}_{ij} c_{ij} \geq 1 && \forall \bar{\mathbf{z}} \in \mathscr{T}_n \\
& c_{ij} \leq c_{ik} + c_{jk} && \forall i,j,k \in V \\
& c_{ij} \geq 0 && \forall \{i,j\} \in E.
\end{aligned}
$$

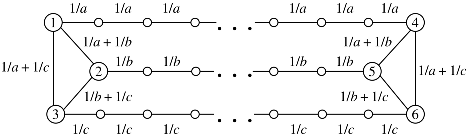
V. , Gualandi,
Mastrolilli, Gambardella (2021)



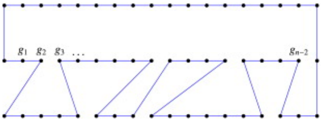
Random 2D - 3D nodes



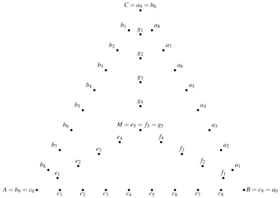
TSPLIB (1995)



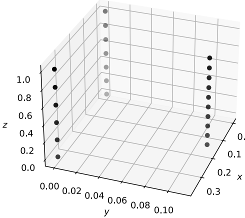

Benoit and Boyd (2008)



Hougardy (2014)



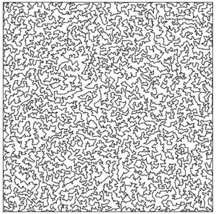

Hougardy and Zhong (2020)



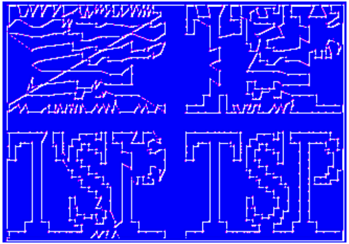

Zhong (2021)

$$
\begin{aligned}
\text{H-OPT}(\bar{\mathbf{x}}^{(h)}) := \min \quad & \sum_{\{i,j\}\in E} \bar{x}_{ij}^{(h)} c_{ij} \\
\text{s.t.} \quad & \sum_{\{i,j\}\in E} \bar{z}_{ij} c_{ij} \geq 1 && \forall \bar{\mathbf{z}} \in \mathcal{T}_n \\
& c_{ij} \leq c_{ik} + c_{jk} && \forall i,j,k \in V \\
& c_{ij} \geq 0 && \forall \{i,j\} \in E.
\end{aligned}
$$

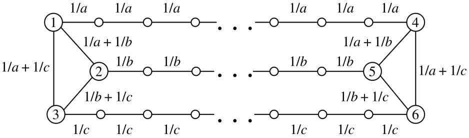
V. , Gualandi,
Mastrolilli, Gambardella (2021)



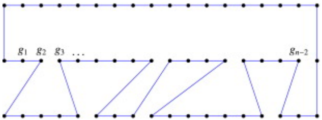
Random 2D - 3D nodes



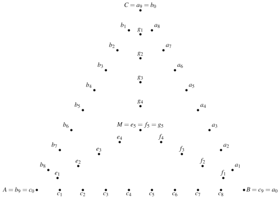
TSPLIB (1995)



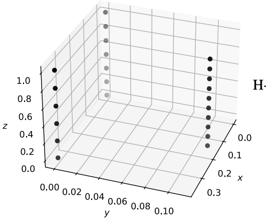

Benoit and Boyd (2008)



Hougardy (2014)

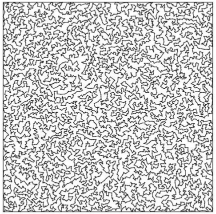



Hougardy and Zhong (2020)




