ML4CP Leuven 2023

Deep Learning & Combinatorial Optimization

Wouter Kool

University of Amsterdam

ORTEC

AMLAB Amsterdam Machine Learning Lab

Wouter

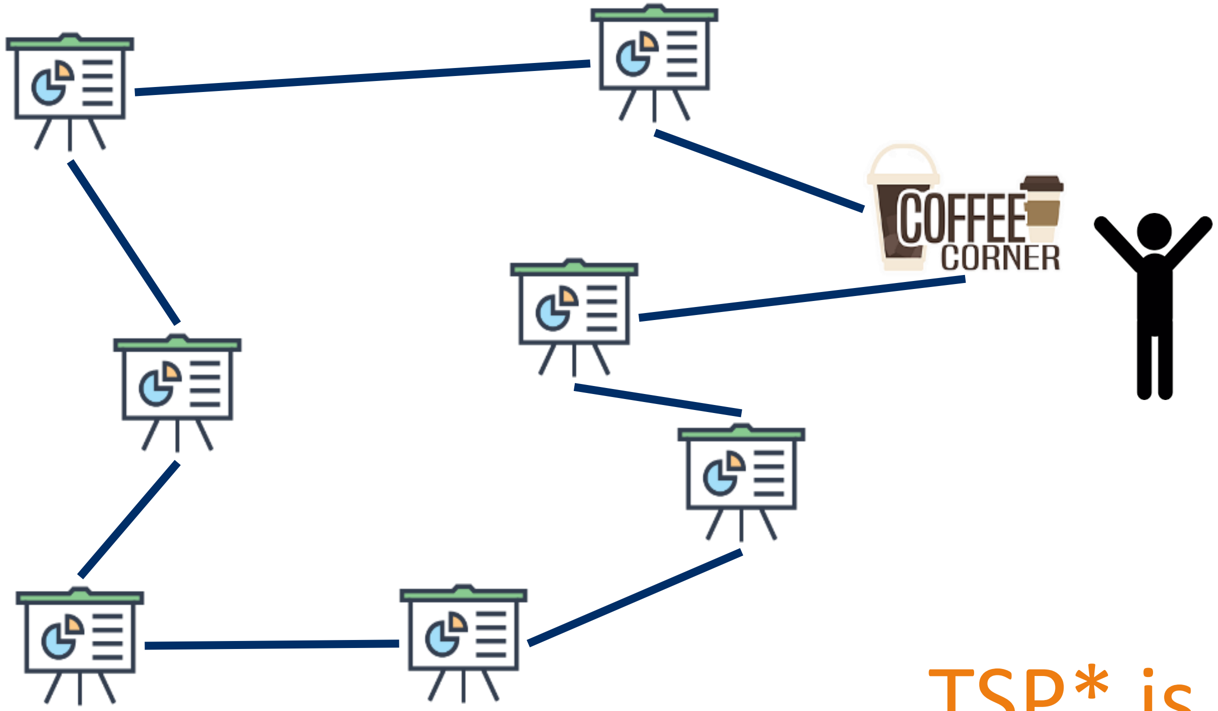
2015 Master of Business Analytics &
2015 Master of Econometrics & OR

2022 PhD in Machine Learning

9 years at ORTEC OR engineer

Research intern @

Travelling Scientist Problem (TSP)



TSP* is (NP-)hard!

Kool et al., 2019

UNIVERSITY OF AMSTERDAM ORTEC

What does it mean?

Finding *optimal* solutions for *all* problem instances

MISSION: IMPOSSIBLE

* unless P = NP

Finding *acceptable* solutions for *relevant* problem instances

MISSION: ~~IM~~POSSIBLE

We use HEURISTICS
Can be seen as ‘rules of thumb’

‘next location should be nearby’

Designing of heuristics is like feature engineering

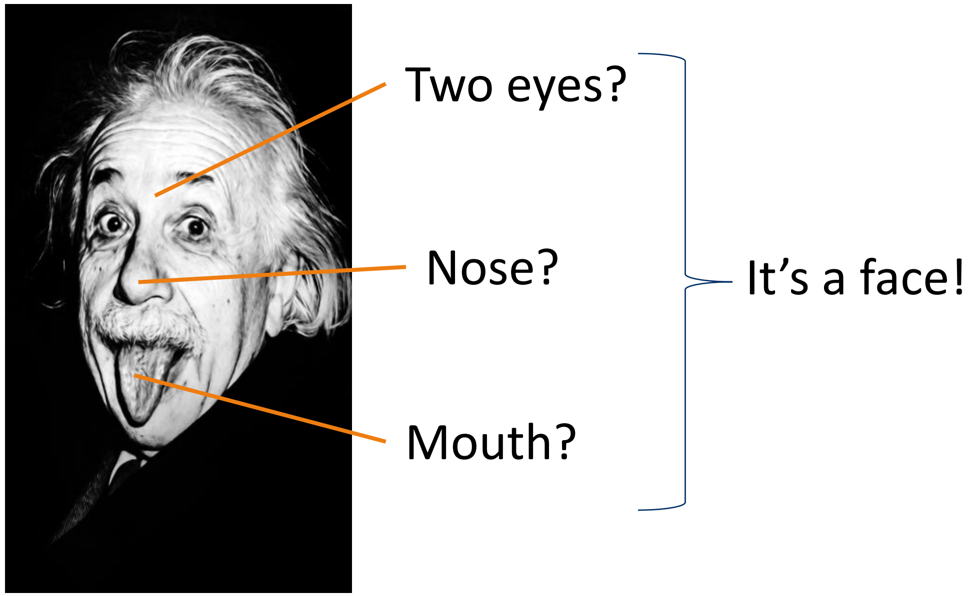
HARD WORK

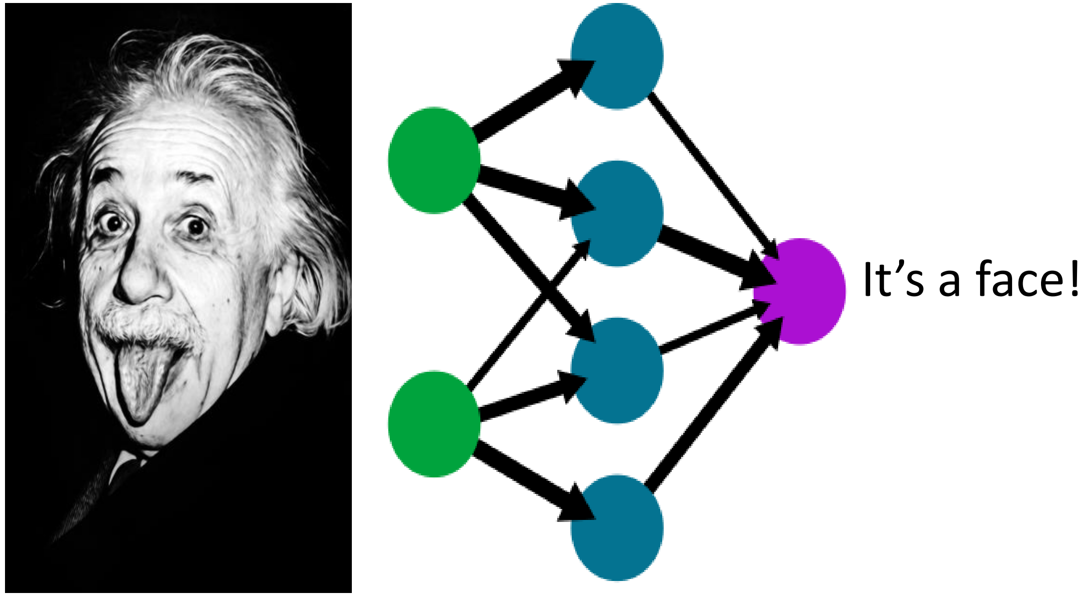
Computer Vision Features
(SIFT, etc.)

Feature engineering
- Needs expert knowledge
- Time consuming hand-tuning

So what do we do?

Designing of heuristics is like feature engineering

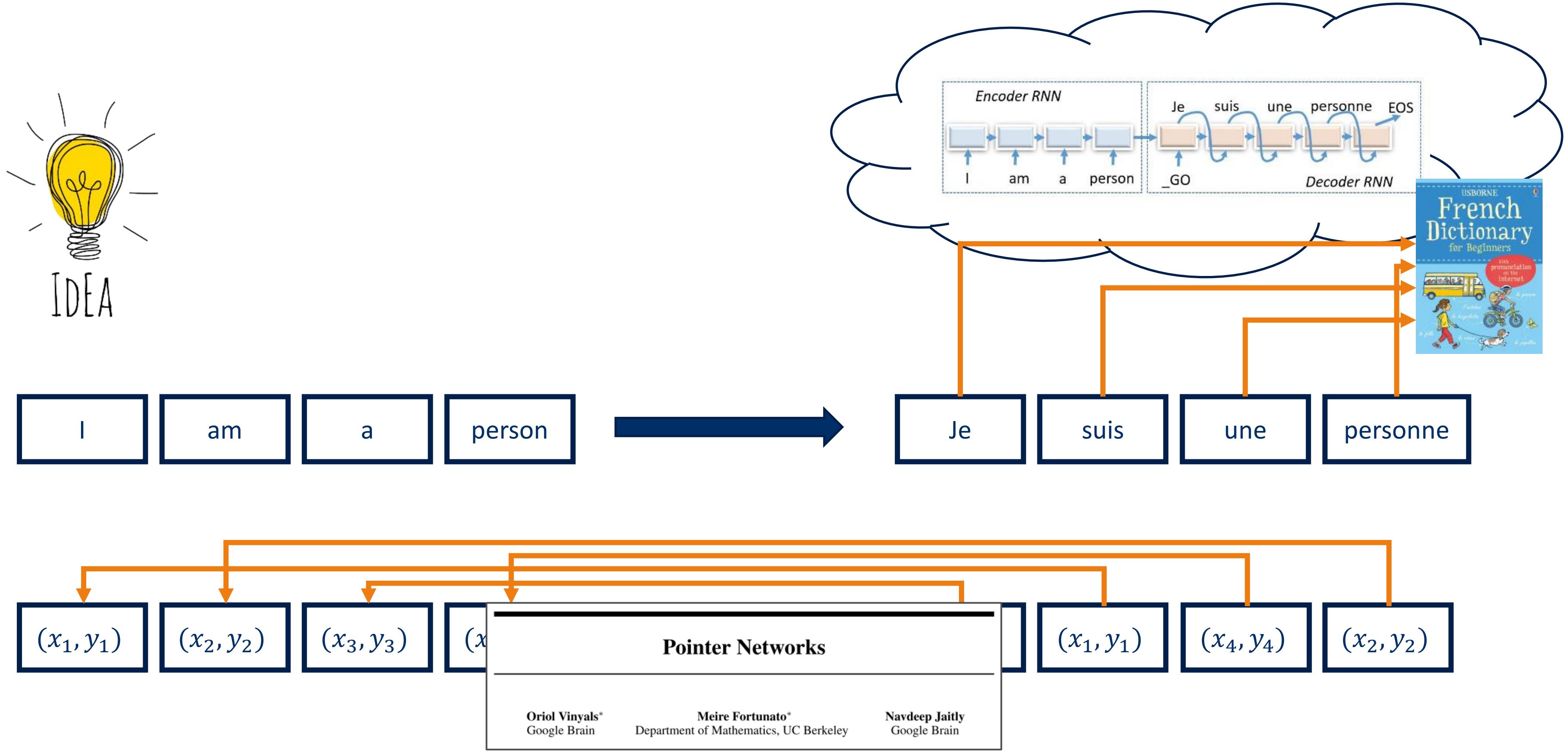




Traditional approach
Feature engineering

Deep Learning
No feature engineering

‘Translate’ problem into solution



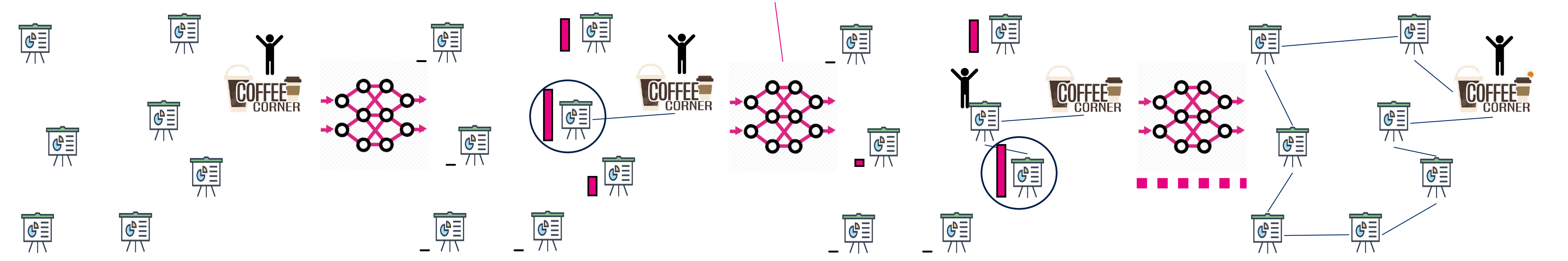
UNIVERSITY OF AMSTERDAM ORTEC

How does that work?

Instance $s = ((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$

Model $p_{\boldsymbol{\theta}}(\pi_t | s, \pi_{<t})$= $p_{\boldsymbol{\theta}}$(next node | partial tour)

Solution $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots)$ with length $L(\boldsymbol{\pi})$



Sample $\pi_1 \sim p_{\boldsymbol{\theta}}(\pi_1 | s)$

Sample $\pi_2 \sim p_{\boldsymbol{\theta}}(\pi_2 | s, \pi_1)$

Sample $\pi_t \sim p_{\boldsymbol{\theta}}(\pi_t | s, \pi_{<t})$

Randomized algorithm with expected cost:

$$E_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)}[L(\boldsymbol{\pi})]$$

How to optimize $\boldsymbol{\theta}$?

NEURAL COMBINATORIAL OPTIMIZATION WITH REINFORCEMENT LEARNING

Irwan Bello*, Hieu Pham*, Quoc V. Le, Mohammad Norouzi, Samy Bengio
Google Brain
{ibello,hyhieu,qvl,mnorouzi,bengio}@google.com

REINFORCE (for dummies)

Repeat

Do something

Result = ?

Good!

Bad!

Do more often!

Do less often!

Increase $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$

Decrease $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$

Sample $\boldsymbol{\pi} \sim p_{\boldsymbol{\theta}}(\cdot\,|s)$

$L(\boldsymbol{\pi}) = 7.43$

We need a *baseline* to compare against: *rollout* earlier model

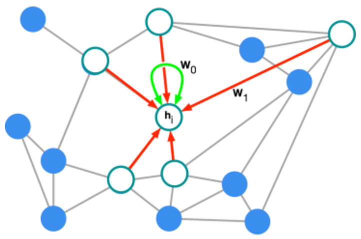
$\boldsymbol{\pi}^{bl} \sim p_{\boldsymbol{\theta}^{bl}}(\cdot\,|s)$ **(greedy!)**

$L(\boldsymbol{\pi}^{bl}) = 6.89$

What’s the model architecture?

$$p_{\theta}(\pi_t | s, \pi_{<t})$$

Graph convolutions



+

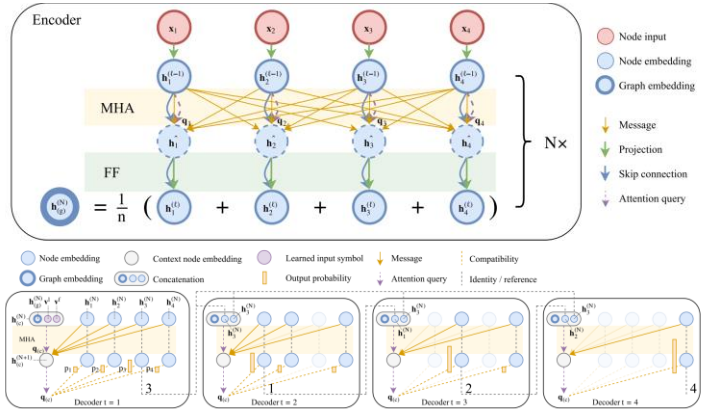
Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Read the paper…

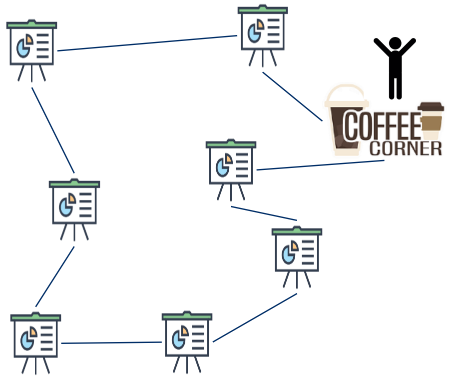
ATTENTION, LEARN TO SOLVE ROUTING PROBLEMS!

