CP 2024 GIRONA

Learning Lagrangian Multipliers for the Travelling Salesman Problem

Augustin Parjadis, Quentin Cappart,
Bistra Dilkina, Aaron Ferber, Louis-Martin Rousseau

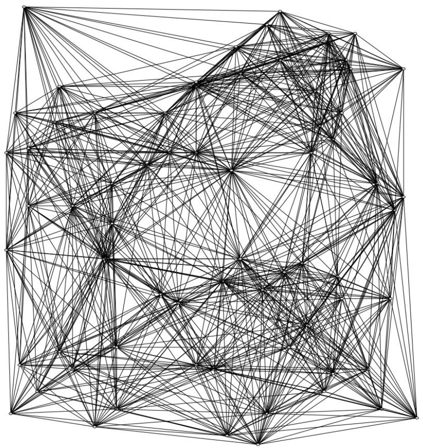
CANADA RESEARCH CHAIR IN HEALTHCARE ANALYTICS AND LOGISTICS HANALOG

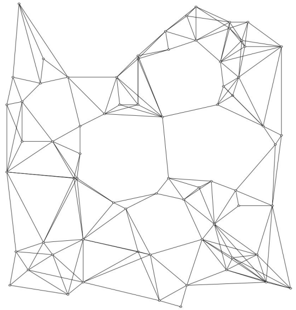
USC University of Southern California

POLYTECHNIQUE MONTRÉAL

Introduction

Improved Filtering for Weighted Circuit Constraints, (Benchimol P, van Hoeve WJ, Régin J-C, Rousseau L-M, Rueher R, 2012, *Constraints* 17:3, 205-233.)





- **WeightedCircuit constraint**: 1-tree relaxation for domain filtering and application to a Branch-and-Bound solver
- **Objective**: improving the 1-tree relaxation for the BnB solver

Plan

1. 1-tree and Held-Karp Relaxation
2. Proposed Approach
3. Application and Results

Plan

1. 1-tree and Held-Karp Relaxation
2. Proposed Approach
3. Application and Results

Introduction: Travelling Salesman Problem (TSP)

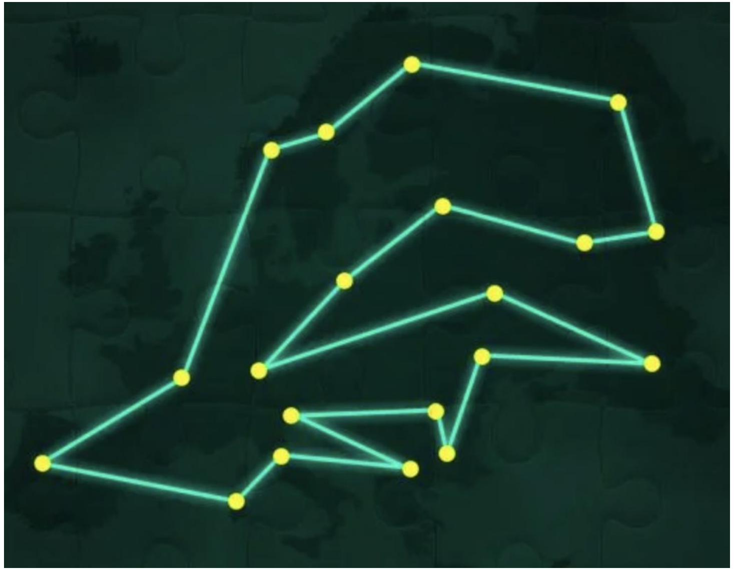
$$\min \sum_{e \in E} c_e x_e$$

$$\text{s. t.} \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V$$

$$\sum_{e \in E} x_e = |V|$$

$$\sum_{e \in S} x_e \leq |S| - 1, \ \forall S \text{ subtour of } G$$

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



Introduction: Travelling Salesman Problem (TSP)

$$\min \sum_{e \in E} c_e x_e$$

$$\text{s. t.} \quad \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V$$

$$\sum_{e \in E} x_e = |V|$$

$$\sum_{e \in S} x_e \leq |S| - 1, \ \forall S \text{ subtour of } G$$

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

1. Minimize total tour cost

Introduction: Travelling Salesman Problem (TSP)

$$\min \sum_{e \in E} c_e x_e$$

$$\text{s. t.} \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V$$

$$\sum_{e \in E} x_e = |V|$$

$$\sum_{e \in S} x_e \leq |S| - 1, \ \forall S \text{ subtour of } G$$

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

1. Minimize total tour cost
2. Each node is connected by 2 edges

Introduction: Travelling Salesman Problem (TSP)

$$\min \sum_{e \in E} c_e x_e$$

$$\text{s. t.} \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V$$

$$\sum_{e \in E} x_e = |V|$$

$$\sum_{e \in S} x_e \leq |S| - 1, \ \forall S \text{ subtour of } G$$

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

1. Minimize total tour cost
2. Each node is connected by 2 edges
3. The tour has the right length

Introduction: Travelling Salesman Problem (TSP)

$$\min \sum_{e \in E} c_e x_e$$

$$\text{s. t.} \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V$$

$$\sum_{e \in E} x_e = |V|$$

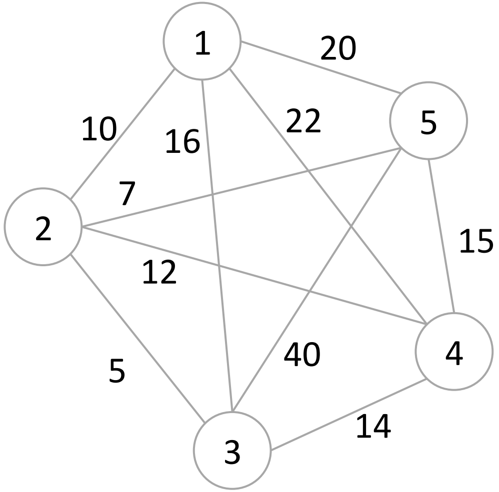
$$\sum_{e \in S} x_e \leq |S| - 1, \ \forall S \text{ subtour of } G$$

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

1. Minimize total tour cost
2. Each node is connected by 2 edges
3. The tour has the right length
4. No disconnected subtour is created

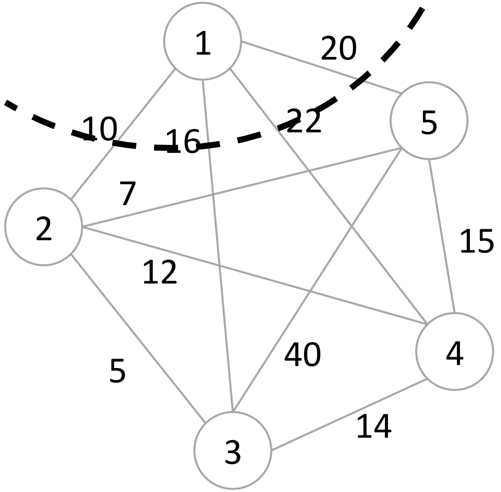
1-tree Relaxation

- A minimum *1-tree* T is a minimum spanning tree on G\{1}, to which {1} is connected back to T with 2 cheapest edges