Zhong (2021)

$$
\begin{aligned}
\text{H-OPT}(\bar{\boldsymbol{x}}^{(h)}) := \min \quad & \sum_{\{i,j\}\in E} \bar{x}_{ij}^{(h)} c_{ij} & \\
\text{s.t.} \quad & \sum_{\{i,j\}\in E} \bar{z}_{ij} c_{ij} \geq 1 & \forall \bar{\mathbf{z}} \in \mathscr{T}_n \\
& c_{ij} \leq c_{ik} + c_{jk} & \forall i,j,k \in V \\
& c_{ij} \geq 0 & \forall \{i,j\} \in E.
\end{aligned}
$$

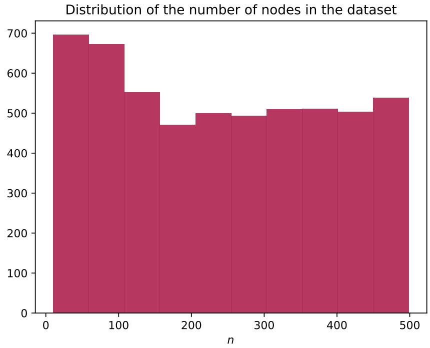


V. , Gualandi,
Mastrolilli, Gambardella (2021)

Random 2D - 3D nodes

# Some statistics on the dataset

- $5446$ Metric TSP instances: $5174$ hard, $272$ easy;
- $1$ hard instance every $19$ easy ones $\Leftarrow$ hard instances are outliers in the metric TSP space;
- From $10$ to $499$ nodes.



# FEATURES

# Normalized Standard Deviation (std)

- Modification of a score proposed by Cheeseman et al. (1991), that relies on the *standard deviation of the cost matrix*
- Author only considered Euclidean instances on $[0,1]^2$
- Author evaluated different instance with a fixed number of nodes
- Author found that harder instances are such that

$$s_1(n) \le std(\boldsymbol{c}) \le s_2(n)$$

We compute the standard deviation on the *normalized* matrix

$$\text{std}(\boldsymbol{c}) = \sqrt{\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\bar{c}_{ij} - \bar{c})^2}, \quad \bar{c}_{ij} = \frac{c_{ij}}{\max(\boldsymbol{c})}, \quad \bar{c} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\bar{c}_{ij}$$

# Tight Triangle Inequality (TTI) index

- In metric TSP, it holds that $c_{ij} \leq c_{ik} + c_{jk}$ for all $i, j, k \in \{1, \ldots, n\}$
- For every triple $(i, j, k)$, it can be proven that at most one of the following holds:

$$c_{ij} = c_{ik} + c_{jk} \quad c_{ik} = c_{ij} + c_{jk} \quad c_{jk} = c_{ik} + c_{ij}$$

If for a lot of triples it holds one of the above, different strategies may be adopted for finding the optimal tour. We define a new score

$$\text{TTI}(\boldsymbol{c}) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} \mathbf{1}_{c_{ij}=c_{ik}+c_{jk}} + \mathbf{1}_{c_{ik}=c_{ij}+c_{jk}} + \mathbf{1}_{c_{jk}=c_{ij}+c_{ik}}}{m(n-2)}$$

# Skewness (sk)

- Plot histograms of $\boldsymbol{c}$ for various instances
- We observe that hard instances have a *right-skewed distribution*;
- Costs in easy instances are barely uniformly distributed
- Computation of the *skewness* as a predictor of hardness:

$$\mathrm{sk}(\boldsymbol{c}) = \frac{\frac{1}{m}\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\overline{c}_{ij} - \overline{c})^3}{\left[\frac{1}{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\overline{c}_{ij} - \overline{c})^2\right]^{3/2}} \quad \overline{c} \text{ sample mean}$$