Wouter Kool, University of Amsterdam, w.w.m.kool@uva.nl
Herke van Hoof, University of Amsterdam, h.c.vanhoof@uva.nl
Max Welling, University of Amsterdam, m.welling@uva.nl



Experiments

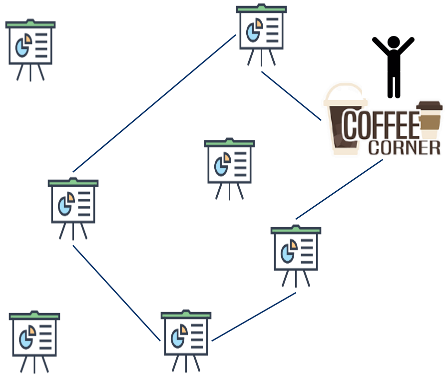
Travelling Salesman Problem (TSP)



Minimize length

Visit all nodes

Orienteering Problem (OP)



Maximize total prize

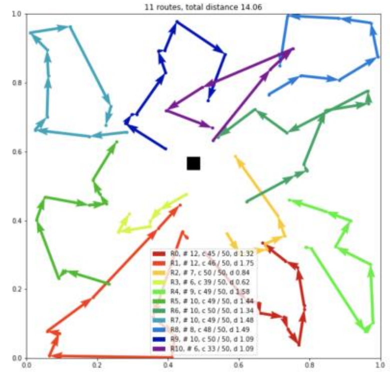
Max length constraint

(Stochastic) Prize Collecting TSP ((S)PCTSP)

Minimize length + penalties of unvisited nodes

Collect minimum total prize

Vehicle Routing Problem (VRP)



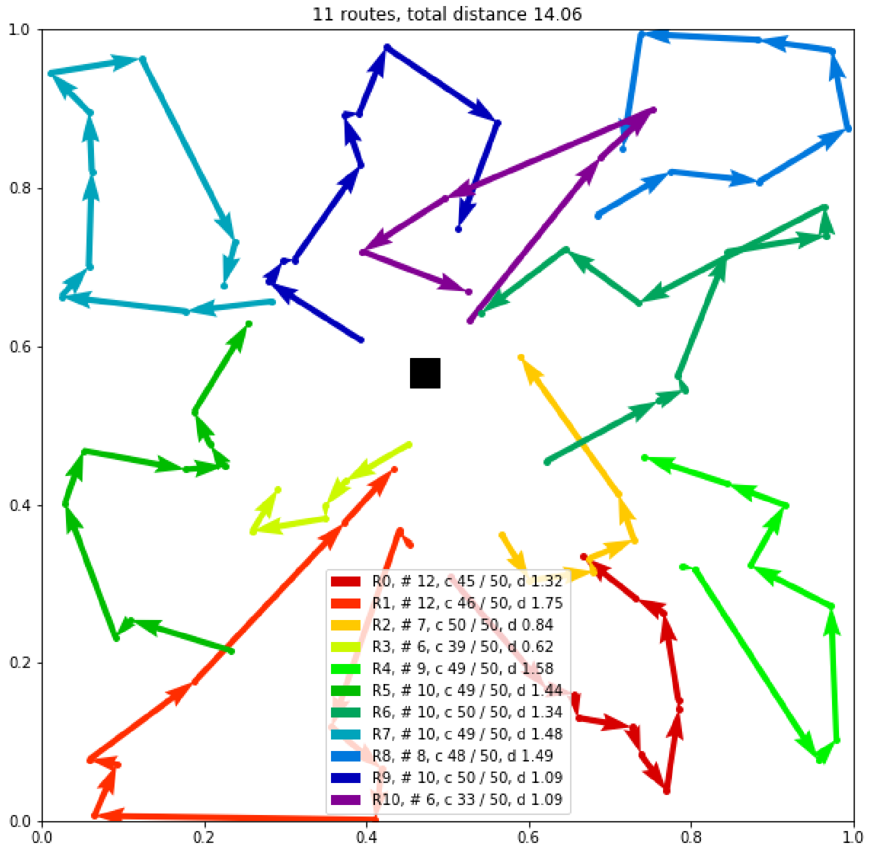
Minimize length

Visit all nodes

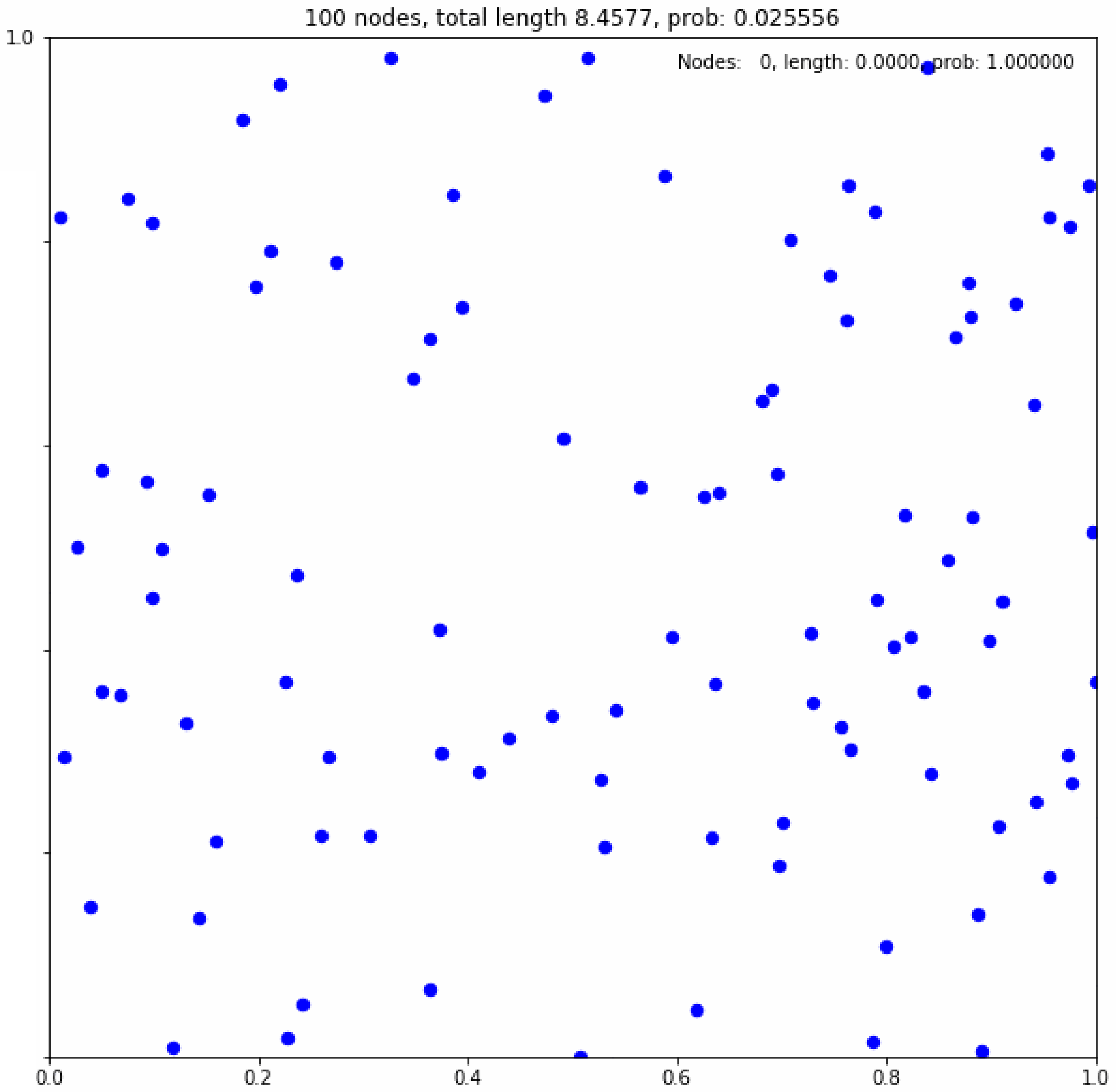
Total route demand must fit vehicle capacity

Train for each problem, *same hyperparameters*!

Results



11 routes, total distance 14.06
R0, # 12, c 45 / 50, d 1.32
R1, # 12, c 46 / 50, d 1.75
R2, # 7, c 50 / 50, d 0.84
R3, # 6, c 39 / 50, d 0.62
R4, # 9, c 49 / 50, d 1.58
R5, # 10, c 49 / 50, d 1.44
R6, # 10, c 50 / 50, d 1.34
R7, # 10, c 49 / 50, d 1.48
R8, # 8, c 48 / 50, d 1.49
R9, # 10, c 50 / 50, d 1.09
R10, # 6, c 33 / 50, d 1.09



100 nodes, total length 8.4577, prob: 0.025556
Nodes: 0, length: 0.0000, prob: 1.000000

Results Attention Model + Rollout Baseline

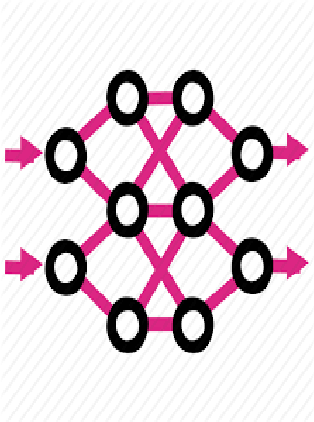
- Improves over classical heuristics!
- Improves over prior learned heuristics!
  - Attention Model improves
  - Rollout helps significantly
- Gets close to single-purpose SOTA (20 to 100 nodes)!
  - TSP 0.34% to 4.53% (greedy)
  - TSP 0.08% to 2.26% (best of 1280 samples)

Let’s analyze this method…

‘Predicting’ translations

Neural
Machine
Translation

Sentence

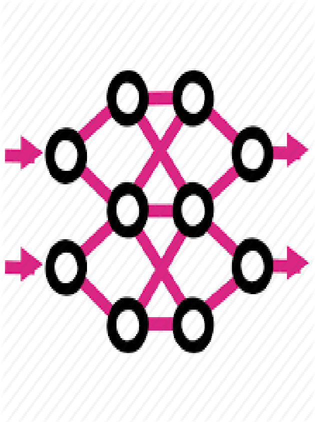


Translation

‘Predicting’ solutions

Neural Combinatorial Optimization

Problem



Solution

It’s not the same!

Machine Translation $\neq$ Combinatorial Optimization

It’s not the same!

Machine Translation $\neq$ Combinatorial Optimization

Learning problem

No problem

Scoring translations (learning a model)

Scoring solution (objective function)

Finding a good translation (inference)

Finding a good solution (optimization)

Computational problem

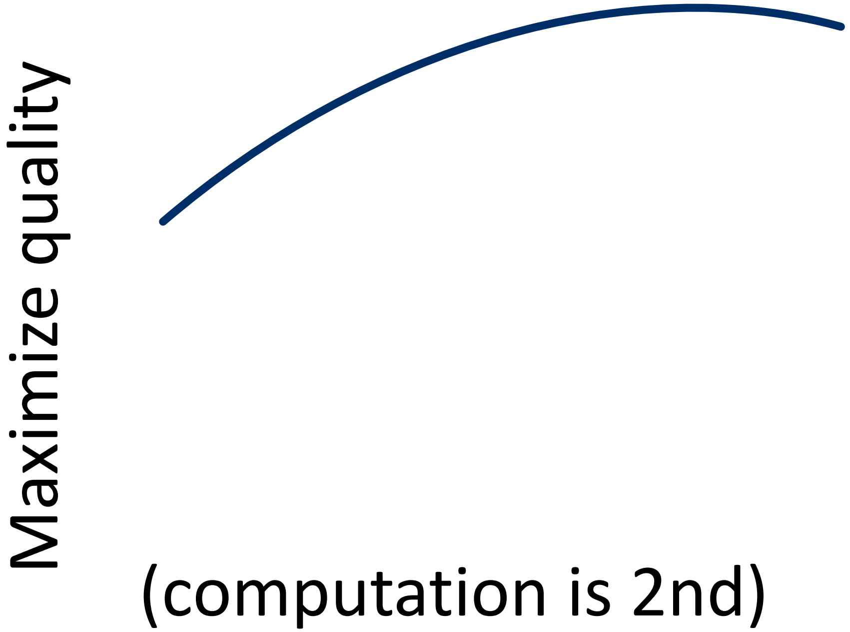
Stahlberg & Byrne, 2019

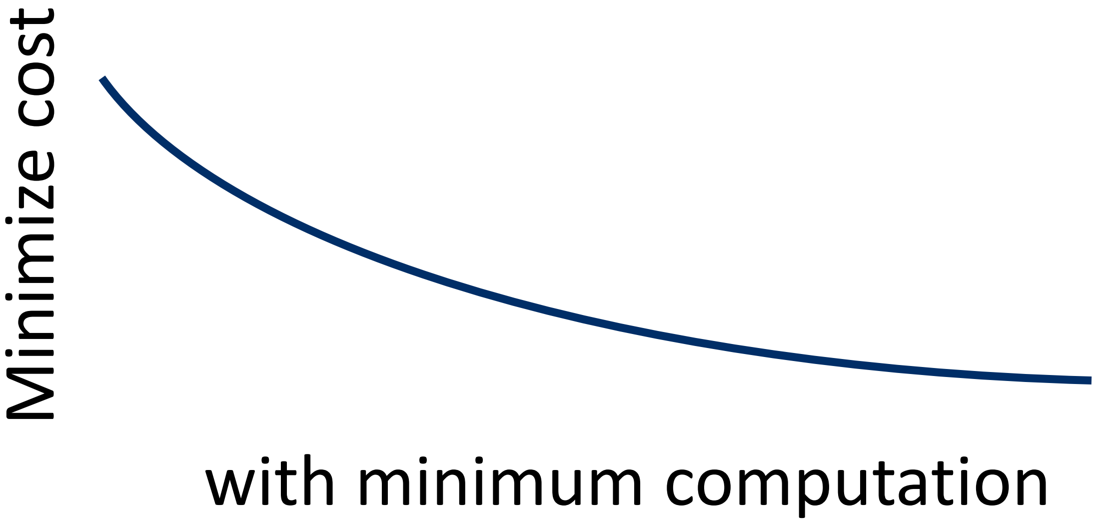
It’s not the same!

Machine Translation	$\neq$	Combinatorial Optimization
Maximize quality		Minimize cost
(computation is 2nd)		with minimum computation

It’s not the same!

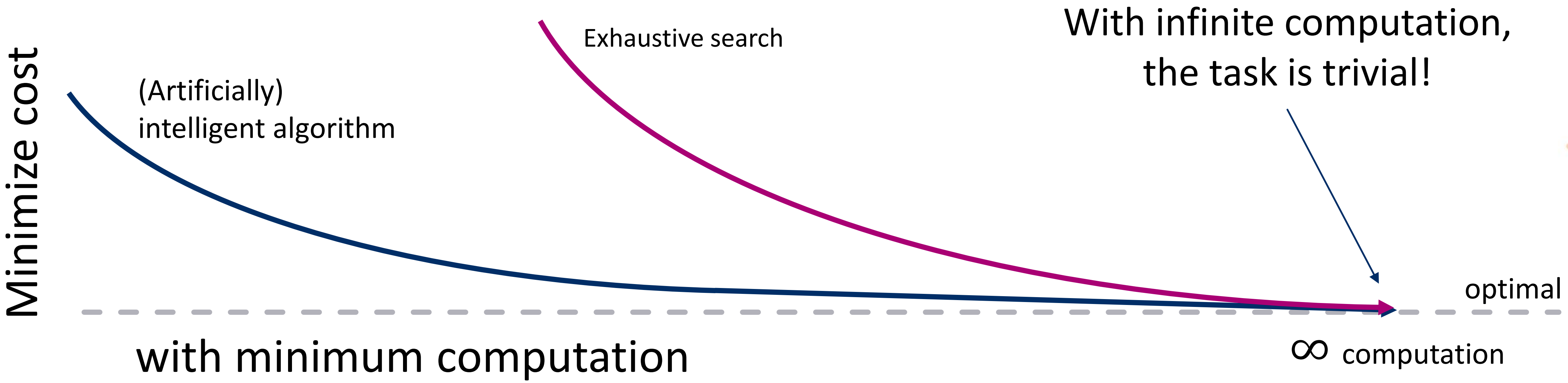
Machine Translation $\neq$ Combinatorial Optimization





If we have infinite computation…

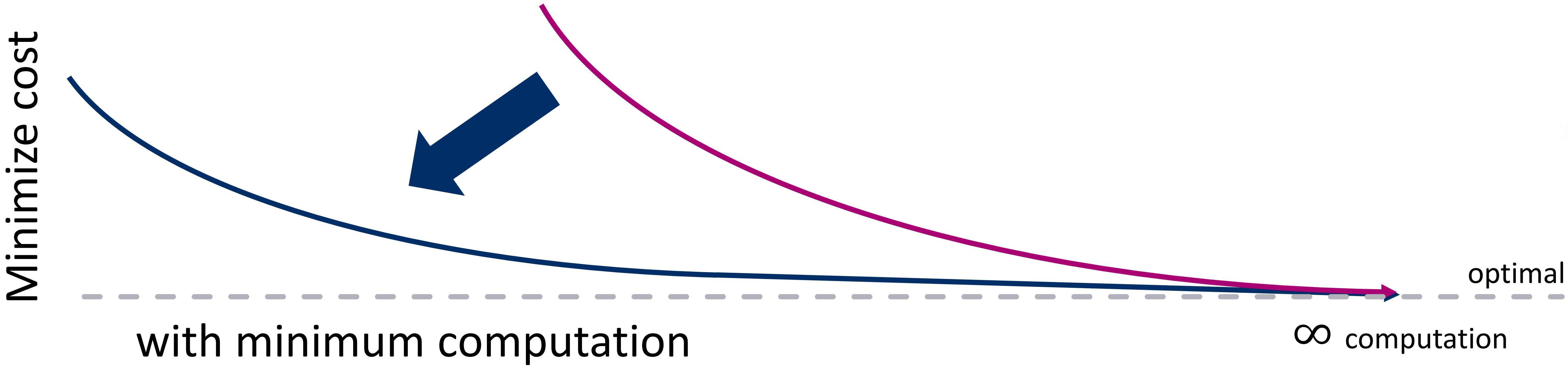
Combinatorial Optimization



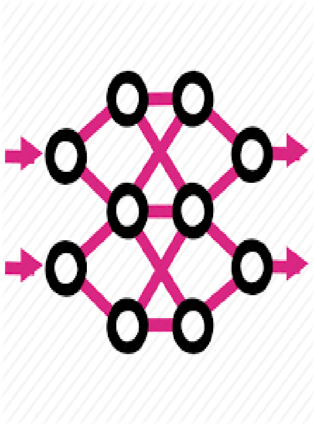
The goal

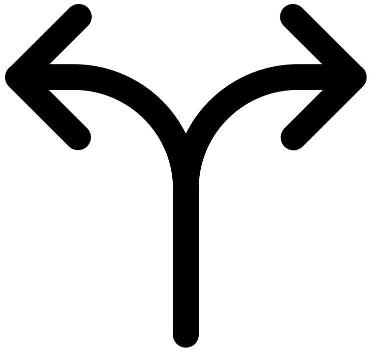
To find better solutions

...with less computation!



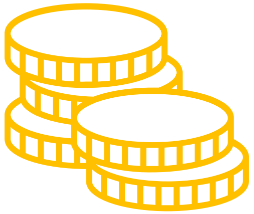
How?





Using neural networks… …to make better (heuristic) decisions!