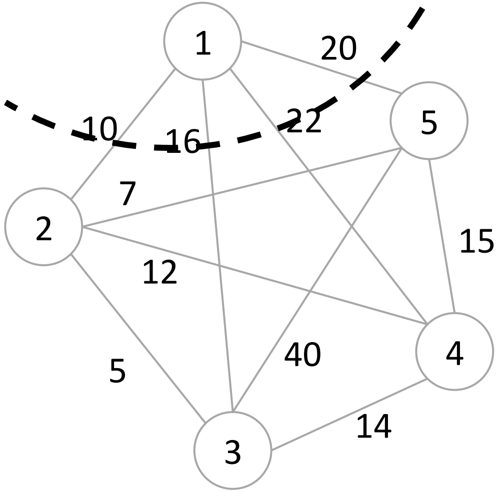
1-tree Relaxation

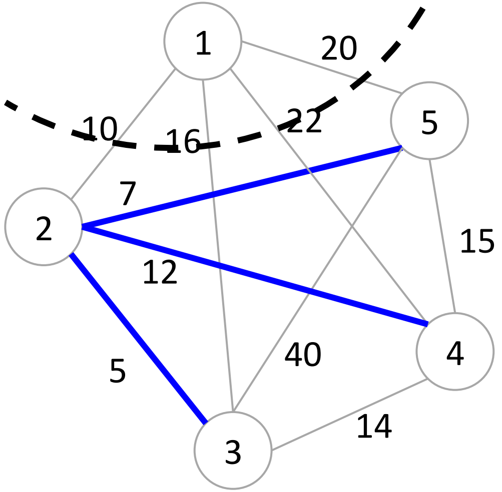
- A minimum *1-tree* T is a minimum spanning tree on G\{1}, to which {1} is connected back to T with 2 cheapest edges



1-tree Relaxation

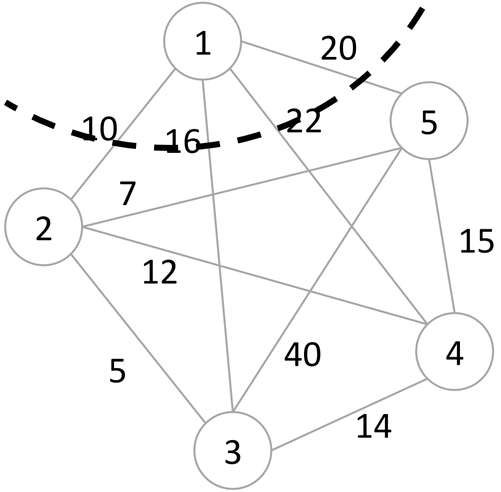
- A minimum *1-tree* T is a minimum spanning tree on G\{1}, to which {1} is connected back to T with 2 cheapest edges

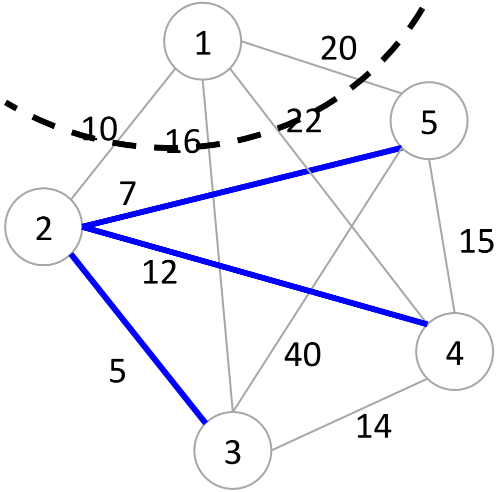


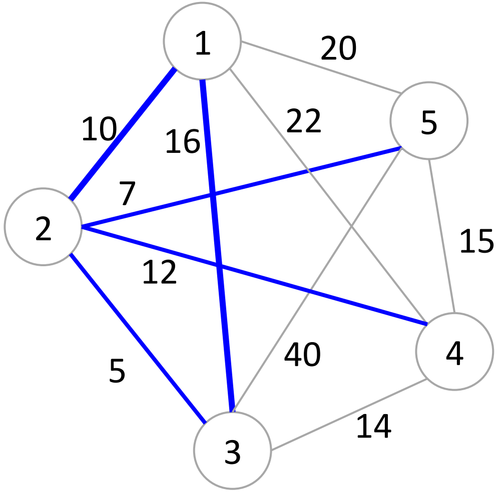


1-tree Relaxation

- A minimum *1-tree* T is a minimum spanning tree on G\{1}, to which {1} is connected back to T with 2 cheapest edges







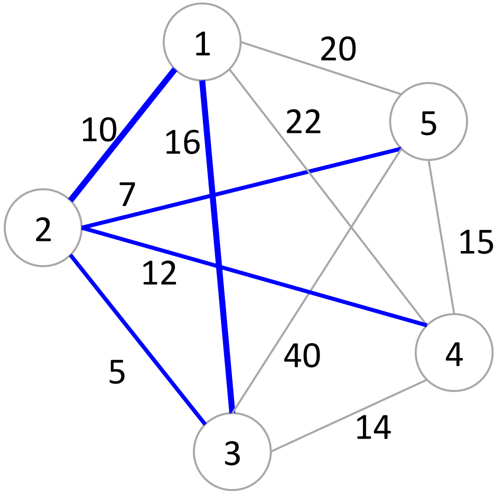
1-tree Relaxation

- A 1-tree is effectively the result of the **relaxation of the degree constraints** in the TSP model

$$
\begin{aligned}
\min \; & \sum_{e \in E} c_e x_e \\
\text{s. t.} \; & \cancel{\sum_{e \in \delta(v)} x_e = 2, \; \forall v \in V} \\
& \sum_{e \in E} x_e = |V| \\
& \sum_{e \in S} x_e \le |S| - 1, \; \forall S \text{ subtour of } G \\
& x_e \in \{0, 1\}, \; \forall e \in E
\end{aligned}
$$

1-tree Relaxation

- A tour is a 1-tree, and finding a minimum 1-tree is fast: **dual bound generation**



Bound: **50**

Held-Karp Relaxation

- The 1-tree can be tightened by Lagrangean relaxation, where **λ** penalizes the node degree violation

$$\begin{aligned} \min \sum_{e \in E} c_e x_e & \\ \text{s. t. } \cancel{\sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V} & \\ \sum_{e \in E} x_e = |V| & \\ \sum_{e \in S} x_e \leq |S| - 1, \ \forall S \text{ subtour of } G & \\ x_e \in \{0,1\}, \ \forall e \in E & \end{aligned}$$

$$\begin{aligned} \min \sum_{e \in E} c_e x_e + \sum_{i \in V} \lambda_i \Big(\sum_{e \in \delta(i)} x_e - 2 \Big) & \\ \sum_{e \in E} x_e = |V| & \\ \sum_{e \in S} x_e \leq |S| - 1, \ \forall S \text{ subtour of } G & \\ x_e \in \{0,1\}, \ \forall e \in E & \end{aligned}$$