**Note:** Skewness has been also considered in Correa et al. (2015):

- In combination with other scores
- With such scores it wasn't so effective

# DECISION TREE

# Training phase

- ✓ Dataset - generic row

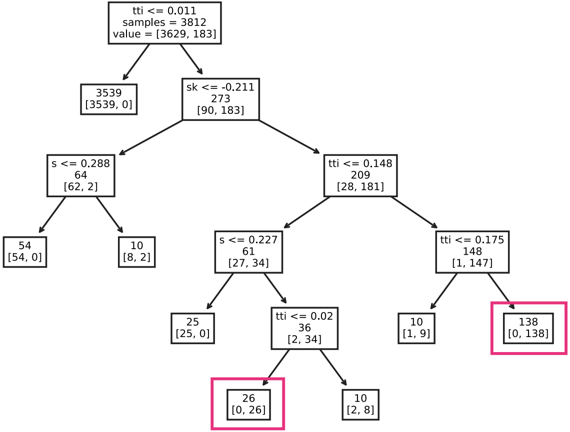
  `instance_name, std(c), TTI(c), sk(c), hard`

- ✓ We train a decision tree 100 times with random training and validation set (using `sckit-learn` (Pedregosa et al., 2011))
- ✓ 66% of data for the training set, 33% for the validation
- ✓ Choose the best tree according to F1-score on the validation set

$$F1 = \frac{\text{TH}}{\text{TH} + \frac{1}{2}(\text{FH} + \text{FE})}$$

- ✓ Best F1-score on the training set: 99.86%
- ✓ Best F1-score on the validation set: 97.63%

# The Decision Tree




- ✓ Positive `sk(c)` plus high `TTI(c)` imply hard instances
- ✓ Positive `sk(c)` plus low `TTI(c)` require high `std(c)` to be classified as hard

# Unseen data - Easy

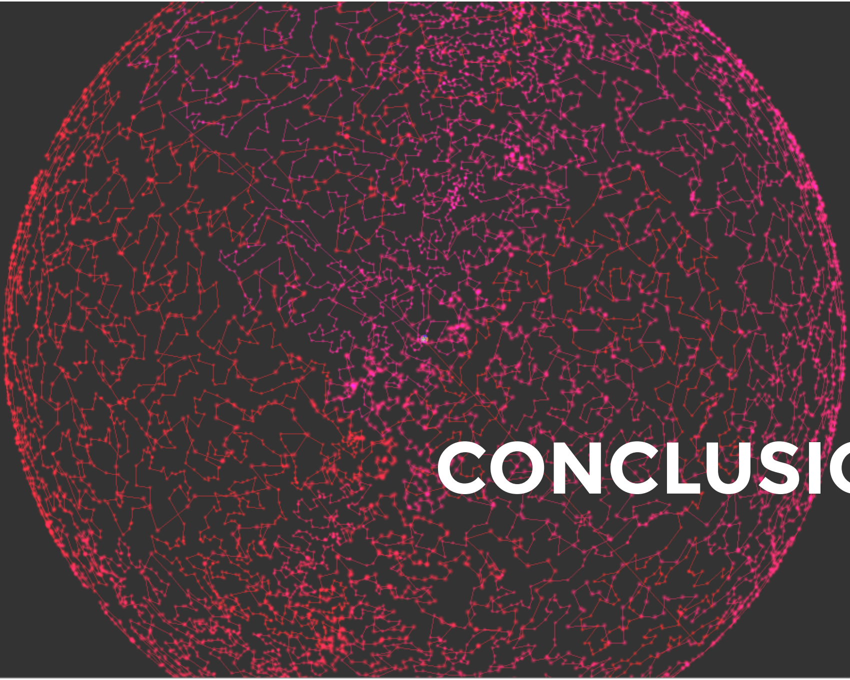
| Instance | Time | Hard? | Pred Hard? |
|---|---|---|---|
| att532 | 7.01 | No | No |
| ali535 | 2.23 | No | No |
| si535 | 3.79 | No | No |
| pa561 | 25.85 | No | No |
| p654 | 1.90 | No | No |
| rat575 | 23.17 | No | No |
| gr666 | 9.32 | No | No |
| si1032 | 2.99 | No | No |
| rand_10.tsp | 0.04 | No | No |
| rand_40.tsp | 0.01 | No | Yes |
| rand_11.tsp | 0.03 | No | Yes |
| rand_25.tsp | 0.02 | No | No |
| rand_28.tsp | 0.01 | No | Yes |
| rand_35.tsp | 0.10 | No | No |
| rand_16.tsp | 0.02 | No | Yes |

- ✓ Metric TSPLIB instances not used in the training set;
- ✓ Random instances.