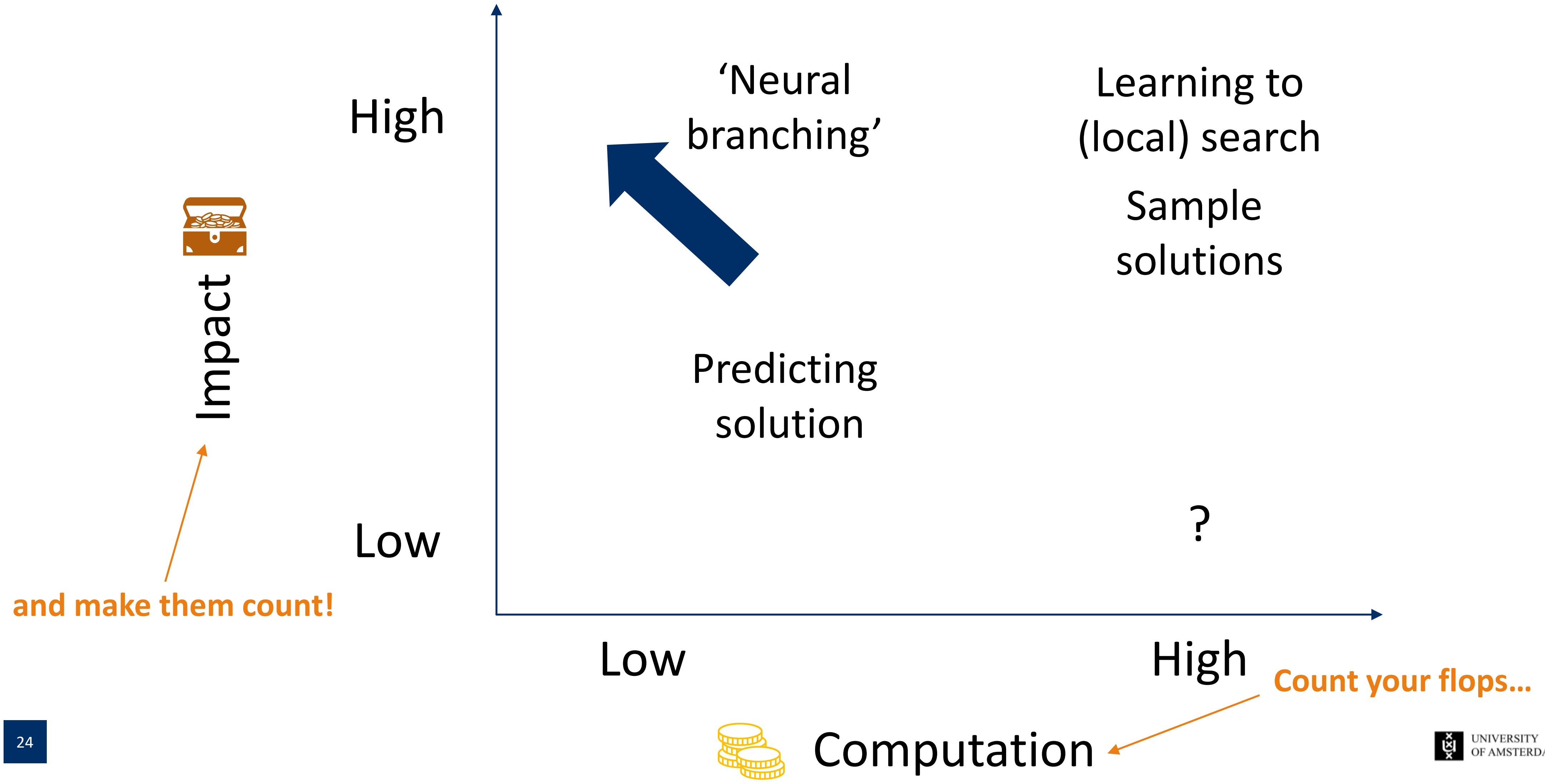
Adding computation… …to reduce computation!



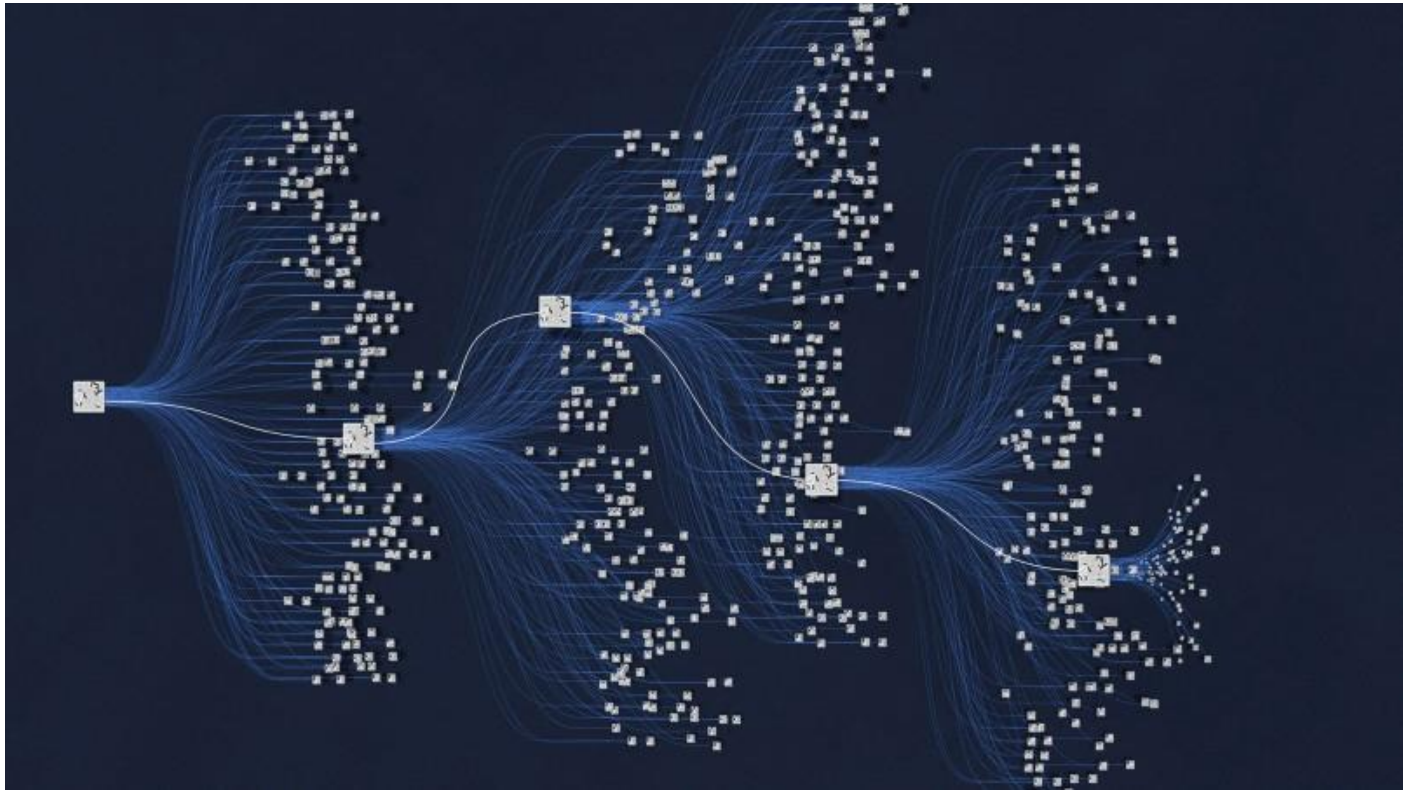
Investment

Pay-off

Impact vs. computation (of your neural network)



Example: AlphaGo

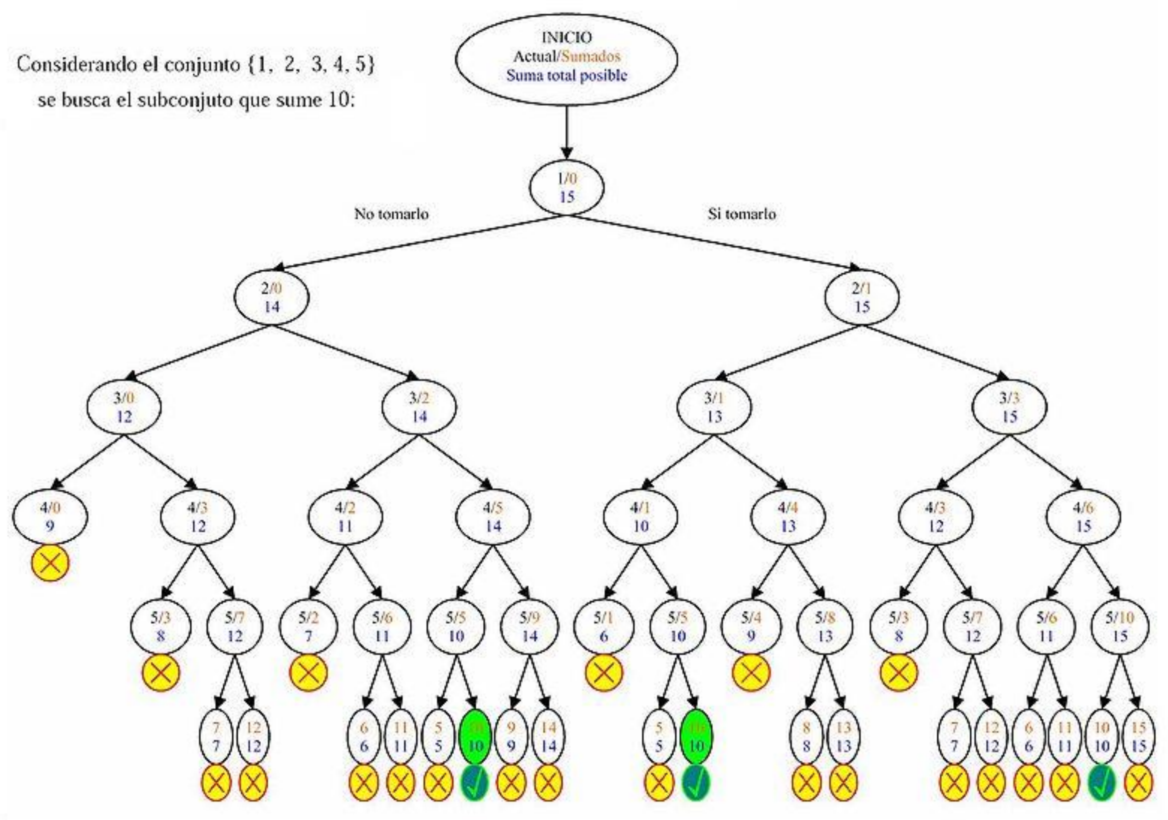


Uses (Deep) Neural Network to predict which part of the search tree to expand

ORTEC

Learning to Branch (& Bound?)

- Success depends on branch selection
- Often done with very simple heuristics
- Learn to predict best branch!
- Example of powerful **exact** method that can be improved by ML



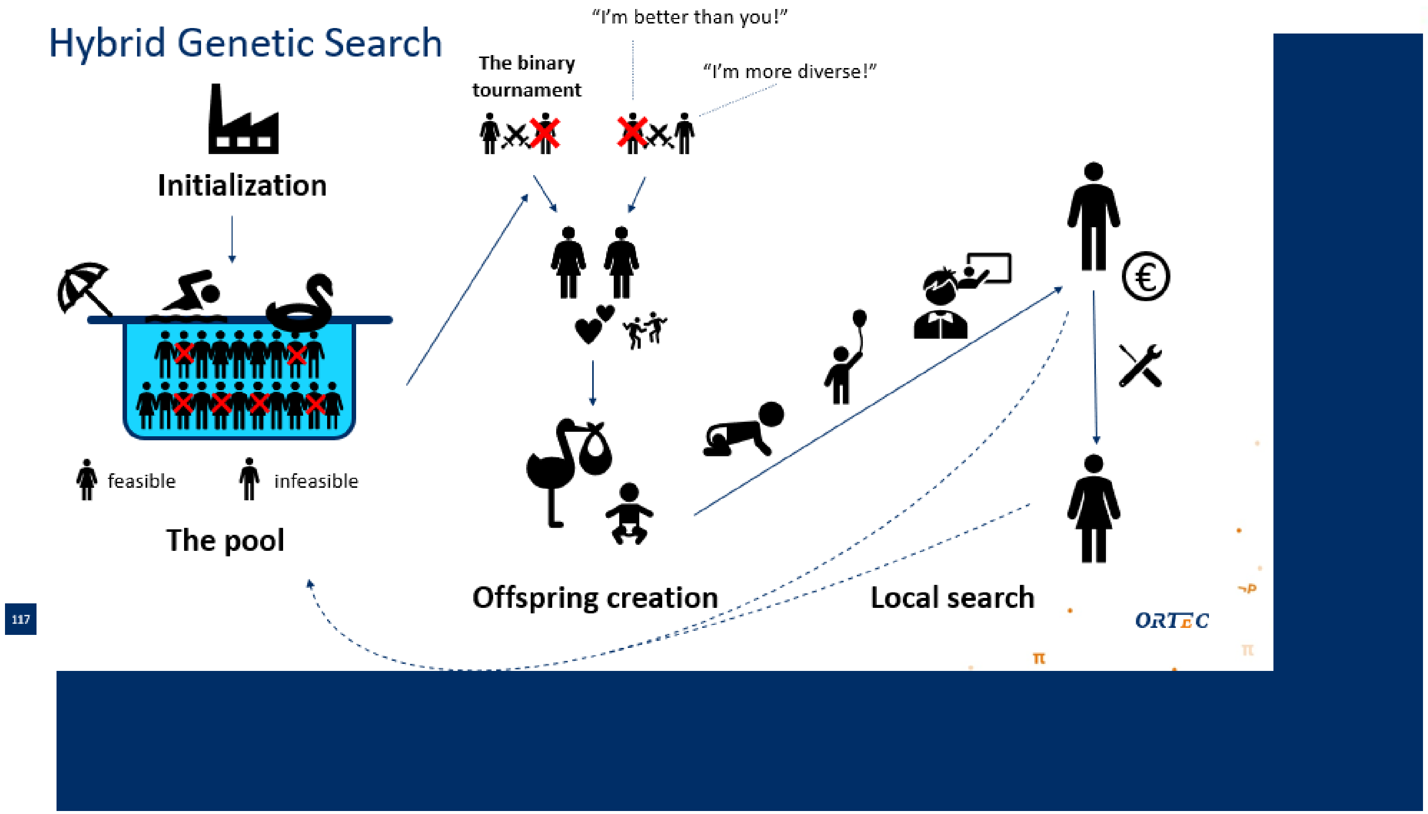
ORTEC

Idea: learn to select parents

- In a Hybrid Genetic Search algorithm

- More on that later…

…as this will be the lab!



ORTEC

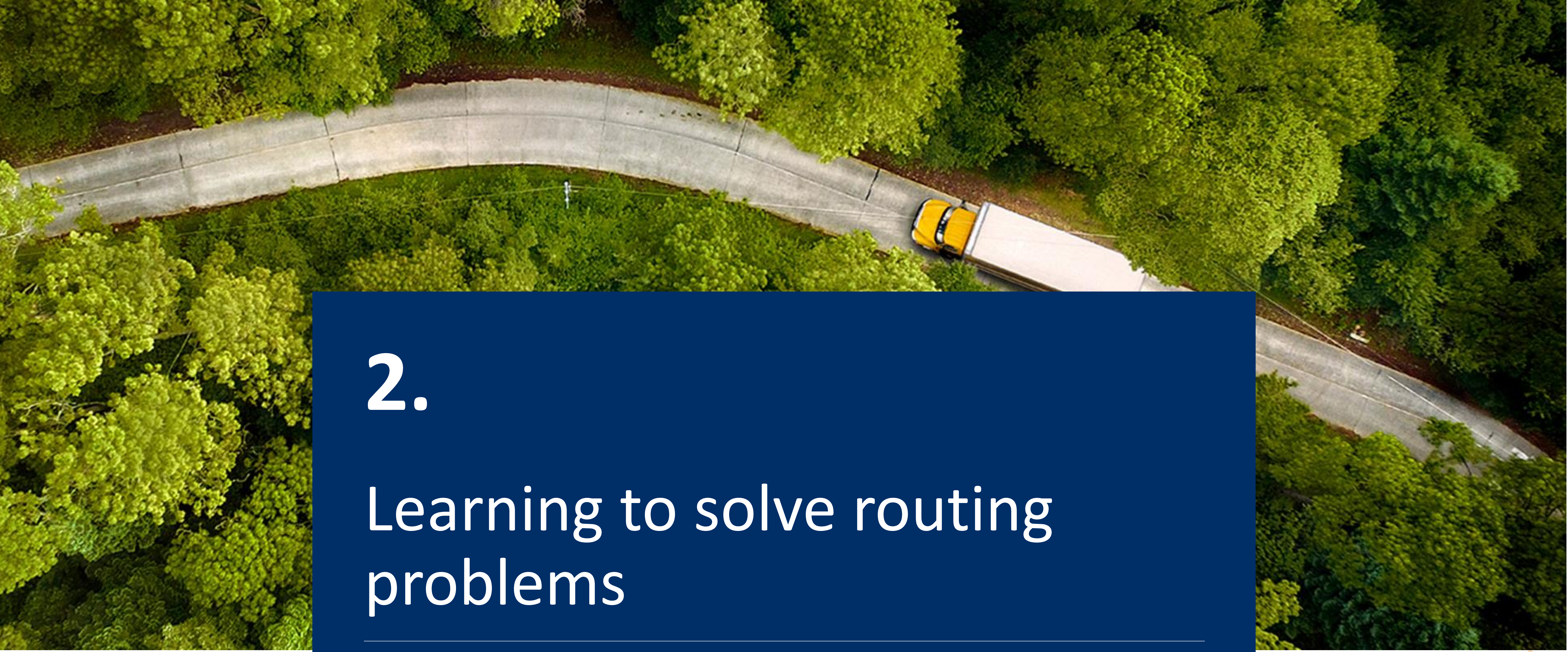
Combining the best of both worlds

Using what we know…

…without being limited by what we know.

There is more out there!