Held-Karp Relaxation

- This is equivalent to modifying the edge costs

$$\min \sum_{e\in E} c_e x_e + \sum_{i\in V} \lambda_i \Big(\sum_{e\in\delta(i)} x_e - 2 \Big) \longrightarrow \min \sum_{e\in E} \underline{(c_e + \lambda_{e_1} + \lambda_{e_2})} x_e$$

$$\sum_{e\in E} x_e = |V| \qquad \sum_{e\in E} x_e = |V|$$

$$\sum_{e\in S} x_e \leq |S| - 1,\ \forall S \text{ subtour of } G \qquad \sum_{e\in S} x_e \leq |S| - 1,\ \forall S \text{ subtour of } G$$

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

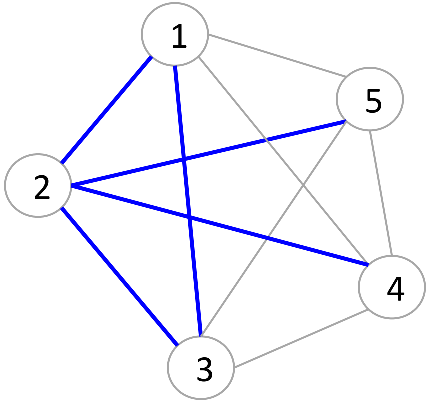
CHAIRE DE RECHERCHE DU CANADA EN ANALYTIQUE ET LOGISTIQUE DE SOINS DE SANTÉ
HANALOG

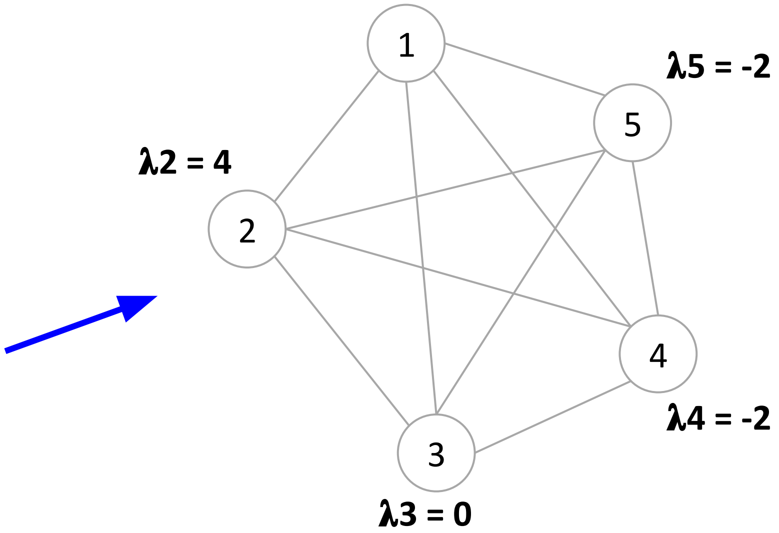
POLYTECHNIQUE MONTRÉAL

Held-Karp Relaxation

- **If T is a tour,** then LB = TSP value, else
- **Iterative factors update** for bound tightening:
  a. positive for degrees > 2 to discourage selected edges,
  b. negative for degrees = 1,
  c. 0 for degrees = 2

Bound: **50**

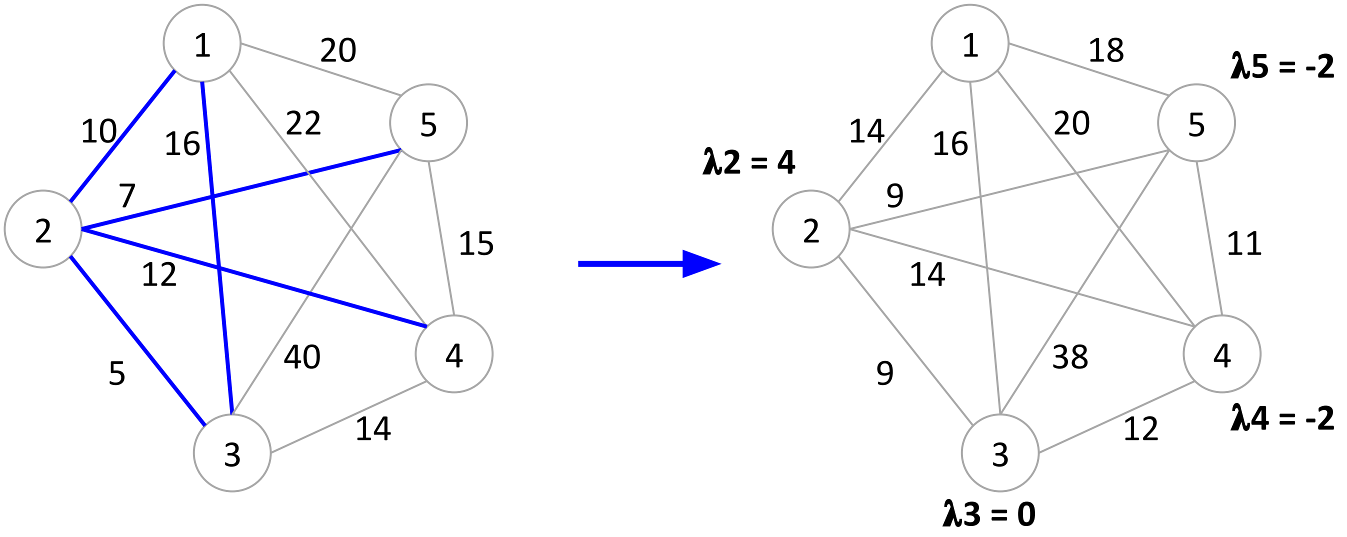




Held-Karp Relaxation

- Adding factors on the nodes modifies the minimum 1-tree but not the optimal tour

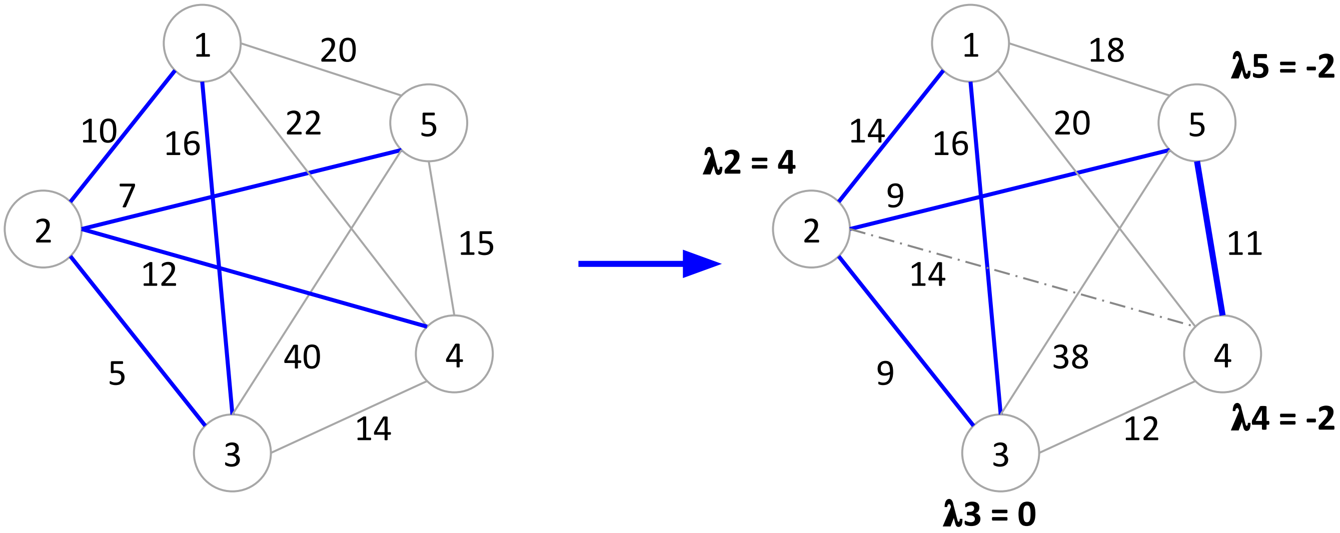
Bound: **50**



Held-Karp Relaxation

- Adding factors on the nodes modifies the minimum 1-tree but not the optimal tour