# Unseen data - Hard

| $n$ | Time | Hard? | Pred Hard? |
|---|---|---|---|
| 15 | 2.631 | Yes | Yes |
| 15 | 3.081 | Yes | Yes |
| 15 | 2.703 | Yes | Yes |
| 20 | 17.948 | Yes | Yes |
| 20 | 9.797 | Yes | Yes |
| 20 | 6.768 | Yes | Yes |
| 25 | 84.877 | Yes | Yes |
| 25 | 101.340 | Yes | Yes |
| 25 | 95.112 | Yes | Yes |
| 30 | 156.553 | Yes | Yes |
| 30 | 124.638 | Yes | Yes |
| 30 | 95.964 | Yes | Yes |
| 35 | 524.868 | Yes | Yes |
| 35 | 133.196 | Yes | Yes |
| 35 | 889.248 | Yes | Yes |

- ✓ Use the instance generator in In V, Gualandi, Mastrolilli, Gambardella.
- ✓ Generate 3 instances for each $n \in \{15, 20, 25, 30, 35\}$.



# CONCLUSION

# Conclusion and future perspectives

## Conclusions

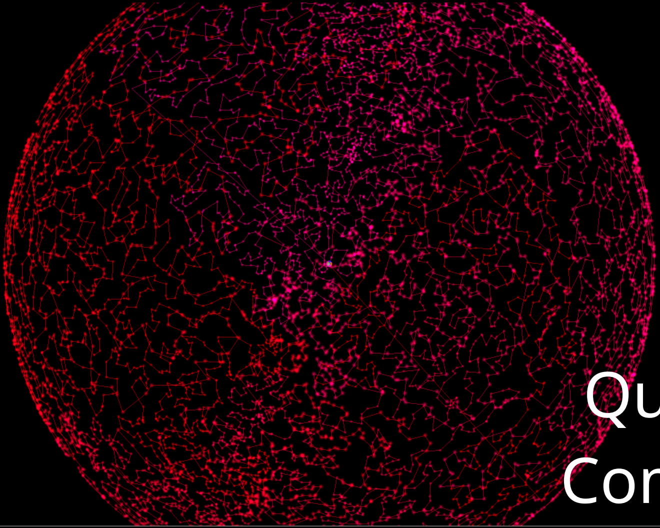
- ✓ Although TSP is an NP-hard problem, not all the instances are actually hard to solve for the state of art solver (concorde)
- ✓ There is computational evidence that some scores, computed from cost vector, are capable of partially predicting the hardness of an instance before actually solving it.

## Future perspectives

- Scores yet available in the literature may be adapted and generalized;
- Scores can be used to train a *generative model*
- Scores can be used as a baseline for the study new cuts for the B&C method for the TSP.

# Thank you for your attention

*This work was made possible thanks to the fruitful collaboration between UNIPV and USI*



Questions?

Comments?

# Metric TSP

Given $n$ cities and a cost $c_{ij}$ to go from city $i$ to city $j$, for every city $i, j$, what is the shortest possible route that visits each city exactly once and returns to the origin city?