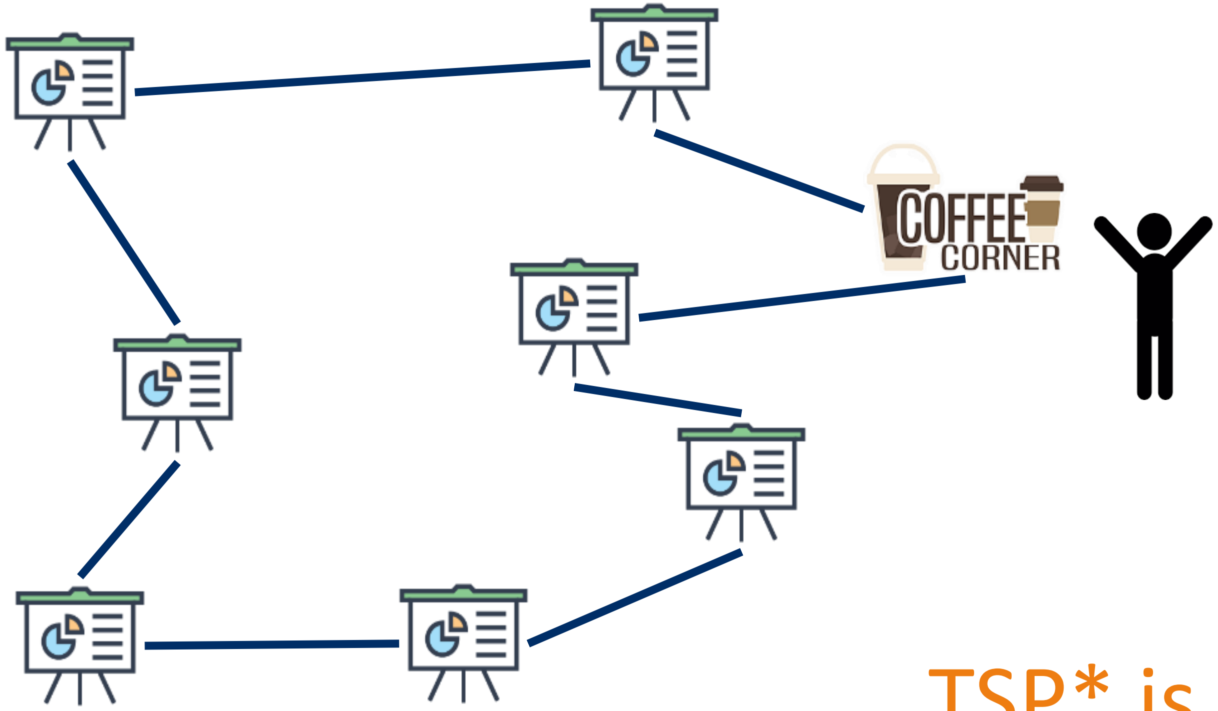
- So far, we considered using ML for learning HOW to optimize
- We can also use ML for learning WHAT to optimize
  - E.g.
    - learning driving durations for use in route optimization
    - learning constraints/objectives/preferences for scheduling
  - Sometimes referred to as ‘predict, then optimize’ framework
- We saw this yesterday, so not for today!



2.

Learning to solve routing problems

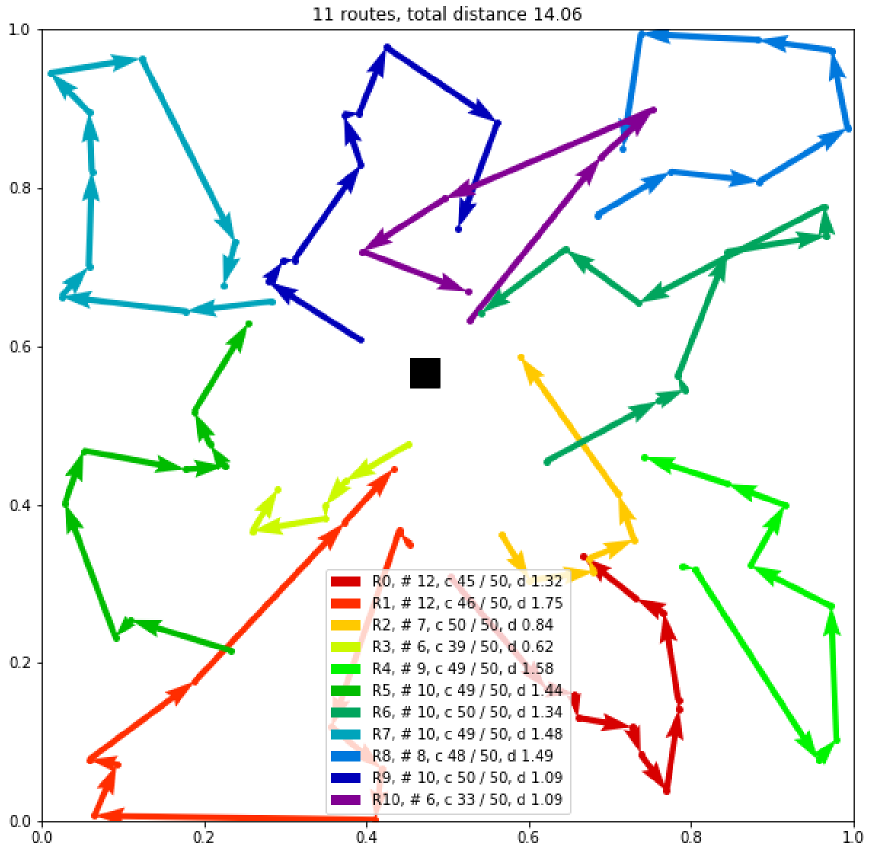
Travelling Scientist Problem (TSP)



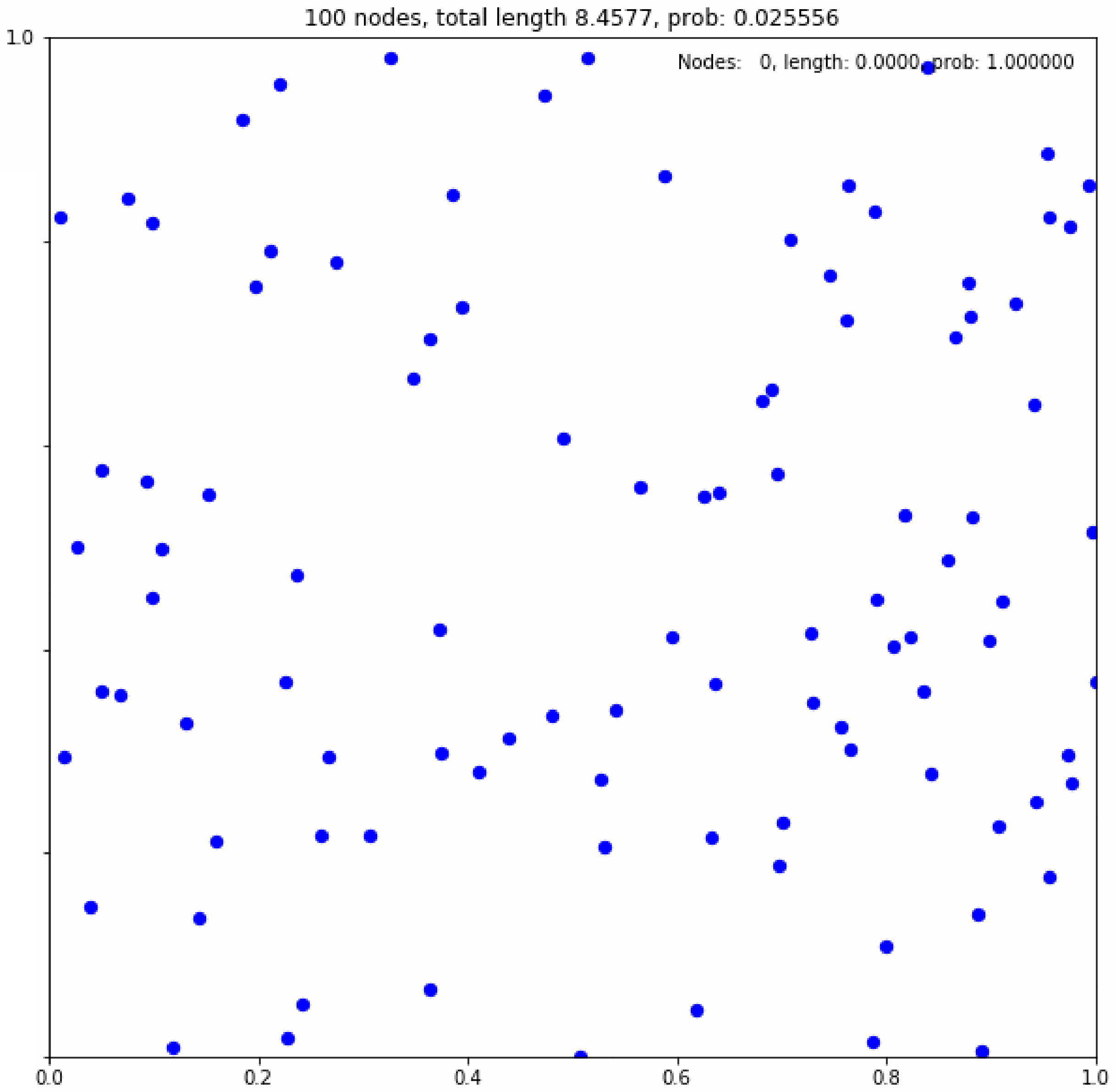
TSP* is (NP-)hard!

Kool et al., 2019

Results



11 routes, total distance 14.06
R0, # 12, c 45 / 50, d 1.32
R1, # 12, c 46 / 50, d 1.75
R2, # 7, c 50 / 50, d 0.84
R3, # 6, c 39 / 50, d 0.62
R4, # 9, c 49 / 50, d 1.58
R5, # 10, c 49 / 50, d 1.44
R6, # 10, c 50 / 50, d 1.34
R7, # 10, c 49 / 50, d 1.48
R8, # 8, c 48 / 50, d 1.49
R9, # 10, c 50 / 50, d 1.09
R10, # 6, c 33 / 50, d 1.09
0.0
0.2
0.4
0.6
0.8
1.0

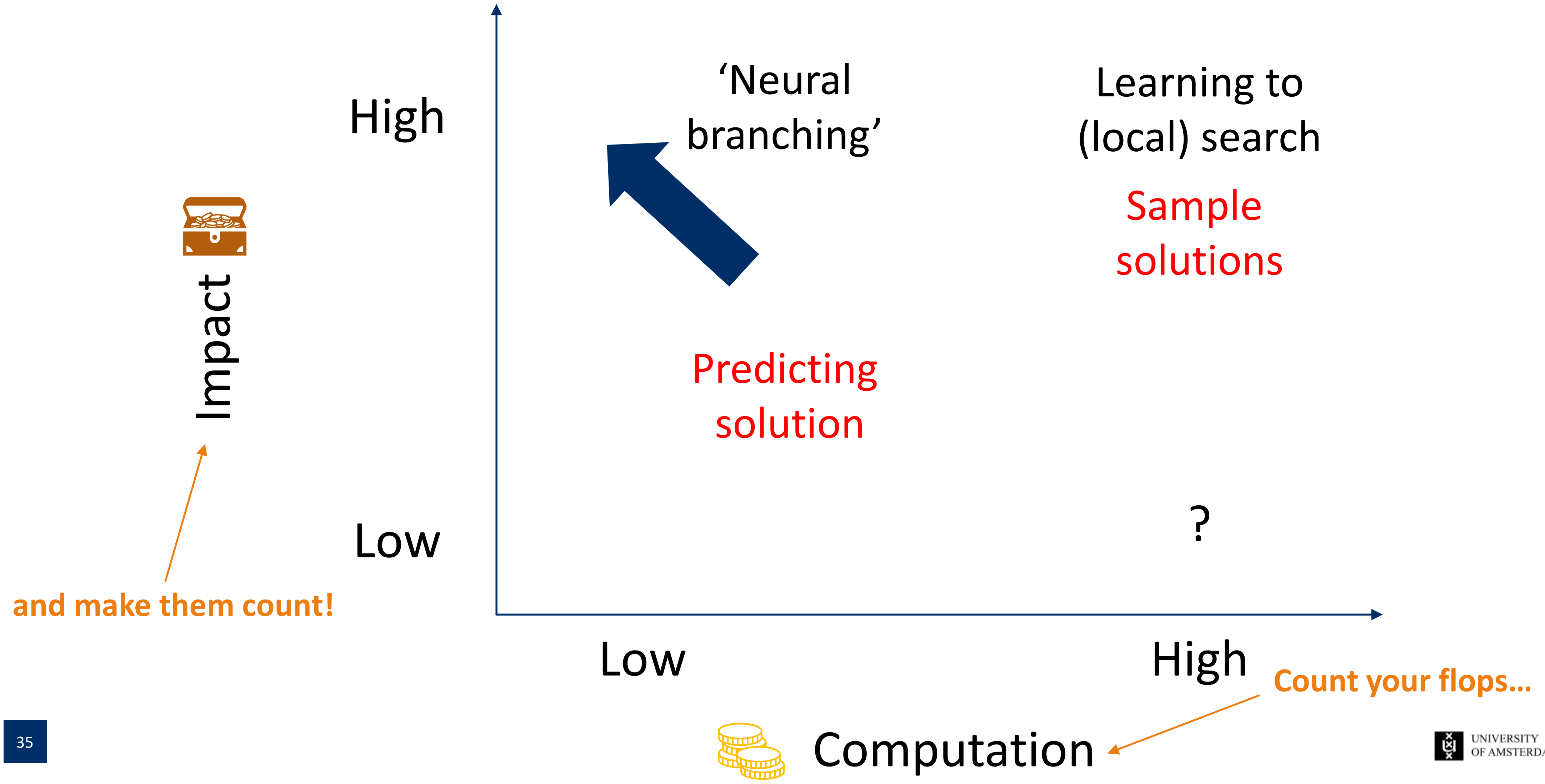


100 nodes, total length 8.4577, prob: 0.025556
Nodes: 0, length: 0.0000, prob: 1.000000
0.0
0.2
0.4
0.6
0.8
1.0

Results Attention Model + Rollout Baseline

- Improves over classical heuristics!
- Improves over prior learned heuristics!
  - Attention Model improves
  - Rollout helps significantly
- Gets close to single-purpose SOTA (20 to 100 nodes)!
  - TSP 0.34% to 4.53% (greedy)
  - TSP 0.08% to 2.26% (best of 1280 samples)

Impact vs. computation (of your neural network)



UNIVERSITY OF AMSTERDAM ORTEC

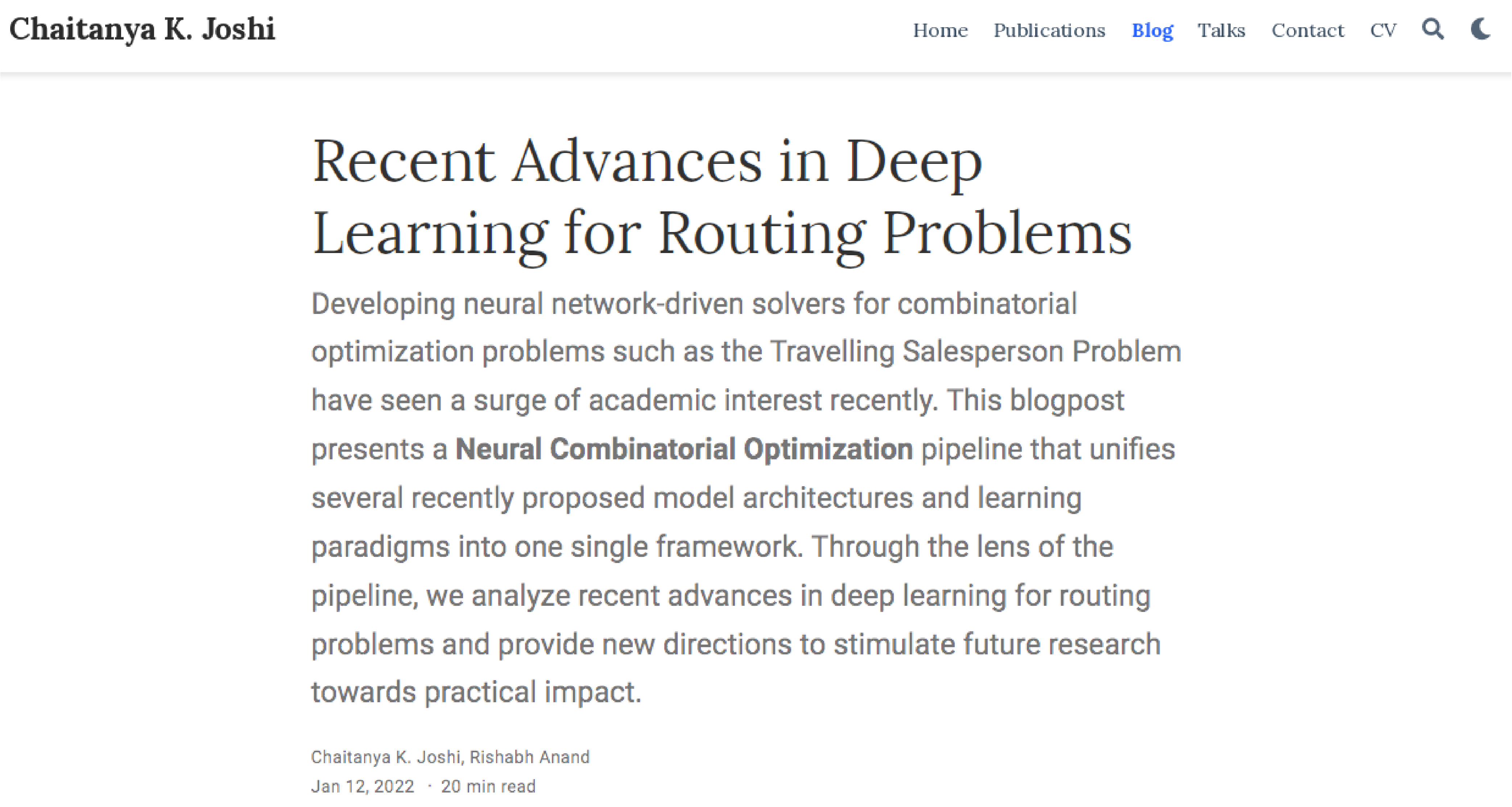
"

Let’s try a different approach

"

- Chaitanya K. Joshi

ORTEC

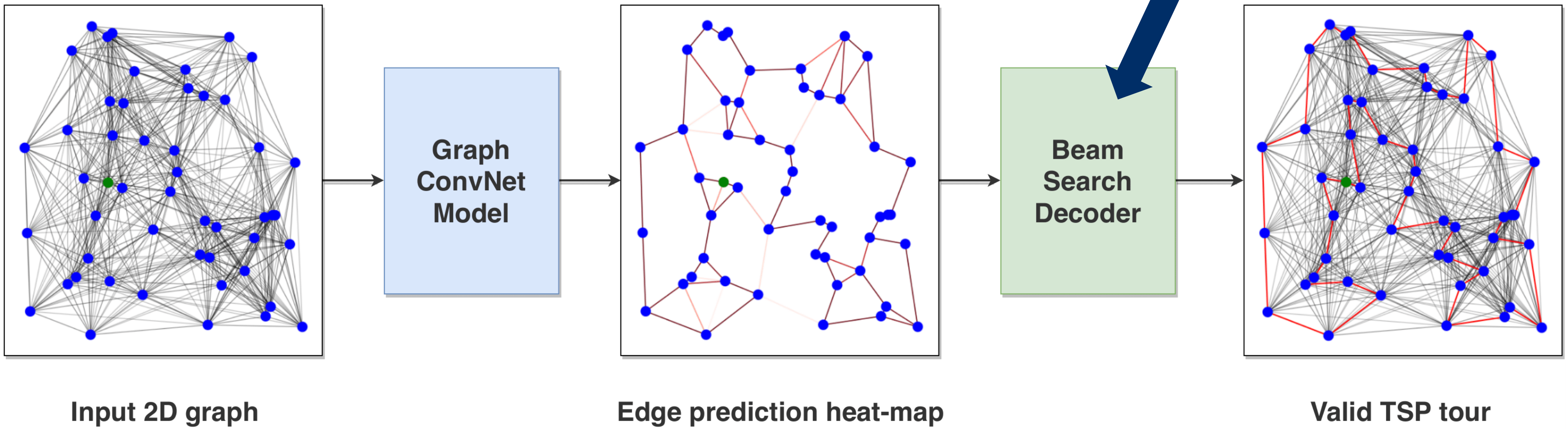


https://www.chaitjo.com/post/deep-learning-for-routing-problems/

ORTEC

Non-autoregressive approach (Joshi et al., 2019)

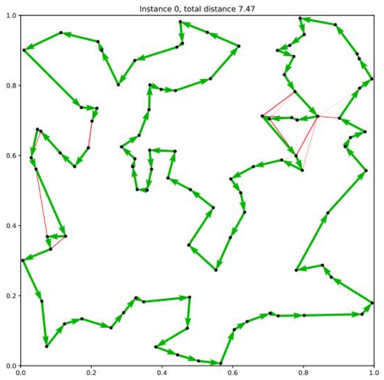
We can do better!