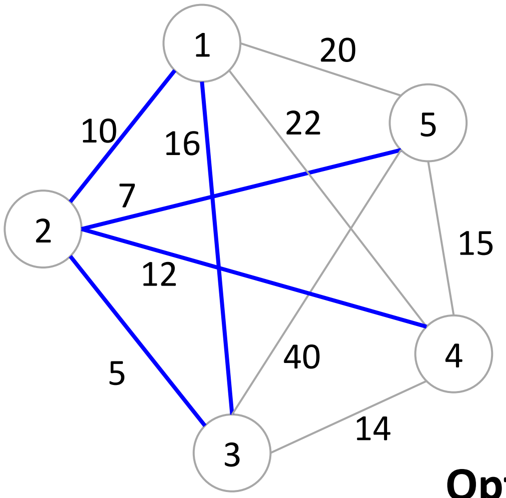
Bound: **50**

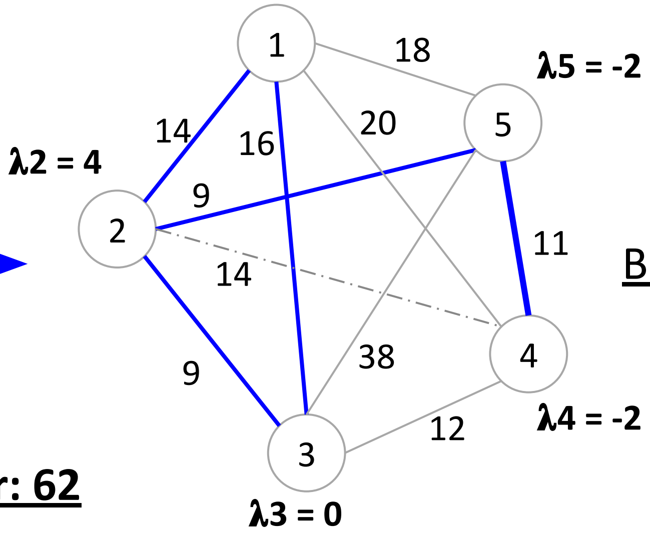


Held-Karp Relaxation

- Adding factors on the nodes modifies the minimum 1-tree but not the optimal tour