This problem can be modelled with a weighted complete graph, $K_n = (V, E)$

Metric property of the costs:

$$
\begin{aligned}
c_{i,i} &= 0 && \forall i \in V \\
c_{i,j} &= c_{j,i} && \forall i, j \in V \\
c_{i,j} &\leq c_{i,k} + c_{j,k} && \forall i, j, k \in V
\end{aligned}
$$

Undirected edges $\rightarrow$ set $e = \{i, j\}$

Number of edges $m \; := \; \frac{n(n-1)}{2}$

# Mathematical Model

$\boldsymbol{x} \in \mathbb{R}^m$, edge incidence vector of $0$ and $1$ that represents a tour

$x_e = 0$ if the edge is picked in the tour, 1 otherwise $1 \leq e \leq m$

$\boldsymbol{c} \in \mathbb{R}^m$, vector of costs that satisfies the metric properties

A set $\mathcal{T}$ of all the possible incidence tours, $\boldsymbol{x}$ vectors.

Solve

$$\min_{x \in \mathcal{T}} \boldsymbol{c}^T \cdot \boldsymbol{x}$$

# Mathematical Model

$\boldsymbol{x} \in \mathbb{R}^m$, edge incidence vector of $0$ and $1$ that represents a tour

$x_e = 0$ if the edge is picked in the tour, 1 otherwise $1 \leq e \leq m$

$\boldsymbol{c} \in \mathbb{R}^m$, vector of costs that satisfies the metric properties

A set $\mathcal{T}$ of all the possible incidence tours, $\boldsymbol{x}$ vectors.

Solve

$$\min_{x \in \mathcal{T}} \boldsymbol{c}^T \cdot \boldsymbol{x}$$

Problem:

$$|\mathcal{T}| = \frac{(n-1)!}{2}$$

# Mathematical Model: Integer Linear Programming (ILP)

Most used mathematical model: Dantzig et al. (1954)

$$\Sigma := \{S \subset V : 3 \leq |S| \leq n - 3\}$$

$$S \subset V,\ \delta(S) := \{\{i,j\} : i \in S, j \notin S\}$$

$$\min \mathbf{c}^T \cdot \mathbf{x}$$

$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2 \quad \forall v \in V \quad \text{Degree constraints}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \in \Sigma \quad \text{Subtour elimination constraints}$$

$$x_e \in \{0, 1\} \quad \forall e \in E$$

Best known solving procedure for integer programs: **branch-and-cut** (Padberg and Rinaldi, 1991), efficiently implemented in `concorde` (Applegate et al., 1998)

# Mathematical Model: Subtour Elimination Problem (SEP)

$\Sigma := \{S \subset V : 3 \leq |S| \leq n - 3\}$
$S \subset V$, $\delta(S) := \{\{i, j\} : i \in S, j \notin S\}$

$$\min \boldsymbol{c}^T \cdot \boldsymbol{x}$$

$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2 \quad \forall v \in V \quad \text{Degree constraints}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \in \Sigma \quad \text{Subtour elimination constraints}$$

$$0 \leq x_e \leq 1 \quad \forall e \in E$$

The optimal value is a lower bound for the ILP optimal value, and the starting point of the branch and cut algorithm.
LP problem can be efficiently solved.

# Percentage of Equal Edges (PEE) index

- In Vercesi et al. (2021) it is conjectured that the number of edges with the same cost influences the difficulty of an instance
- The more regular the structure is, the harder the instance
- We try to encode this concept in a score called *Percentage of Equal Edges*-index
- Percentage of edges equal to the most frequent edge
- Let $c^*$ be the most frequent cost

$$PPE(\boldsymbol{c}) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbf{1}_{c_{ij}=c^*}}{m}$$

# Integer Programming Formulation

- One variable $x_e \in \{0, 1\}$ for each edge $e$ (See, e.g Dantzig et al. (1954))
- Thus, one $\boldsymbol{x} \in \mathbb{R}^m$ for each tour

$$\boldsymbol{x} = \begin{cases} x_e = 0 & e \text{ is not in the tour} \\ x_e = 1 & e \text{ is tour} \end{cases}$$