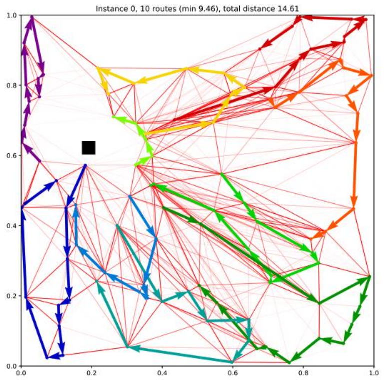
Picture by Joshi et al., 2019

More efficient AND better results!

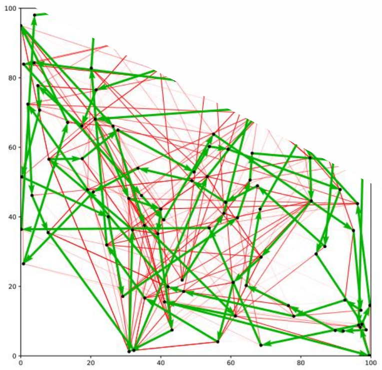
Bringing us to DPDP



(a) Travelling Salesman Problem



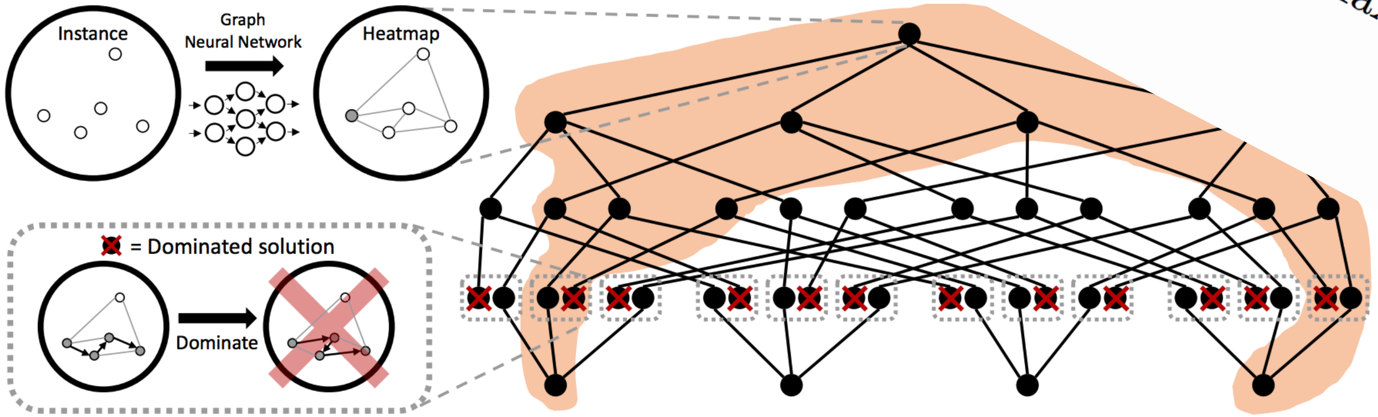
(b) Vehicle Routing Problem



(c) TSP with Time Windows

Deep Policy Dynamic Programming
for Vehicle Routing Problems

Wouter Kool*[1,2], Herke van Hoof[1], Joaquim Gromicho[1,2] and Max Welling[1]
[1]University of Amsterdam
[2]ORTEC



TL;DR

"

DPDP is a flexible framework for vehicle routing problems…

…that restricts a dynamic program to promising parts of the state space…

…using a heatmap of promising edges predicted by a neural network!

"

Held-Karp DP for TSP (Held & Karp, 1962; Bellman, 1962)

Cost/distance from j to i

$$C(S, i) = \min_{j \in S \setminus \{i\}} C(S \setminus \{i\}, j) + c_{ji}$$

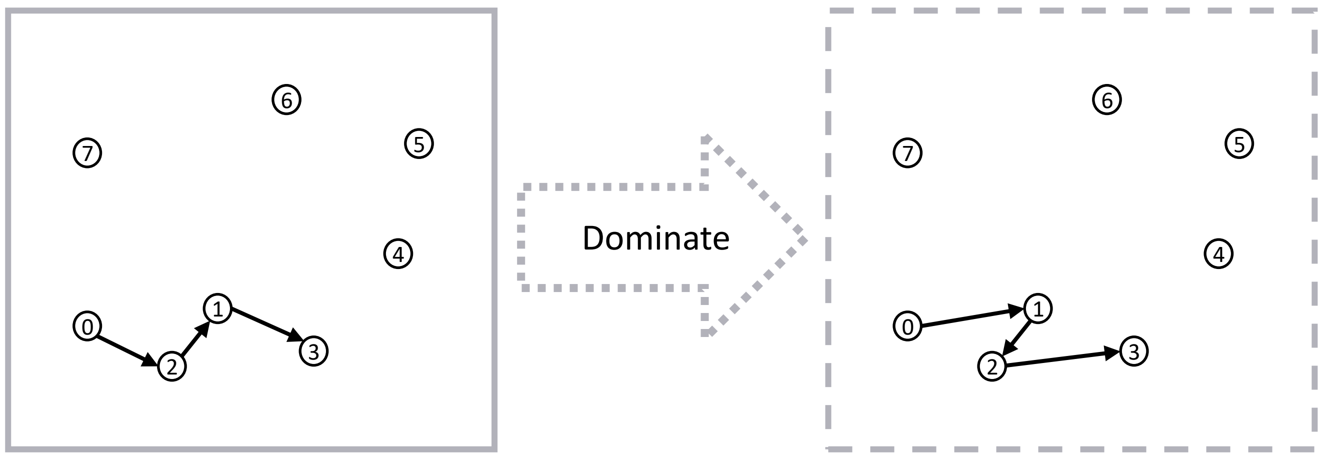
Minimum cost to go from 0 to i visiting all nodes in S

Set of visited nodes

Current node

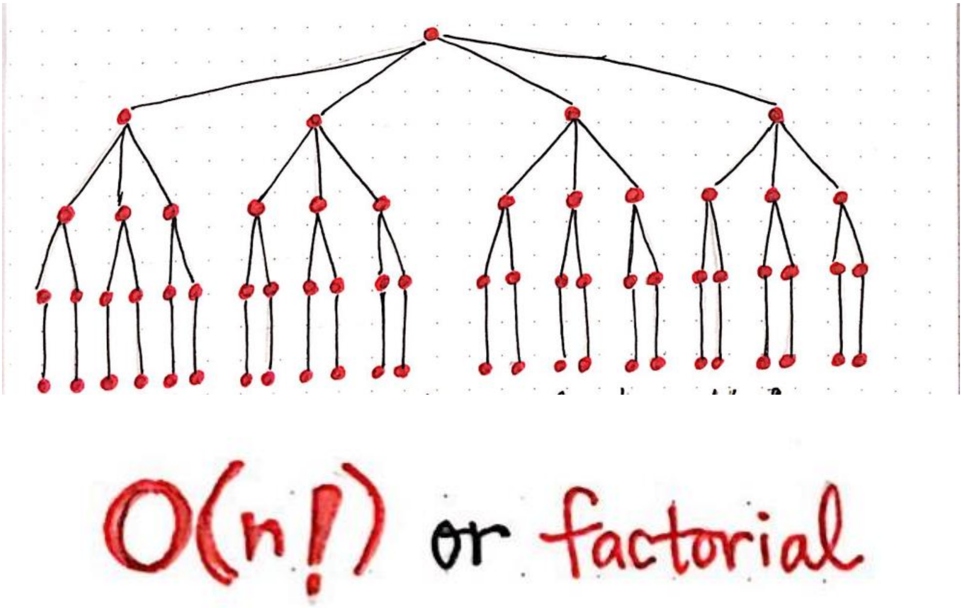
DP state

Find best solution for each DP state

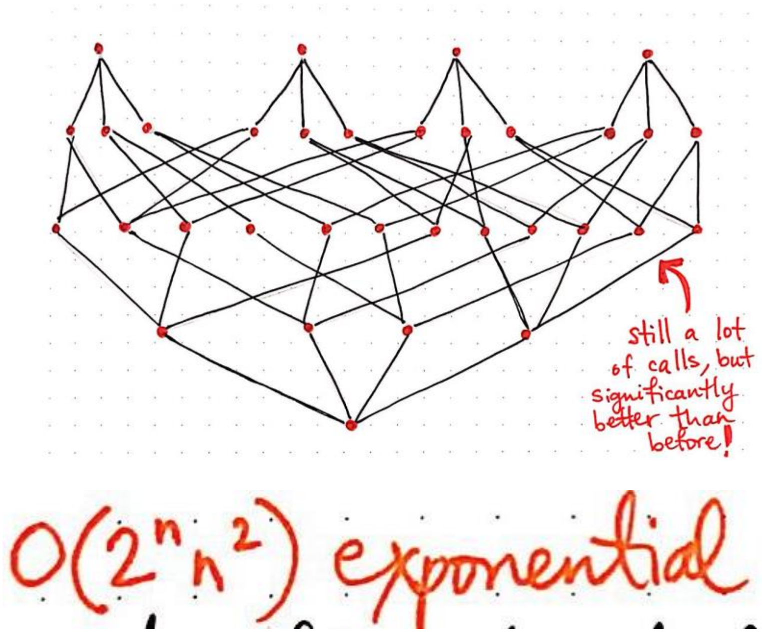


Held-Karp DP for TSP (Held & Karp, 1962; Bellman, 1962)

Brute-force (forward view)



DP (top-down or backward view)

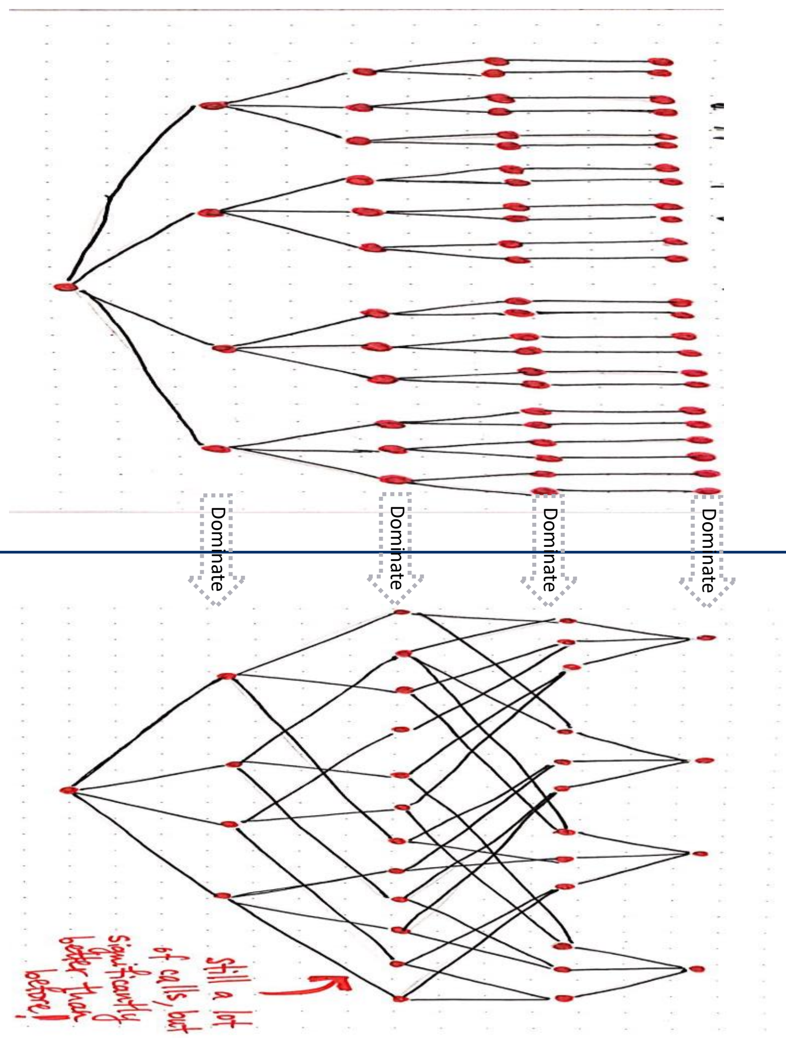


Artwork by Vaidehi Joshi, https://medium.com/basecs/speeding-up-the-traveling-salesman-using-dynamic-programming-b76d7552e8dd

Held-Karp DP for TSP (Held & Karp, 1962; Bellman, 1962)

Brute-force

$O(n!)$ or factorial



Forward view

Still impractical!

DP

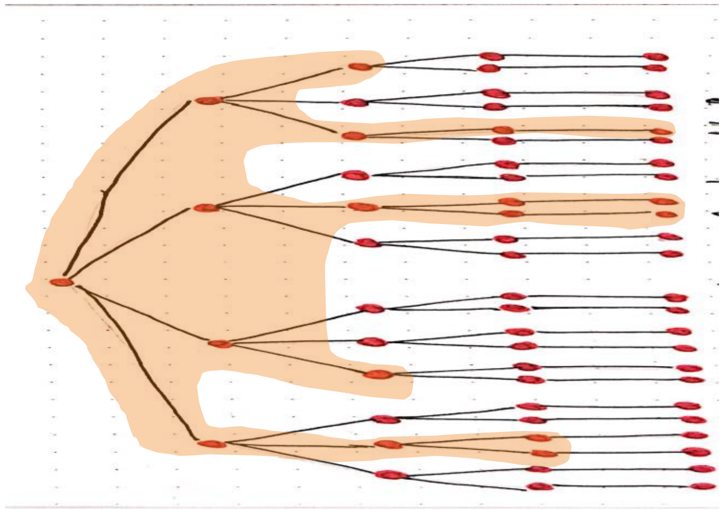
$O(2^n n^2)$ exponential

UNIVERSITY OF AMSTERDAM ORTEC

Held-Karp DP for TSP (Held & Karp, 1962; Bellman, 1962)

Beam search

$O(Bn)$ or *linear*

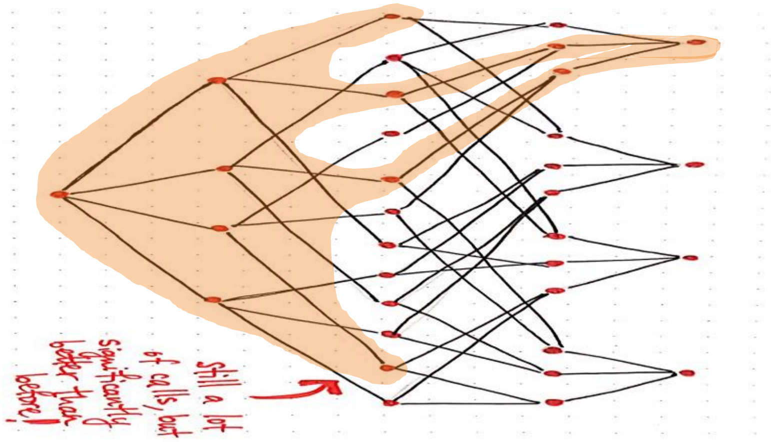


We need a *good policy* to restrict the search space!