Bound: **50**



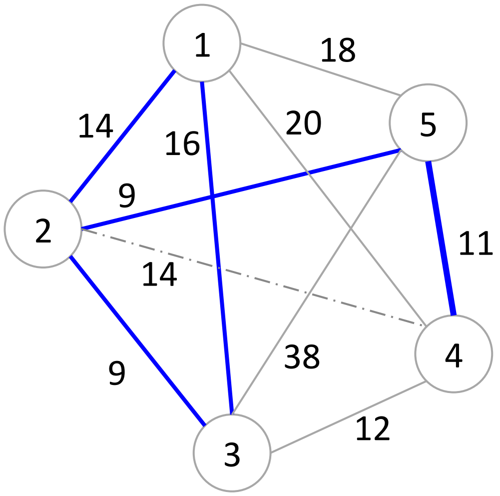


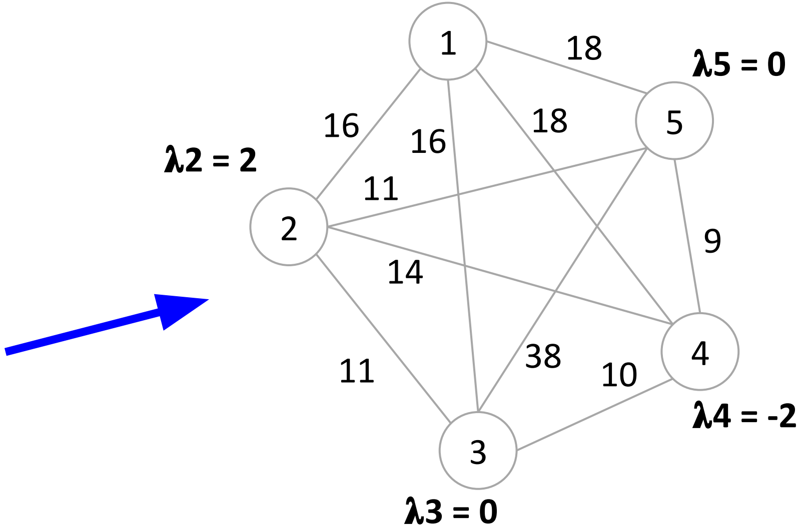
Bound: 59

Optimal tour: 62

Held-Karp Relaxation

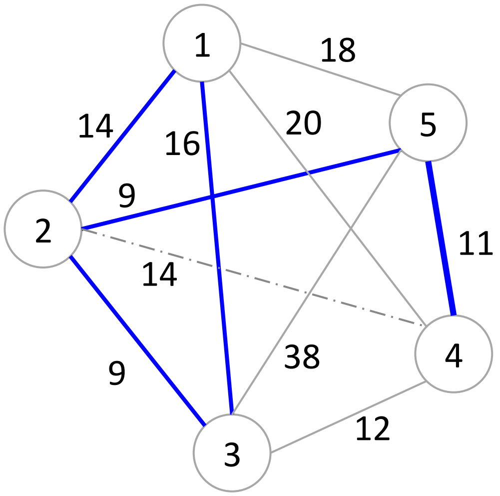
Bound: **59**

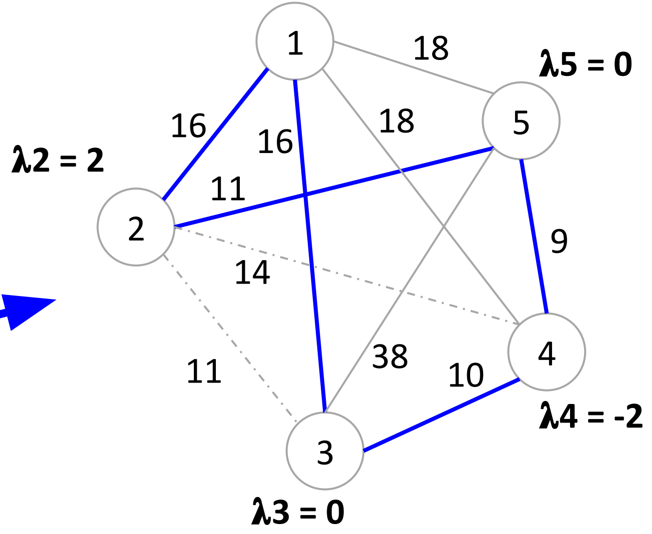




Held-Karp Relaxation

Bound: **59**





Bound: **62**

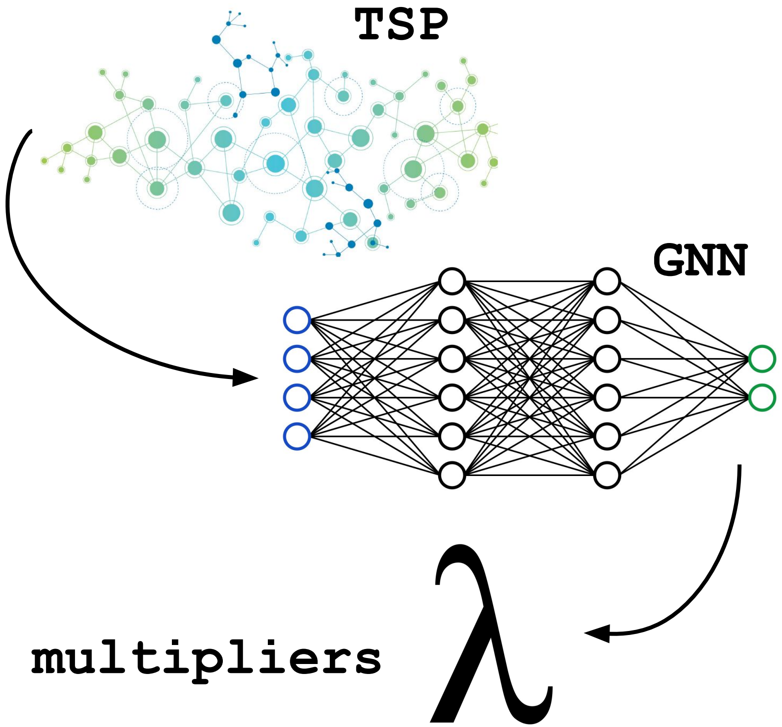
Plan

1. 1-tree and Held-Karp Relaxation
2. Proposed Approach
3. Application and Results

Proposed Approach

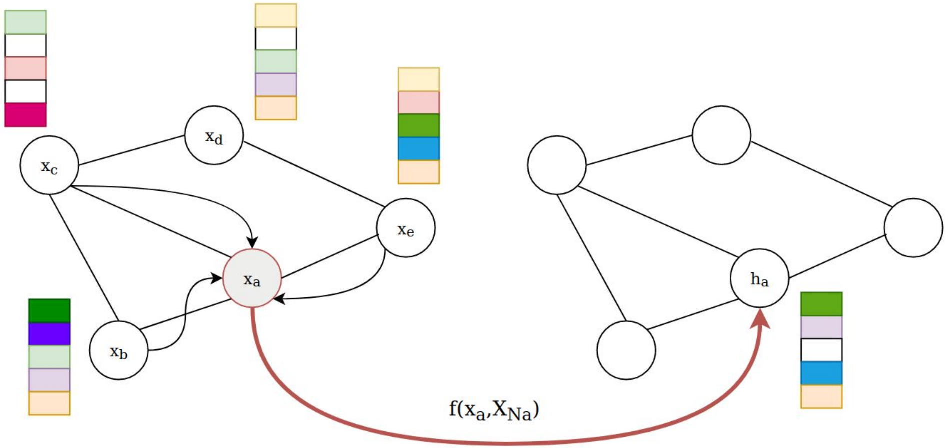
- Predicting the **Lagrangian Multipliers** to kickstart the Lagrangian relaxation

- Leverage the graph structure of the TSP by using **graph neural networks (GNN)** for the multipliers inference



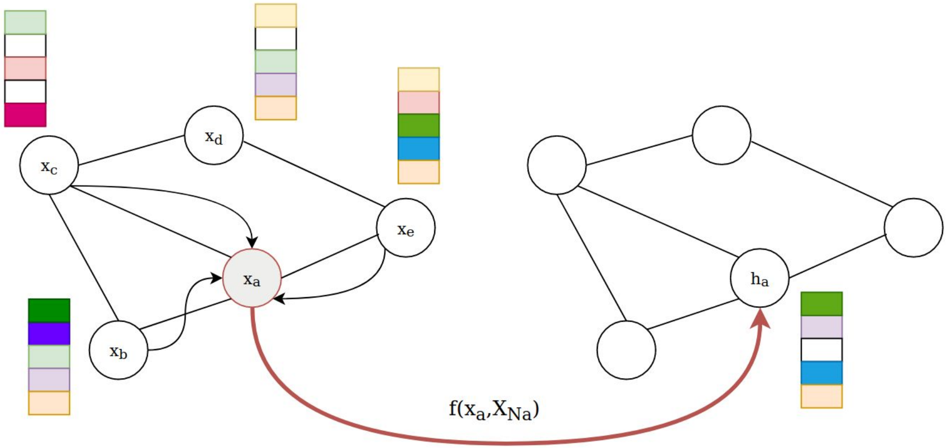
Graph Neural Networks

- **Graph Neural Networks** (GNNs) process graph-structured data:
    - **Node Representation:** GNNs learn to represent nodes based on their features and their neighbors'
    - **Message Passing:** Iterative process where nodes update their states by exchanging information with adjacent nodes



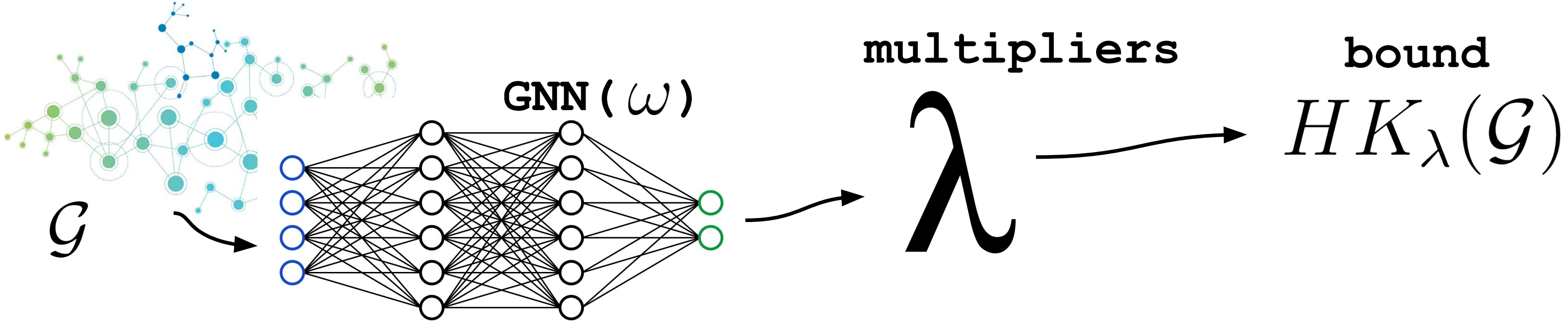
Graph Neural Networks

- **GNNs for TSP:**
  - **TSP graph structure:** GNNs handle the **features** and **variable size** of TSP representations
  - **Multipliers generation:** GNNs capture complex dependencies and patterns in the graph, aiding in better optimization