- Search space

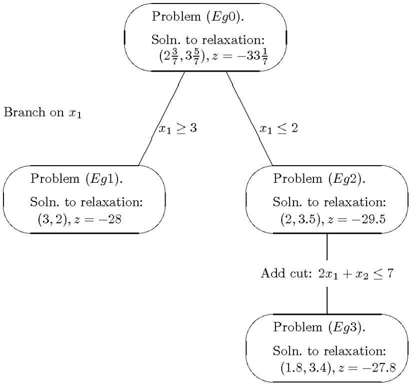
$$P := \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\} \cap \{0, 1\}^m$$

  for a suitable $\boldsymbol{A}$ matrix.
- Problem:

$$\begin{aligned} \min\ & \boldsymbol{c}^T\boldsymbol{x} \\ & \boldsymbol{x} \in P. \end{aligned}$$

- **NP-Hard problem!** (Kannan and Monma (1978))

# How to solve an ILP? Branch-and-Cut (B&C)




Picture from Mitchell (1988)

- ✓ Start by solving one *relaxation* of the given problem → enlarge the search space.
- ✓ Relaxation that provides a *lower bound* (minimization framework)
- ✓ Solve a tree of easier sub-problems (with possibly non integer solution)
- ✓ Add extra-constraints to the sub-problems (*cuts*)
- ✓ Prune the leaves of the tree according to some deductions on the value of the problem /solution
- ✓ Stop when you find an integer optimal solution

# Instance generator in Vercesi et al. (2021)

- Let $\boldsymbol{c}_0$ a TSP instance
- Let $\boldsymbol{x}_0$ a solution of the Subtour Elimination Problem (Linear relaxation)
- We solve

$$
\begin{aligned}
\text{H-OPT}(\bar{\boldsymbol{x}}^{(h)}) := \min \quad & \sum_{\{i,j\} \in E} \bar{x}_{ij}^{(h)} c_{ij} && && (1) \\
\text{s.t.} \quad & \sum_{\{i,j\} \in E} \bar{z}_{ij} c_{ij} \geq 1 && \forall \bar{\boldsymbol{z}} \in \mathcal{T}_n && (2) \\
& c_{ij} \leq c_{ik} + c_{jk} && \forall i, j, k \in V && (3) \\
& c_{ij} \geq 0 && \forall \{i,j\} \in E. && (4)
\end{aligned}
$$

# Instance generator in Vercesi et al. (2021)

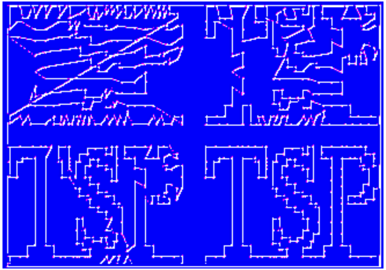
Obtaining $\boldsymbol{c}$ as a solution

It is possible to prove that

$$IG_{\boldsymbol{c}_0} \leq IG_{\boldsymbol{c}}$$

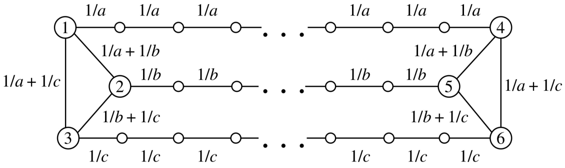
We also have computational evidence that the instances obtained in such ways are hard to solve

# TSPLIB



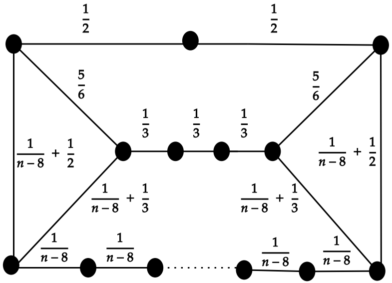
- State-of-art library for symmetric TSP
- Metric and non metric, picked only the metric ones.
- **Easy!**

# Benoit and Boyd (2008)



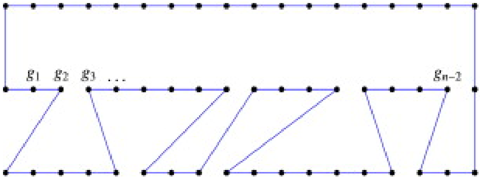

- Metric instances;
- Defined by 3 fixed parameters $(a, b, c)$, related with both the weight and the number of nodes
- Picked instances with $n \leq 100$
- **Easy!**