Forward view

Restricted DP

$O(Bn)$ or *linear*



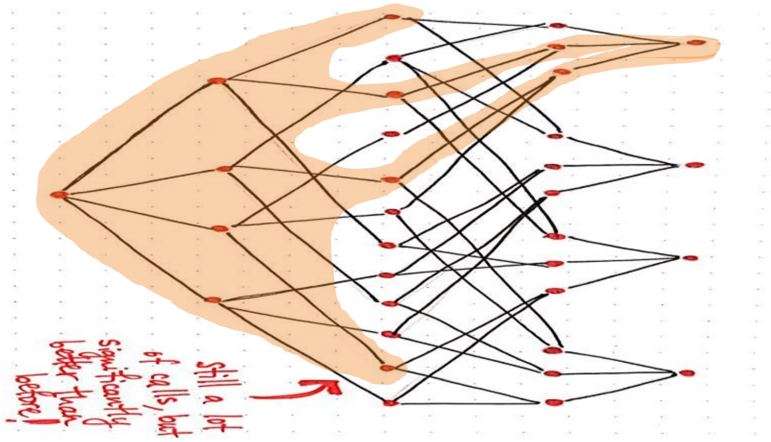
Malandraki & Dial, 1996
Gromicho et al., 2012

Deep Policy Dynamic Programming (DPDP)

https://arxiv.org/abs/2102.11756

DPDP

$O(Bn)$ or *linear*



For each iteration
- Expand solutions
- Remove dominated solutions
- Select top B according to *policy*
- Repeat

Forward view

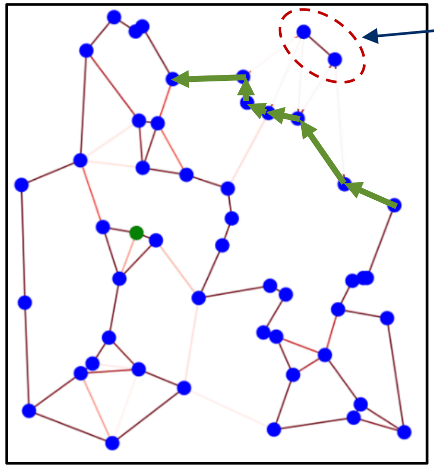
Policy: select top B solutions that maximize the score.

$$\text{SCORE} = \text{HEAT} + \text{POTENTIAL}$$

Heat of edges in solution

Estimate for unvisited nodes
based on remaining edges

Potential avoids 'skipped nodes'

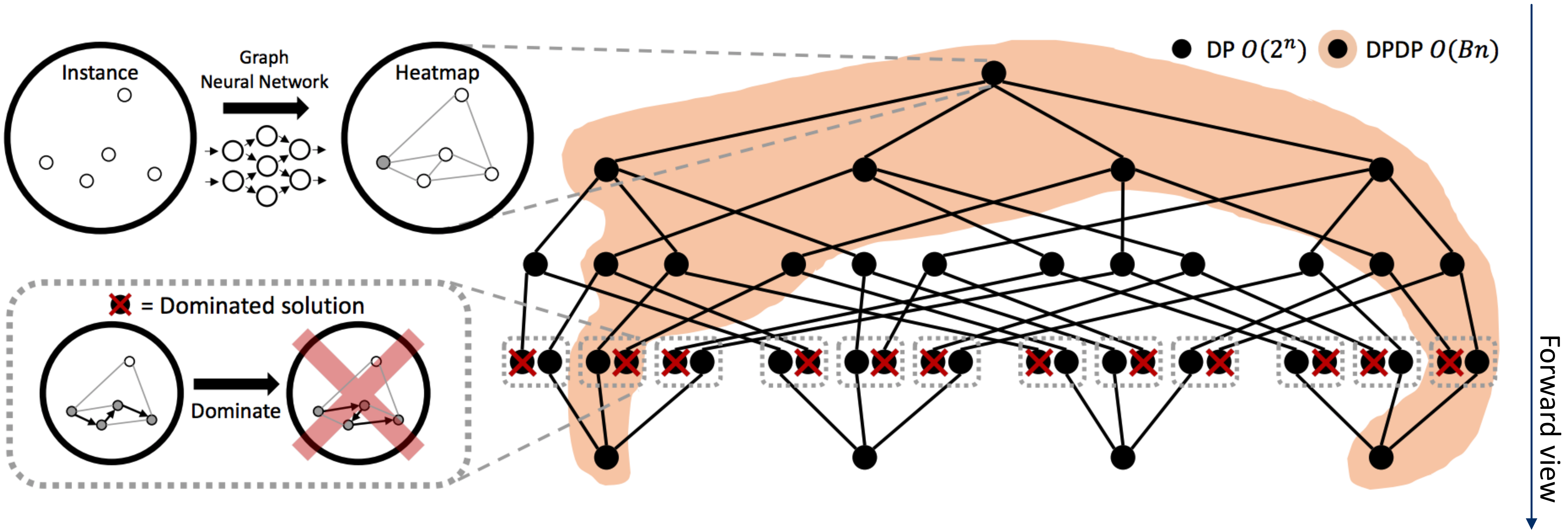


Heat
$h_{ij} \in (0,1)$

Picture by Joshi et al., 2019

Deep Policy Dynamic Programming (DPDP)

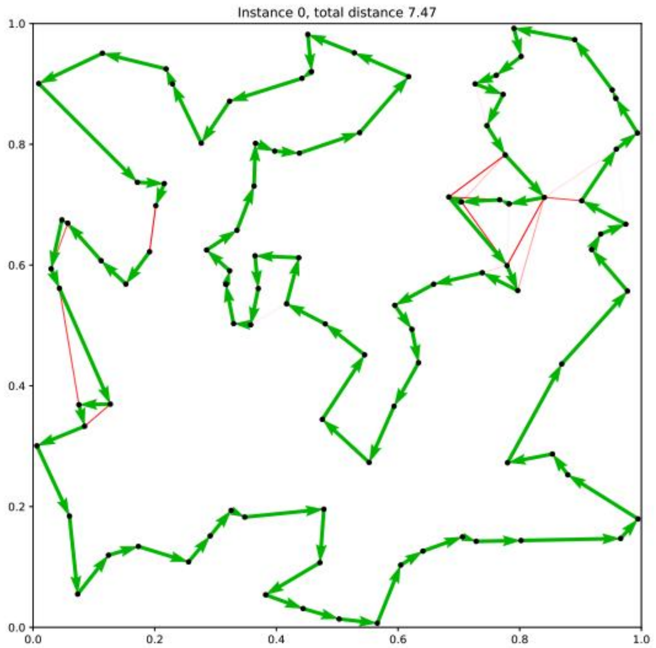
https://arxiv.org/abs/2102.11756



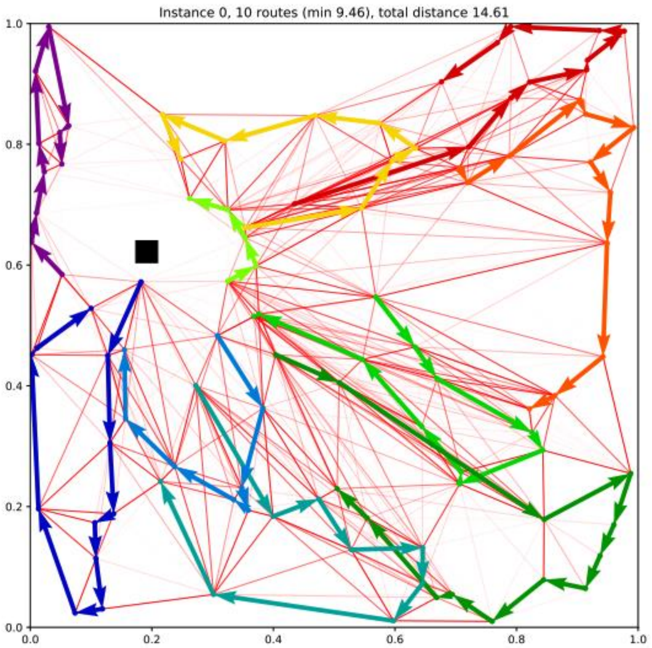
$O(Bn)$ or *linear*

For each iteration
- Expand solutions
- Remove dominated solutions
- Select top B according to *policy*
- Repeat

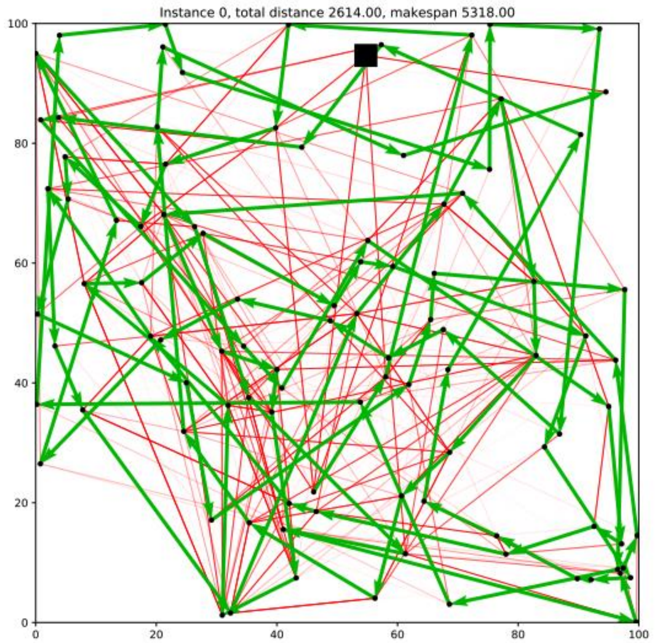
Experiments



(a) Travelling Salesman Problem



(b) Vehicle Routing Problem



(c) TSP with Time Windows

Results (TSP/VRP)

Table 1: Mean cost, gap and *total time* to solve 10000 TSP/VRP test instances.

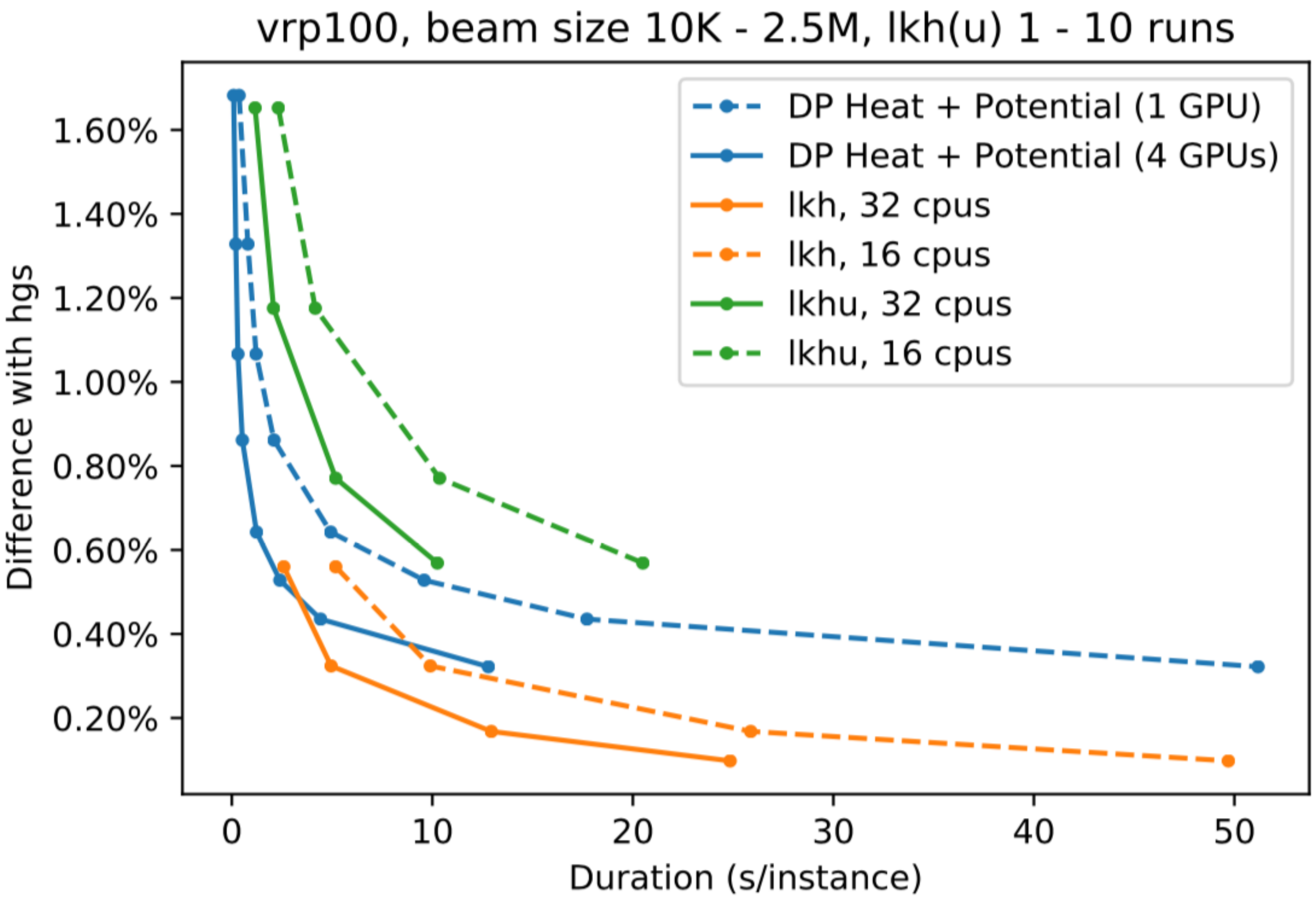
Problem	TSP100			VRP100		
Method	Cost	Gap	Time	Cost	Gap	Time
Concorde [2]	7.765	0.000 %	6m			
Hybrid Genetic Search [63, 62]				15.563	0.000 %	6h11m
Gurobi [24]	7.776	0.151 %	31m			
LKH [27]	7.765	0.000 %	42m	15.647	0.536 %	12h57m
GNN Heatmap + Beam Search [32]	7.87	1.39 %	40m			
Learning 2-opt heuristics [11]	7.83	0.87 %	41m			
Merged GNN Heatmap + MCTS [19]	7.764*	0.04 %	4m + 11m			
Attention Model + Sampling [36]	7.94	2.26 %	1h	16.23	4.28 %	2h
Step-wise Attention Model [67]	8.01	3.20 %	29s	16.49	5.96 %	39s
Attn. Model + Coll. Policies [34]	7.81	0.54 %	12h	15.98	2.68 %	5h
Learning improv. heuristics [66]	7.87	1.42 %	2h	16.03	3.00 %	5h
Dual-Aspect Coll. Transformer [45]	7.77	0.09 %	5h	15.71	0.94 %	9h
Attention Model + POMO [37]	7.77	0.14 %	1m	15.76	1.26 %	2m
NeuRewriter [9]				16.10	3.45 %	1h
Dynamic Attn. Model + 2-opt [54]				16.27	4.54 %	6h
Neur. Lrg. Neighb. Search [30]				15.99	2.74 %	1h
Learn to improve [43]				15.57*	-	4000h
DPDP 10K	7.765	0.009 %	10m + 16m	15.830	1.713 %	10m + 50m
DPDP 100K	7.765	0.004 %	10m + 2h35m	15.694	0.843 %	10m + 5h48m
DPDP 1M				15.627	0.409 %	10m + 48h27m

Results (TSP with time windows)

Table 3: Mean cost, gap and *total time* to solve TSPTW100 instances.

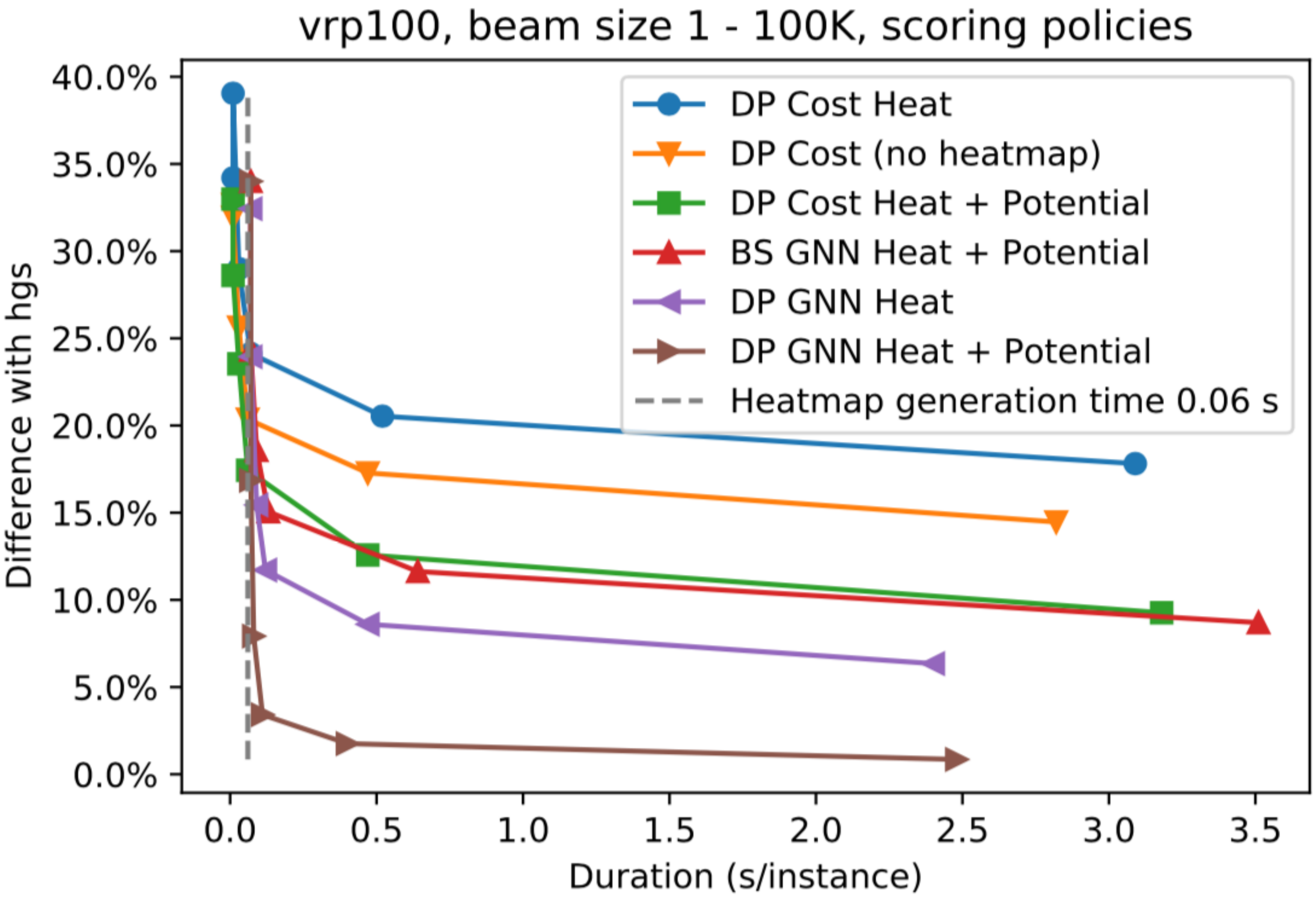
Problem Method	Small time windows [8] (100 inst.) Cost	Gap	Fail	Time	Large time windows [12] (10K inst.) Cost	Gap	Fail	Time
GVNS 30x [12]	5129.58	0.000 %		7s	2432.112	0.000 %		37m15s
GVNS 1x [12]	5129.58	0.000 %		<1s	2457.974	1.063 %		1m4s
LKH 1x [27]	5130.32	0.014 %	1.00 %	5m48s	2431.404	-0.029 %		34h58m
BaB-DQN* [8]	5130.51	0.018 %		25h				
ILDS-DQN* [8]	5130.45	0.017 %		25h				
DPDP 10K	5129.58	0.000 %		6s + 1s	2431.143	-0.040 %		10m + 8m7s
DPDP 100K	5129.58	0.000 %		6s + 1s	2430.880	- 0.051 %		10m + 1h16m

Quality vs. computation



UNIVERSITY OF AMSTERDAM ORTEC

Ablations


vrp100, beam size 1 - 100K, scoring policies
Difference with hgs
Duration (s/instance)
40.0%
35.0%
30.0%
25.0%
20.0%
15.0%
10.0%
5.0%
0.0%
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
DP Cost Heat
DP Cost (no heatmap)
DP Cost Heat + Potential
BS GNN Heat + Potential
DP GNN Heat
DP GNN Heat + Potential
Heatmap generation time 0.06 s

Deep Policy Dynamic Programming (DPDP)

https://arxiv.org/abs/2102.11756

- DP is flexible framework for many VRP variants e.g. time windows
- Suitable for GPU implementation
- Natural trade-off compute vs. performance -> asymptotically optimal
- Supervised training based on example solutions
- Test time: only evaluate NN once!