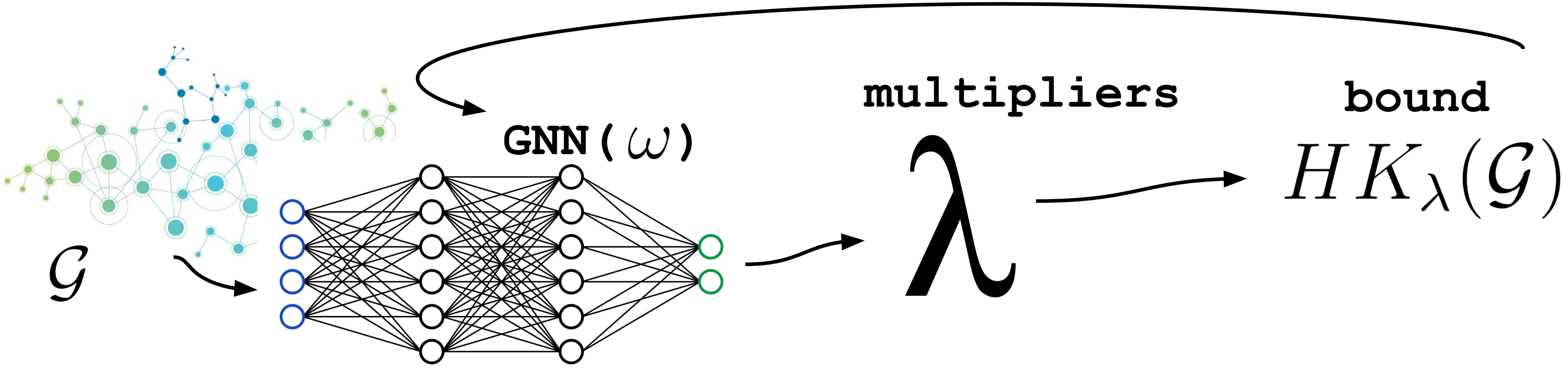
Learning Approach

- **Self-supervised Learning** for the **prediction of Lagrangian Multipliers:**
  - A GNN with parameters ω is used to predict Lagrangian multipliers,
  - the GNN is trained by computing the gradient on the bound



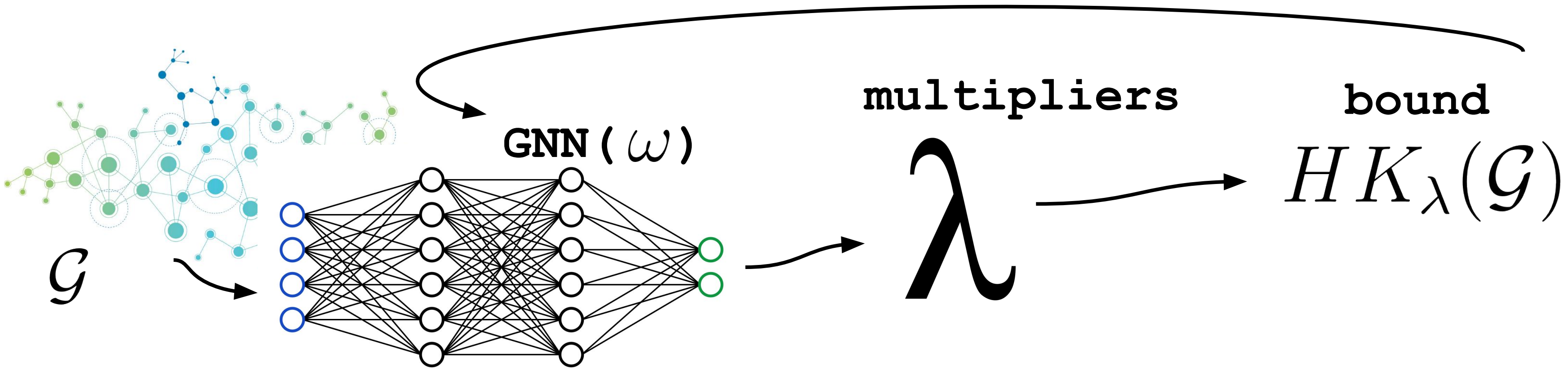
Learning Approach

$$\frac{\partial HK_\lambda(\mathcal{G})}{\partial \omega}$$



Learning Approach

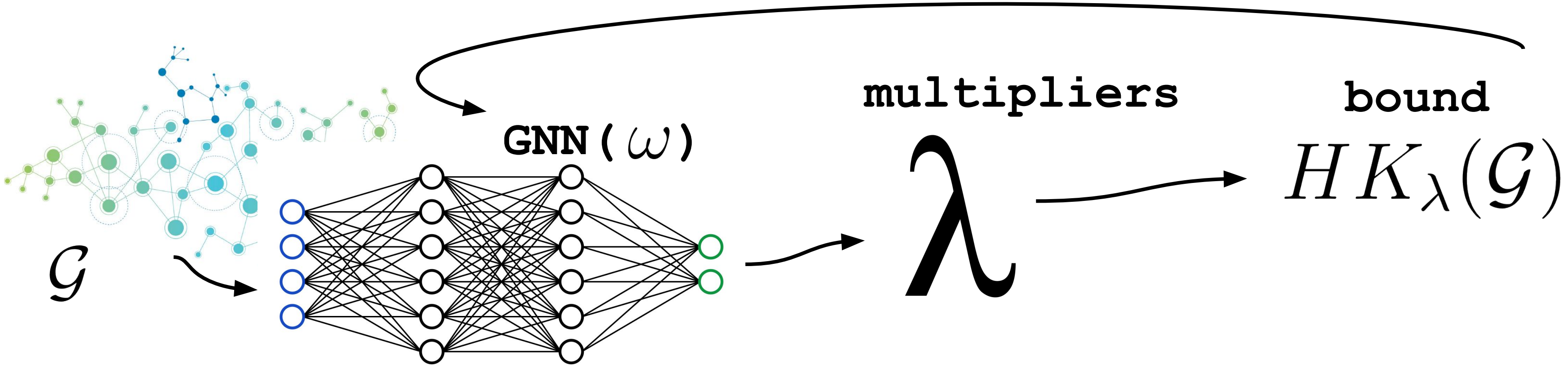
$$\frac{\partial HK_\lambda(\mathcal{G})}{\partial \omega} = \frac{\partial HK_\lambda(\mathcal{G})}{\partial \lambda} \times \frac{\partial \lambda}{\partial \omega}$$



Learning Approach

$$\frac{\partial HK_\lambda(\mathcal{G})}{\partial \omega} = \frac{\partial HK_\lambda(\mathcal{G})}{\partial \lambda} \times \frac{\partial \lambda}{\partial \omega}$$