# $U_3$



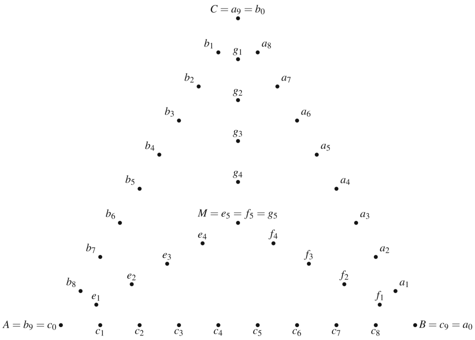

- Metric instances;
- Only depends to the number of nodes;
- Picked instances with $n \leq 100$
- **Hard!**

# Hougardy (2014)



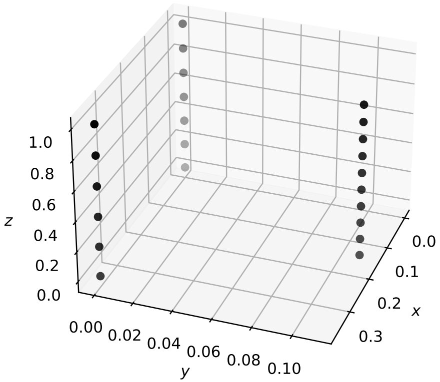

- Euclidean instances in $\mathbb{R}^2$
- One instance for each $n$ number of nodes.
- Picked instances with $n \leq 100$
- **Easy!**

# Hougardy and Zhong (2020)




- Euclidean instances in $\mathbb{R}^2$
- Depends on 2 parameters $(m, n)$, related with the number of nodes on each edge
- Picked instances with $n \leq 200$
- **Hard!**

# Zhong (2021)



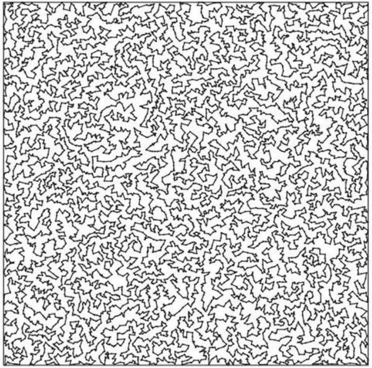

- Rectilinear instances in $\mathbb{R}^3$
- Depends on 3 parameters $(i, j, k)$, related with the number of nodes on each edge
- Picked instances with $n \leq 100$
- **Hard!**

# V, Gualandi, Mastrolilli, Gambardella

$$
\begin{aligned}
\text{H-OPT}(\bar{\boldsymbol{x}}^{(h)}) := \min \quad & \sum_{\{i,j\}\in E} \bar{x}_{ij}^{(h)} c_{ij} & \\
\text{s.t.} \quad & \sum_{\{i,j\}\in E} \bar{z}_{ij} c_{ij} \geq 1 & \forall \bar{\mathbf{z}} \in \mathscr{T}_n \\
& c_{ij} \leq c_{ik} + c_{jk} & \forall i,j,k \in V \\
& c_{ij} \geq 0 & \forall \{i,j\} \in E.
\end{aligned}
$$

- Not a family, but a *generator* of metric instances;
- Published 41 hard-to-solve instances
- All the instances have $n \leq 79$
- **Hard!**

# Random Euclidean / Rectilinear 2D - 3D instances



- Random generate $10 \leq n \leq 500$
- Random generate $p \in \{1, 2\}$
- Random generate $k \in \{2, 3\}$
- Random generate $n$ vectors in $\mathbb{R}^k$
- Compute costs using the $L^p$ norm
- **Easy!**
- Contribute to the dataset with 5000 instances