Hybrid Genetic Search for the DIMACS VRPTW Challenge

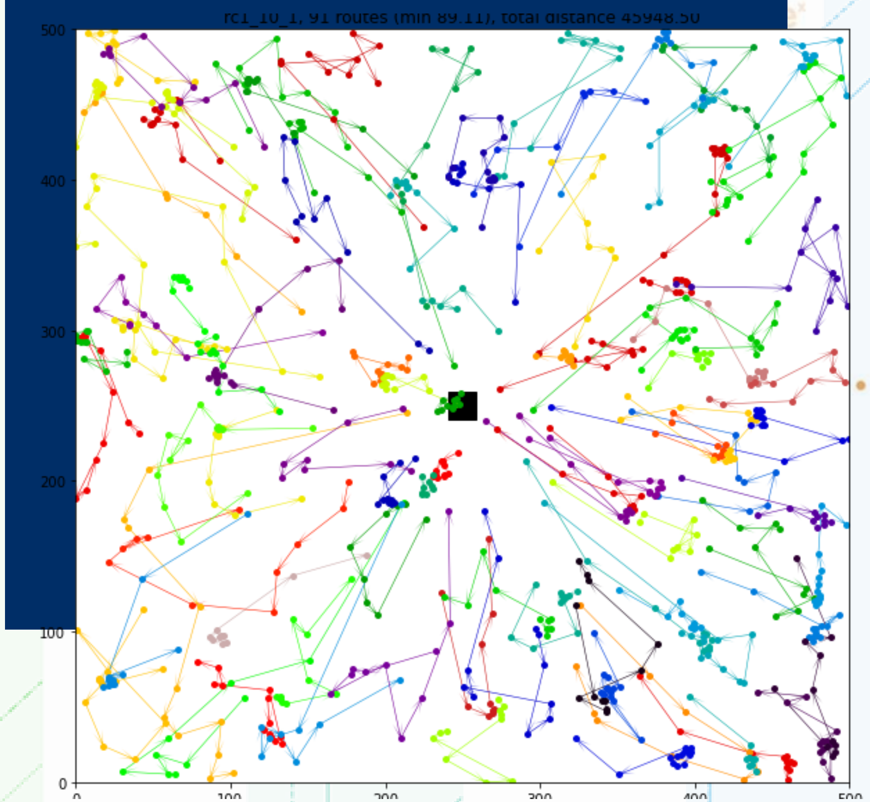
Wouter Kool & co(lleagues). -- original HGS by Thibaut Vidal
Joep Olde Juninck
Ernst Roos
Kamiel Cornelissen
Pim Agterberg
Jelke van Hoorn
Thomas Visser

Public

ORTEC

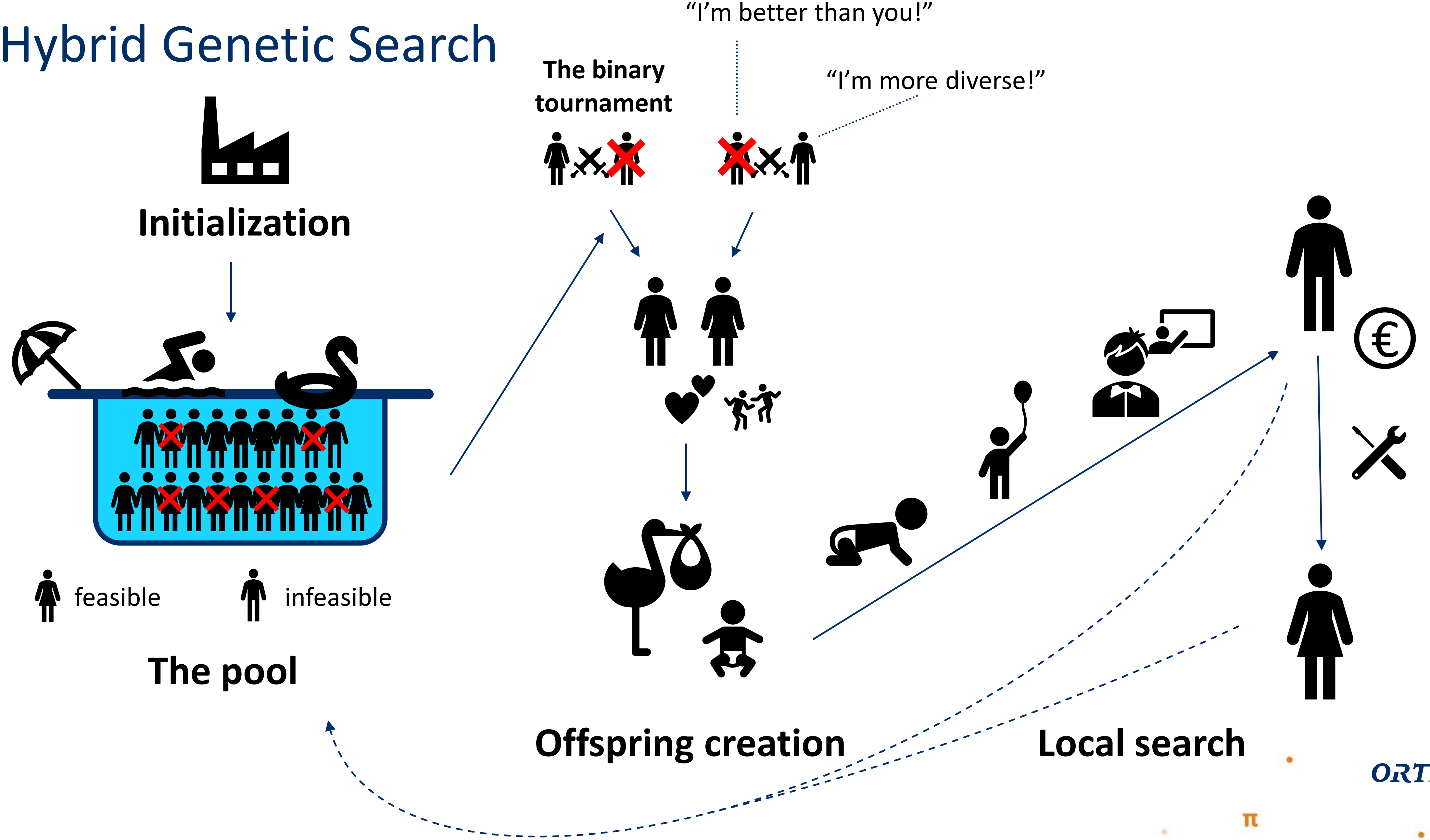
In the DIMACS challenge

- Vehicle routing problem with capacities
- Every customer must be served within a time window
- DIMACS variant: Minimize distance only (not vehicles)

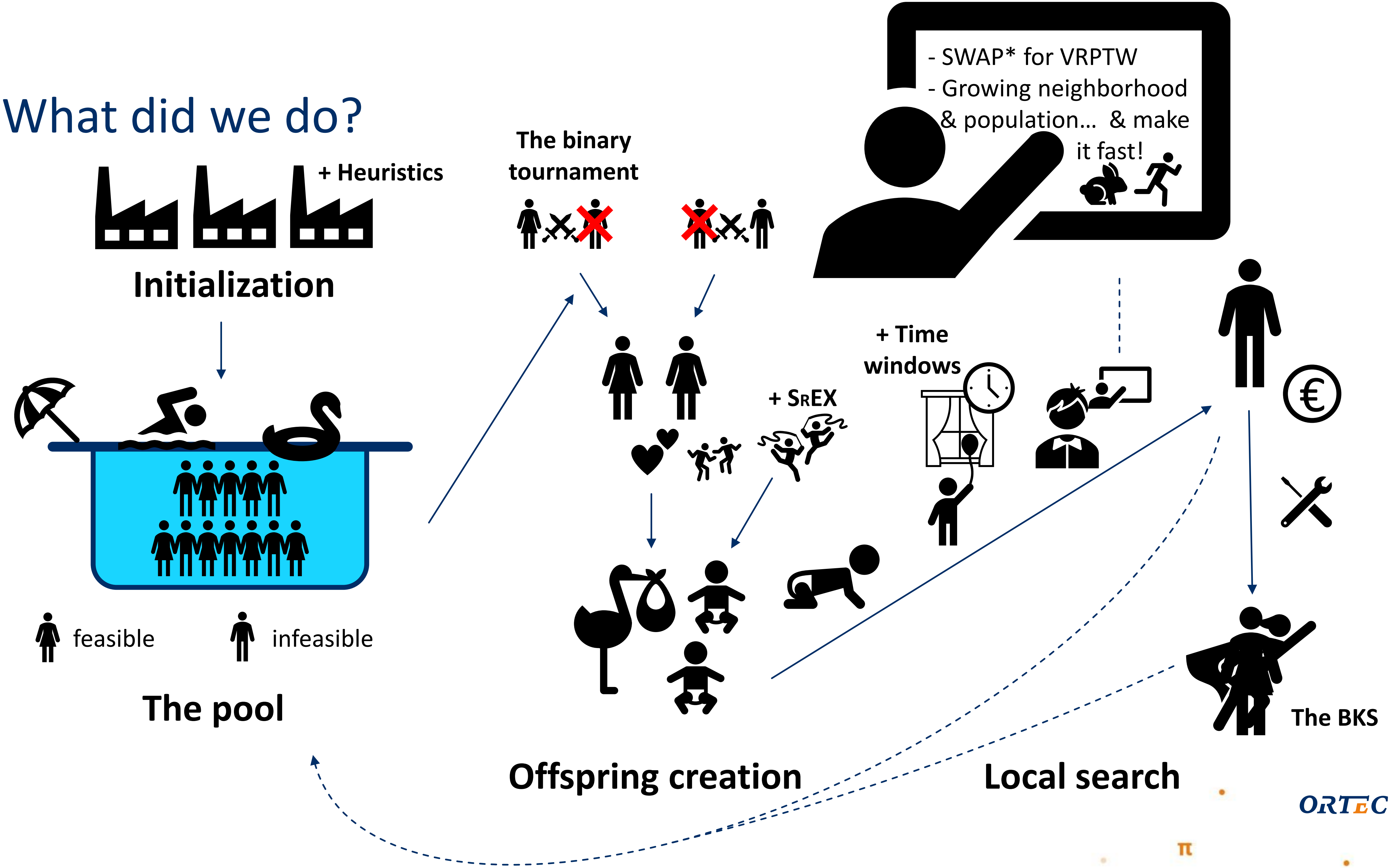


ORTEC

Hybrid Genetic Search



What did we do?



ORTEC

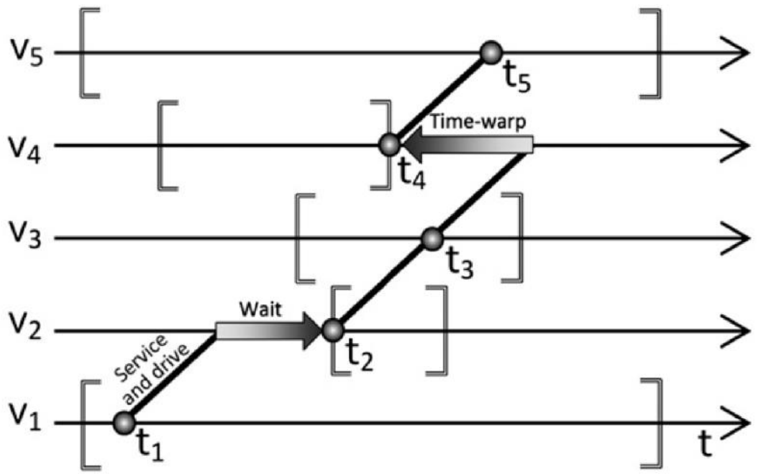


Fig. 1. Illustration of waiting times and time warps.

Proposition 1 *(Concatenation of two sequences). Let* $\sigma = (\sigma_i, \ldots, \sigma_j)$ *and* $\sigma' = (\sigma'_{i'}, \ldots, \sigma'_{j'})$ *be two subsequences of visits. The concatenated subsequence* $\sigma \oplus \sigma'$ *is characterized by the following data:*

$$D(\sigma \oplus \sigma') = D(\sigma) + D(\sigma') + \delta_{\sigma_j \sigma'_{i'}} + \Delta_{WT} \quad (5)$$

$$TW(\sigma \oplus \sigma') = TW(\sigma) + TW(\sigma') + \Delta_{TW} \quad (6)$$

$$E(\sigma \oplus \sigma') = \max\{E(\sigma') - \Delta, E(\sigma)\} - \Delta_{WT} \quad (7)$$

$$L(\sigma \oplus \sigma') = \min\{L(\sigma') - \Delta, L(\sigma)\} + \Delta_{TW} \quad (8)$$

$$C(\sigma \oplus \sigma') = C(\sigma) + C(\sigma') + c_{\sigma_j \sigma'_{i'}} \quad (9)$$

$$Q(\sigma \oplus \sigma') = Q(\sigma) + Q(\sigma') \quad (10)$$

where $\Delta = D(\sigma) - TW(\sigma) + \delta_{\sigma_j \sigma'_{i'}}$, $\Delta_{WT} = \max\{E(\sigma') - \Delta - L(\sigma), 0\}$ *and* $\Delta_{TW} = \max\{E(\sigma) + \Delta - L(\sigma'), 0\}$.

Source: Vidal et al. 2012

Supporting time windows

- Use time-warp principle
- Cache computation for prefix and postfix of routes
- Use two-level hierarchy for fast queries in middle of route
- Penalty booster: increase penalty by 100% if no feasible solution found

Offspring generation

Selective Route Exchange (SREX)

540   Y. Nagata and S. Kobayashi

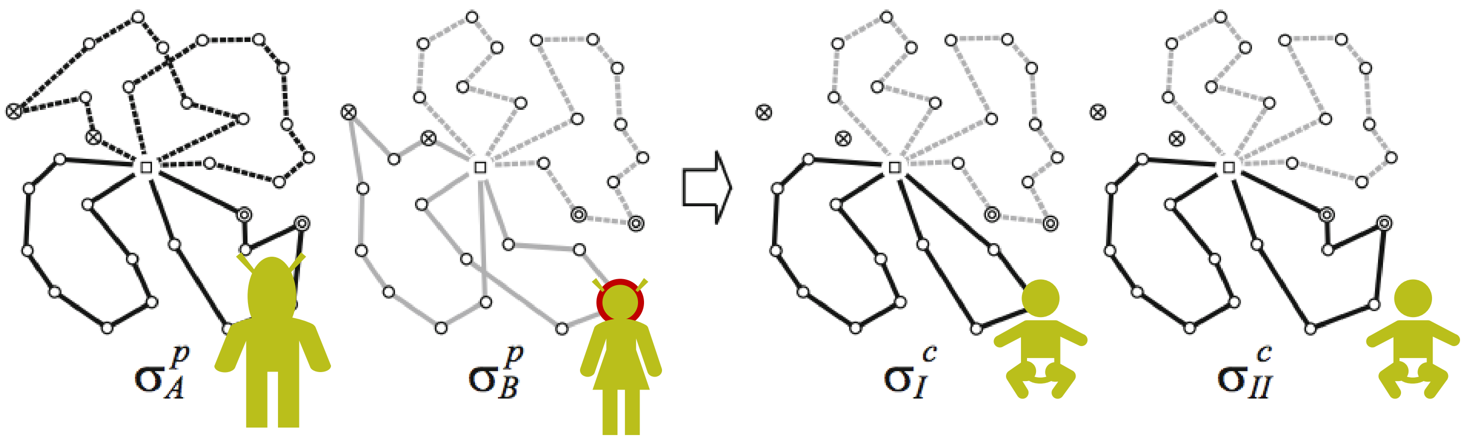


Fig. 1. Illustration of the SREX. σ_A^p and σ_B^p are parents. Routes S_A and S_B are represented as dotted lines, customer nodes $V_{A\setminus B}$ are represented by circles with x-mark, and the customer nodes in $V_{B\setminus A}$ are represented by double circles. σ_I^c and σ_{II}^c are intermediate offspring solutions obtained after Step 2.