GNN backprop

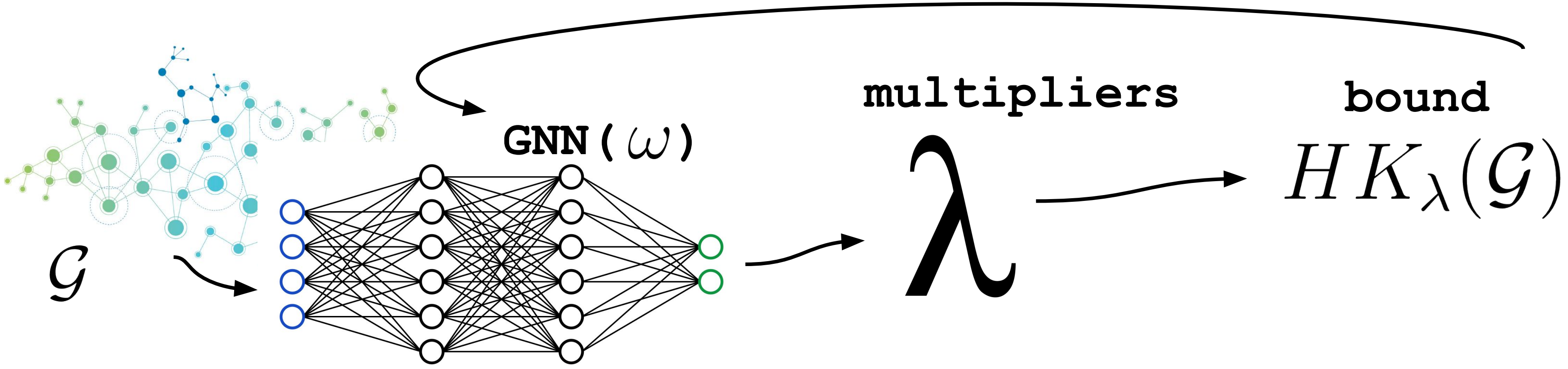


Learning Approach

$$\frac{\partial HK_\lambda(\mathcal{G})}{\partial \omega} = \frac{\partial HK_\lambda(\mathcal{G})}{\partial \lambda} \times \frac{\partial \lambda}{\partial \omega}$$

HK multipliers update

GNN backprop



Held-Karp multipliers update

$$\frac{\partial HK_\lambda(\mathcal{G})}{\partial \omega} = \frac{\partial HK_\lambda(\mathcal{G})}{\partial \lambda} \times \frac{\partial \lambda}{\partial \omega}$$

Held-Karp relaxation

$$\min \sum_{e \in E} c_e x_e + \sum_{i \in V} \lambda_i \Big(\sum_{e \in \delta(i)} x_e - 2 \Big) \longrightarrow \frac{\partial \mathrm{HK}(.)}{\partial \lambda_i} = \sum_{e \in \delta(i)} x_e - 2$$

$$\sum_{e \in E} x_e = |V|$$

$$\sum_{e \in S} x_e \leq |S| - 1, \ \forall S \text{ subtour of } G$$

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

HK multipliers update

Plan

1. 1-tree and Held-Karp Relaxation
2. Proposed Approach
3. Application and Results

Training - Datasets

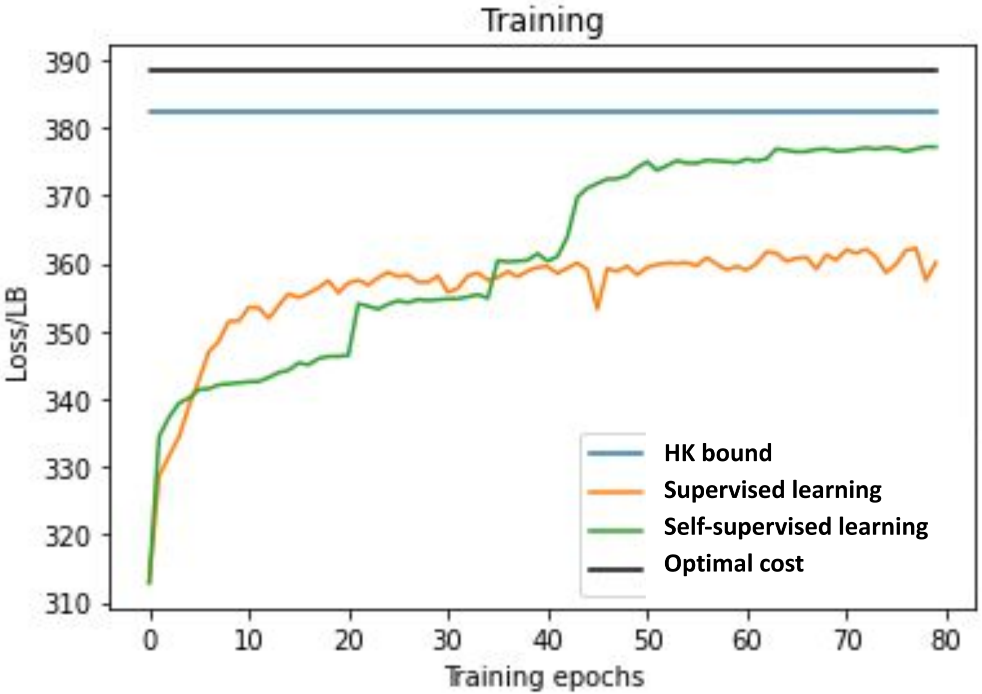
- Dataset randomly generated: **`Random`**, **`Clustered`** in sizes 100 and 200
- Existing dataset: **`Hard`**

- **Training set**: TSP instances + branch-and-bound subgraphs
- **Features**: distances and BnB state on *edges*, average/min distances on *nodes*

Training - Model and Baselines

- Baselines:
  - Optimal Solution,
  - HK Relaxation,
  - Supervised (graph attention network + fully-connected NN)
- Trained model:
  - Self-Supervised (graph attention network + fully-connected NN)

Training



Training
390
380
370
360
350
340
330
320
310
Loss/LB
0
10
20
30
40
50
60
70
80
Training epochs
HK bound
Supervised learning
Self-supervised learning
Optimal cost

Integration with Branch-and-Bound

- $HK_{\mathrm{GNN}_\omega}(\mathcal{G})$ provides a tight bound at each step of a BnB algorithm, enabling **pruning** and **filtering**
- Domain filtering: edges are **forced** or **excluded** from the solution based on replacement or insertion costs based on the 1-tree

Benchimol P, van Hoeve WJ, Régin J-C, Rousseau L-M, Rueher R, (2012), **Improved Filtering for Weighted Circuit Constraints**, *Constraints* 17:3, 205-233.

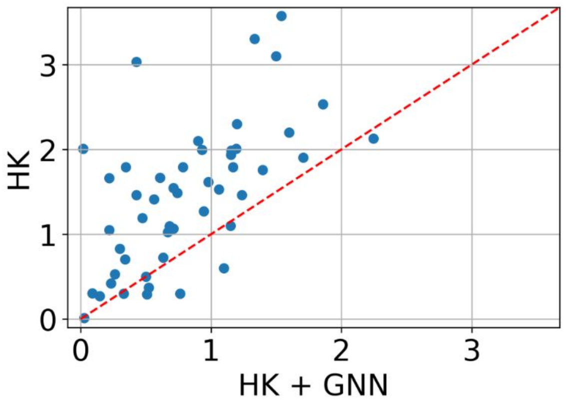
Integration with Branch-and-Bound

Practical considerations:

1. **Edges forced or excluded** from the solution modify the graph, which is **reflected in the features**
2. GNN-based multipliers generation is used in the **10 first nodes** of the BnB
3. **HK+GNN**: the trained GNN is used to bootstrap the multipliers generation

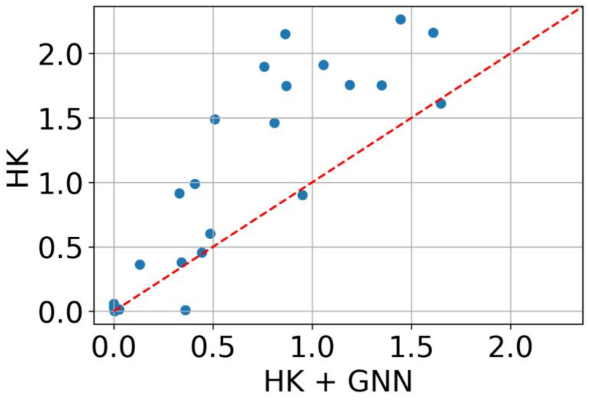
Experimental Results

- Compared to a BnB using HK alone, **HK+GNN** multiplier generation:
  - is 10% quicker in solving instances to optimality
  - reduces by 50% the optimality gap on timeout instances



(a) Results on Random200.

optimality gap in %



(b) Results on Clustered200.