Source: Nagata et al. 2010

Ordered Crossover (OX)

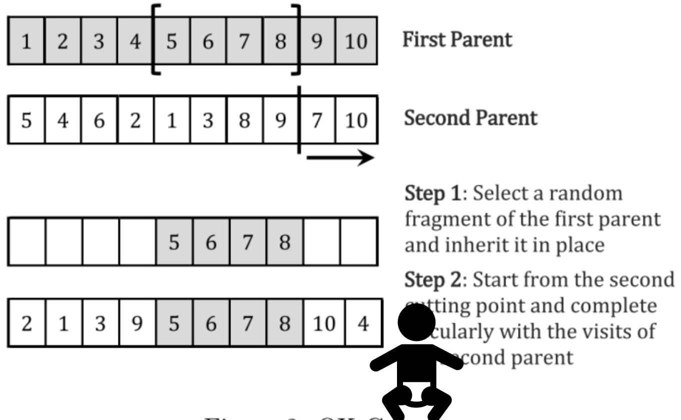


Figure 2: OX Crossover

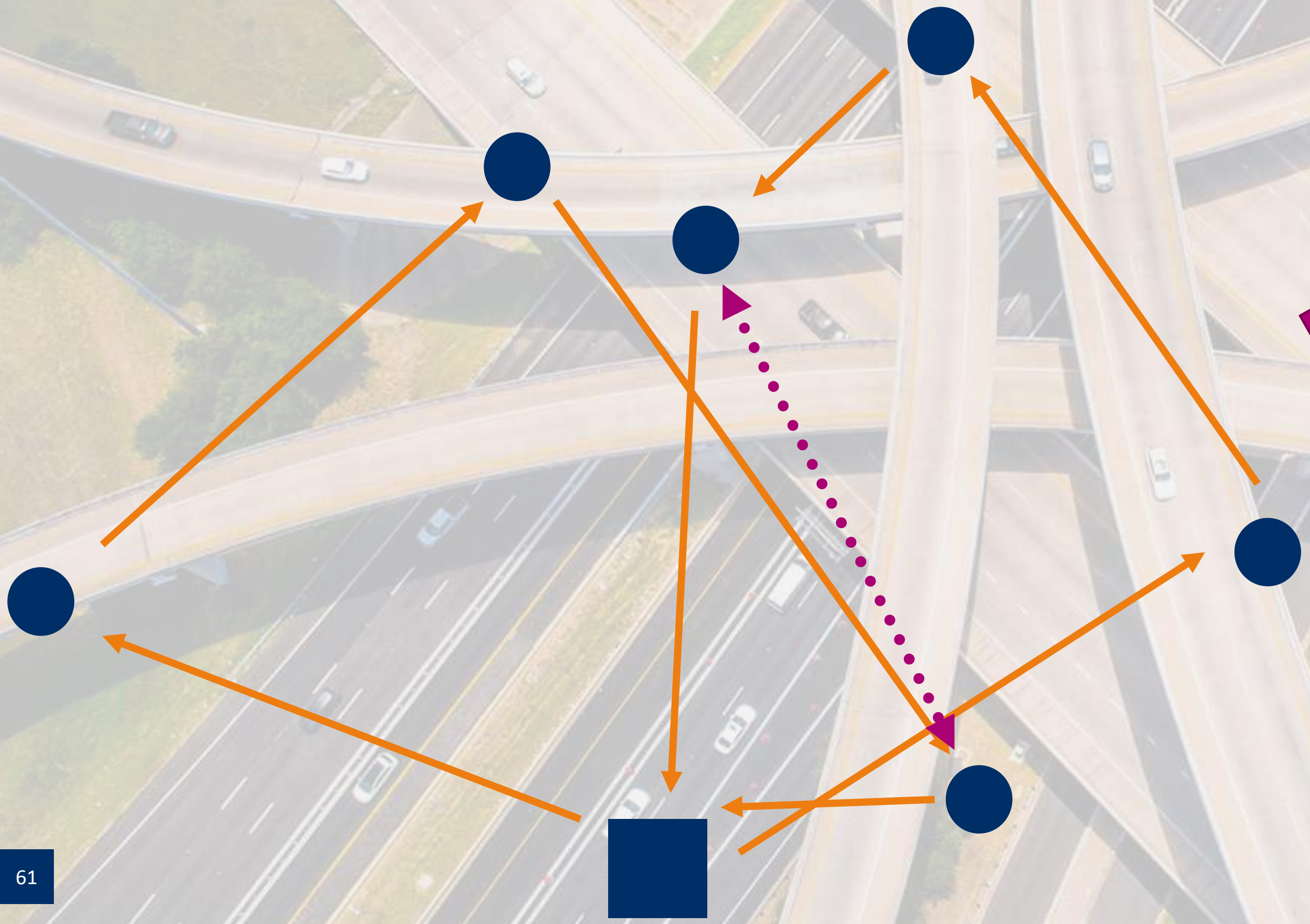
Source: Vidal 2021

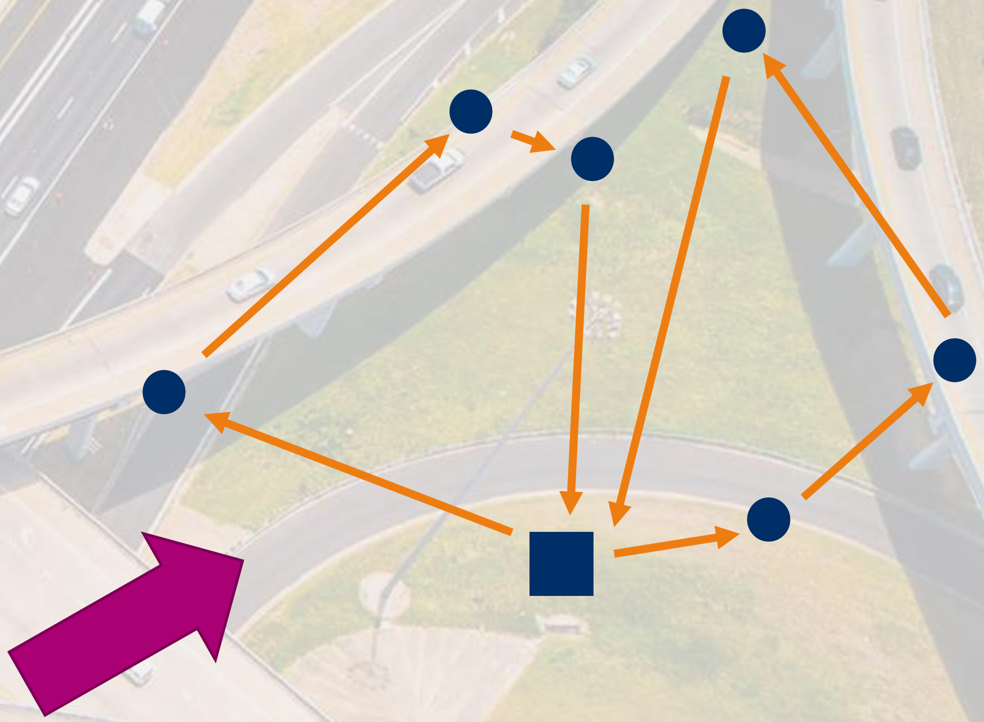
ORTEC

Local search

- SWAP, RELOCATE, 2-OPT, 2-OPT*
- Moves between near neighbors
- Smart ‘pre-checks’
- SWAP*, see next slide

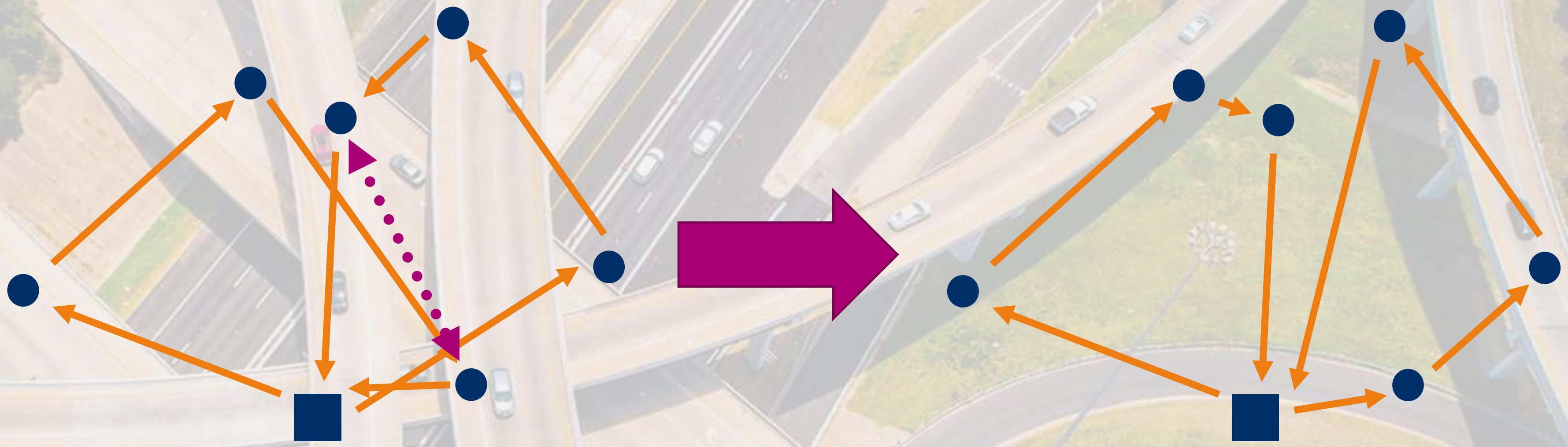
SWAP*





Exchange two nodes, insert at best position in other route

SWAP*



- Cache top 3 insertion positions
- Exact for CVRP
- Approximate for VRPTW

ORTEC

Growing the neighborhood & population

- Every 10K iterations
  - Grow neighborhood by 5
  - Grow population size by 5

- *Slightly different schedule for different instances

Results (in DIMACS)

Dataset	C1	C2	R1	R2	RC1	RC2	Mean
Solomon	0,000%	0,000%	-0,003%	0,000%	0,000%	0,000%	0,000%
GH200	0,000%	0,004%	0,001%	0,009%	0,016%	0,026%	0,009%
GH400	0,000%	0,000%	-0,009%	0,028%	-0,030%	-0,050%	-0,010%
GH600	-0,014%	0,022%	0,047%	-0,022%	-0,012%	-0,123%	-0,017%
GH800	0,030%	-0,018%	0,147%	0,090%	0,112%	-0,222%	0,023%
GH1000	0,123%	-0,013%	0,174%	-0,090%	0,094%	-0,158%	0,022%
Mean	0,023%	-0,001%	0,060%	0,002%	0,030%	-0,088%	0,004%

(a) Gap to reference solution

ORTEC

The next challenge...

Goal: bring together

Operations Research and *Machine Learning*

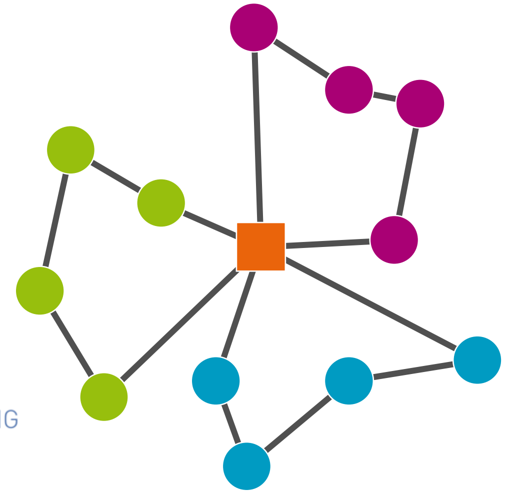
to solve a *static* and *dynamic* VRP with time windows!

EURO Meets NeurIPS 2022 Vehicle Routing Competition

EURO
THE ASSOCIATION OF EUROPEAN OPERATIONAL RESEARCH SOCIETIES

EWG DSO
DATA SCIENCE MEETS OPTIMIZATION

EWG VEROLOG
VEHICLE ROUTING AND LOGISTICS



NEURAL INFORMATION PROCESSING SYSTEMS

ORTEC

EAISI EINDHOVEN AI SYSTEMS INSTITUTE

TU/e

More info? https://euro-neurips-vrp-2022.challenges.ortec.com/

The next challenge…

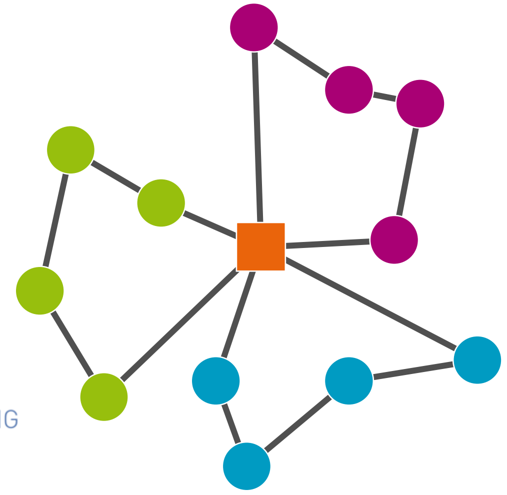
EURO Meets NeurIPS 2022 Vehicle Routing Competition

- Real data!
- Static variant: DIMACS VRPTW
- Dynamic variant: new requests arrive during the day. Challenge: which orders to dispatch now or delay?

EURO
THE ASSOCIATION OF EUROPEAN OPERATIONAL RESEARCH SOCIETIES

EWG DSO
DATA SCIENCE MEETS OPTIMIZATION

EWG VEROLOG
VEHICLE ROUTING AND LOGISTICS



NEURAL INFORMATION PROCESSING SYSTEMS

ORTEC

EAISI EINDHOVEN AI SYSTEMS INSTITUTE

TU/e

More info? https://euro-neurips-vrp-2022.challenges.ortec.com/

Why?

Operations Research (OR)

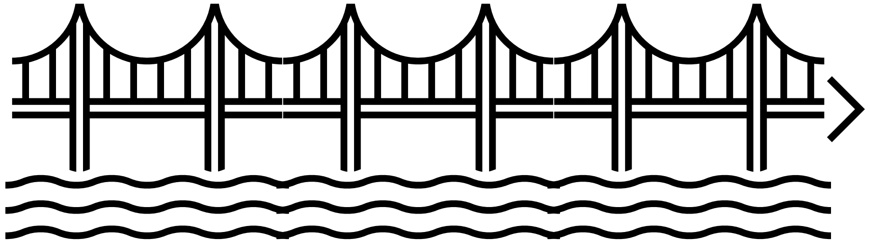
- OR researchers also start using ML
- but often 'simple' techniques
- leaving deep learning potential on the table!

Machine Learning (ML)

- ML research for VRP is hot…
- but unable to outperform SOTA OR techniques
- and fair/independent comparison is lacking!

To get the best results, we must *bridge the gap* between OR and ML

OR

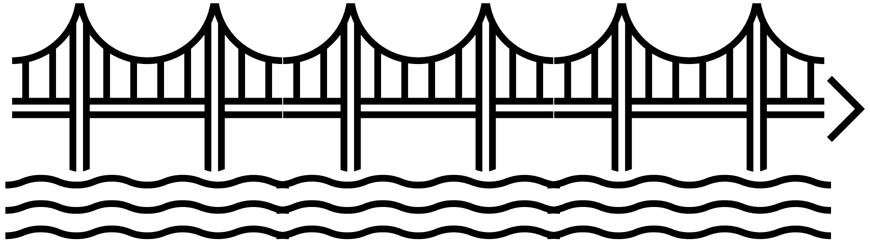


ML

NEURAL INFORMATION PROCESSING SYSTEMS
2022

How?

3-6 JULY
EURO 2022
AALTO UNIVERSITY
ESPOO FINLAND



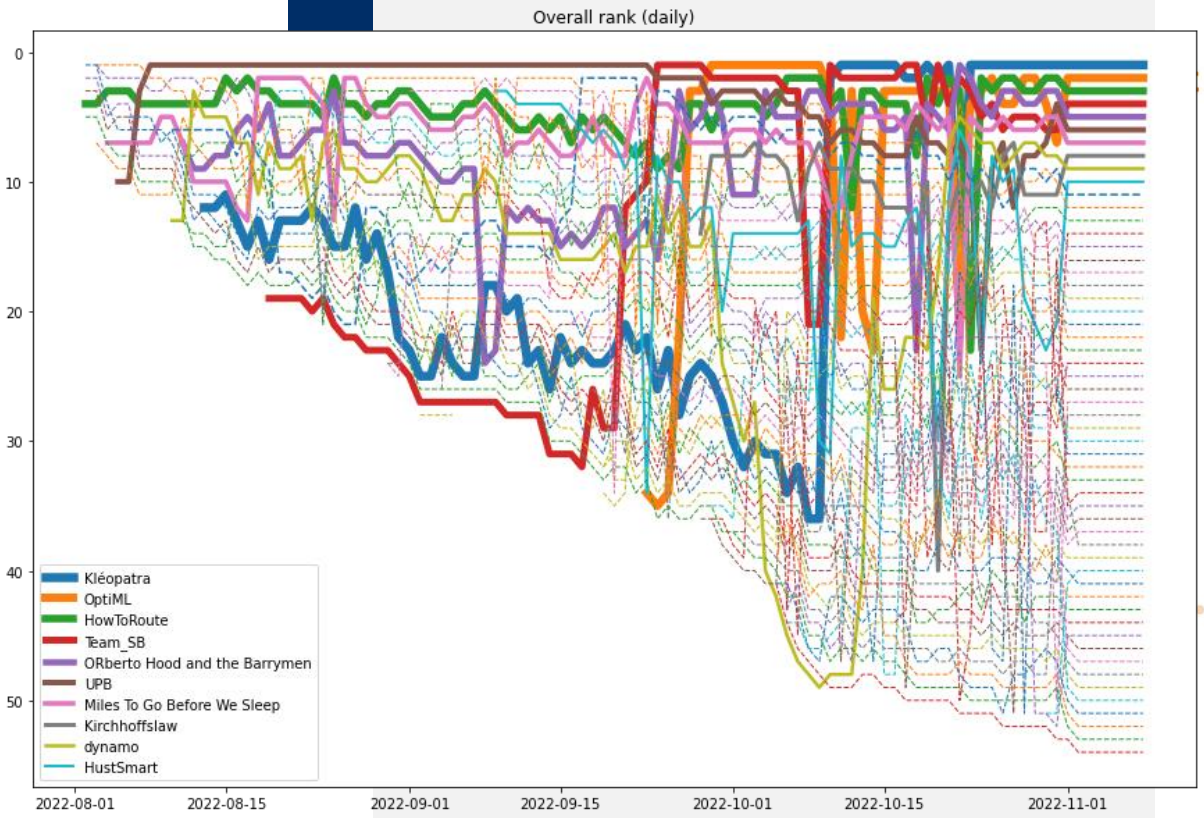
NEURAL INFORMATION PROCESSING SYSTEMS

- Starting the competition at EURO (OR) and end it at NeurIPS (ML) 2022
- Bringing together participants from OR and ML community
- Adding real data from US-based grocery delivery service
- Providing a SOTA VRPTW baseline (Hybrid Genetic Search)
- Encouraging ML approaches by GPU availability and dynamic variant
- Code submission + real time leaderboard for engagement

Results

- 150 teams registered
- 50 teams submitted
- 800 submissions

- 1: Kleopatra (TU Munich)
- 2: OptiML (VU Amsterdam/ Groningen Univ.)
- 3: Team_sb (Samsung/Bielefeld univ.)



ORTEC

README.md

PyVRP

pypi package 0.3.0 | CI passing | docs passing | codecov 94%

The `pyvrp` package is an open-source, state-of-the-art vehicle routing problem solver.

`pyvrp` may be installed in the usual way as

```
pip install pyvrp
```

This also resolves the few core dependencies `pyvrp` has. The documentation is available here.

> If you are new to metaheuristics or vehicle routing, you might benefit from reading the introduction to HGS for VRP page.

Lab assignment: learning within HGS

https://colab.research.google.com/drive/1n4l0qiL0IGQBi_Scyptn72FNzzScCHHN?usp=sharing

Ideas for learning

- Learn when/how to grow (/shrink?) neighborhood/population
- Learn which parents to ‘do the dance’
- Learn which neighbors and/or moves to consider in local search
- Etc.


OPTIMIZED BY ORTEC
ORTEC.COM