Experimental Results

- Compared to a BnB using HK alone, **HK+GNN** multiplier generation:
  - is 10% quicker in solving instances to optimality
  - reduces by 50% the optimality gap on timeout instances

Configuration	Branch-and-bound with standard Held-Karp (HK)					Branch-and-bound with our approach (GNN + HK)				
	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)
Random100	559	41/50	1127k	75.9	0.88					
Random200	1800	0/50	4.71m	67.8	1.82					
Clustered100	643	38/50	497k	17.7	0.19					
Clustered200	1800	0/50	922k	9.9	0.68					
Hard	1800	0/6	9.59M	6.4	0.32					

Experimental Results

- Compared to a BnB using HK alone, **HK+GNN** multiplier generation:
  - is 10% quicker in solving instances to optimality
  - reduces by 50% the optimality gap on timeout instances

Configuration	Branch-and-bound with standard Held-Karp (HK)					Branch-and-bound with our approach (GNN + HK)				
	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)
Random100	559	41/50	1127k	75.9	0.88	497 (- 11%)				
Random200	1800	0/50	4.71m	67.8	1.82	1800				
Clustered100	643	38/50	497k	17.7	0.19	590 (- 8%)				
Clustered200	1800	0/50	922k	9.9	0.68	1800				
Hard	1800	0/6	9.59M	6.4	0.32	1800				

Experimental Results

- Compared to a BnB using HK alone, **HK+GNN** multiplier generation:
  - is 10% quicker in solving instances to optimality
  - reduces by 50% the optimality gap on timeout instances

Configuration	Branch-and-bound with standard Held-Karp (HK)					Branch-and-bound with our approach (GNN + HK)				
	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)
Random100	559	41/50	1127k	75.9	0.88	497 (- 11%)	46/50 (+ 10%)			
Random200	1800	0/50	4.71m	67.8	1.82	1800	0/50 (+ 0%)			
Clustered100	643	38/50	497k	17.7	0.19	590 (- 8%)	40/50 (+ 5%)			
Clustered200	1800	0/50	922k	9.9	0.68	1800	0/50 (+ 0%)			
Hard	1800	0/6	9.59M	6.4	0.32	1800	0/6 (+ 0%)			

Experimental Results

- Compared to a BnB using HK alone, **HK+GNN** multiplier generation:
  - is 10% quicker in solving instances to optimality
  - reduces by 50% the optimality gap on timeout instances

Configuration	Branch-and-bound with standard Held-Karp (HK)					Branch-and-bound with our approach (GNN + HK)				
	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)
Random100	559	41/50	1127k	75.9	0.88	497 (- 11%)	46/50 (+ 10%)	965k (- 14%)		
Random200	1800	0/50	4.71m	67.8	1.82	1800	0/50 (+ 0%)	4.26m (- 10%)		
Clustered100	643	38/50	497k	17.7	0.19	590 (- 8%)	40/50 (+ 5%)	470k (- 5%)		
Clustered200	1800	0/50	922k	9.9	0.68	1800	0/50 (+ 0%)	690k (- 25%)		
Hard	1800	0/6	9.59M	6.4	0.32	1800	0/6 (+ 0%)	9.36M (- 2%)		

Experimental Results

- Compared to a BnB using HK alone, **HK+GNN** multiplier generation:
  - is 10% quicker in solving instances to optimality
  - reduces by 50% the optimality gap on timeout instances

Configuration	Branch-and-bound with standard Held-Karp (HK)					Branch-and-bound with our approach (GNN + HK)				
	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)
Random100	559	41/50	1127k	75.9	0.88	497 (- 11%)	46/50 (+ 10%)	965k (- 14%)	77.7 (+ 2%)	
Random200	1800	0/50	4.71m	67.8	1.82	1800	0/50 (+ 0%)	4.26m (- 10%)	70.6 (+ 4%)	
Clustered100	643	38/50	497k	17.7	0.19	590 (- 8%)	40/50 (+ 5%)	470k (- 5%)	20.3 (+ 15%)	
Clustered200	1800	0/50	922k	9.9	0.68	1800	0/50 (+ 0%)	690k (- 25%)	12.6 (+27%)	
Hard	1800	0/6	9.59M	6.4	0.32	1800	0/6 (+ 0%)	9.36M (- 2%)	6.5 (+1%)	

Experimental Results

- Compared to a BnB using HK alone, **HK+GNN** multiplier generation:
  - is 10% quicker in solving instances to optimality
  - reduces by 50% the optimality gap on timeout instances

Configuration	Branch-and-bound with standard Held-Karp (HK)					Branch-and-bound with our approach (GNN + HK)				
	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)	Time (sec.)	# solved (/50)	PDI	Filt. (%)	Opt. gap (%)
Random100	559	41/50	1127k	75.9	0.88	497 (- 11%)	46/50 (+ 10%)	965k (- 14%)	77.7 (+ 2%)	0.48 (- 45%)
Random200	1800	0/50	4.71m	67.8	1.82	1800	0/50 (+ 0%)	4.26m (- 10%)	70.6 (+ 4%)	0.59 (- 68%)
Clustered100	643	38/50	497k	17.7	0.19	590 (- 8%)	40/50 (+ 5%)	470k (- 5%)	20.3 (+ 15%)	0.08 (- 58%)
Clustered200	1800	0/50	922k	9.9	0.68	1800	0/50 (+ 0%)	690k (- 25%)	12.6 (+27%)	0.38 (- 44%)
Hard	1800	0/6	9.59M	6.4	0.32	1800	0/6 (+ 0%)	9.36M (- 2%)	6.5 (+1%)	0.31 (- 3%)

Conclusion

- Introduction of a **self-supervised learning approach** using **GNNs** and **Held-Karp Lagrangian relaxation** to predict accurate Lagrangian multipliers

- The approach has potential applications for **more challenging TSP variants**, and **other Lagrangian relaxations**

- Limitations: **costly GNN** calls, best results when the graph distribution is known beforehand

Learning Lagrangian Multipliers for the TSP

Thanks!

Augustin Parjadis, Louis-Martin Rousseau,
Quentin Cappart, Bistra Dilkina, Aaron Ferber

arxiv.org/abs/2312.14836