Neural Network Heuristics for Classical Planning:
A Study of Hyperparameter Space

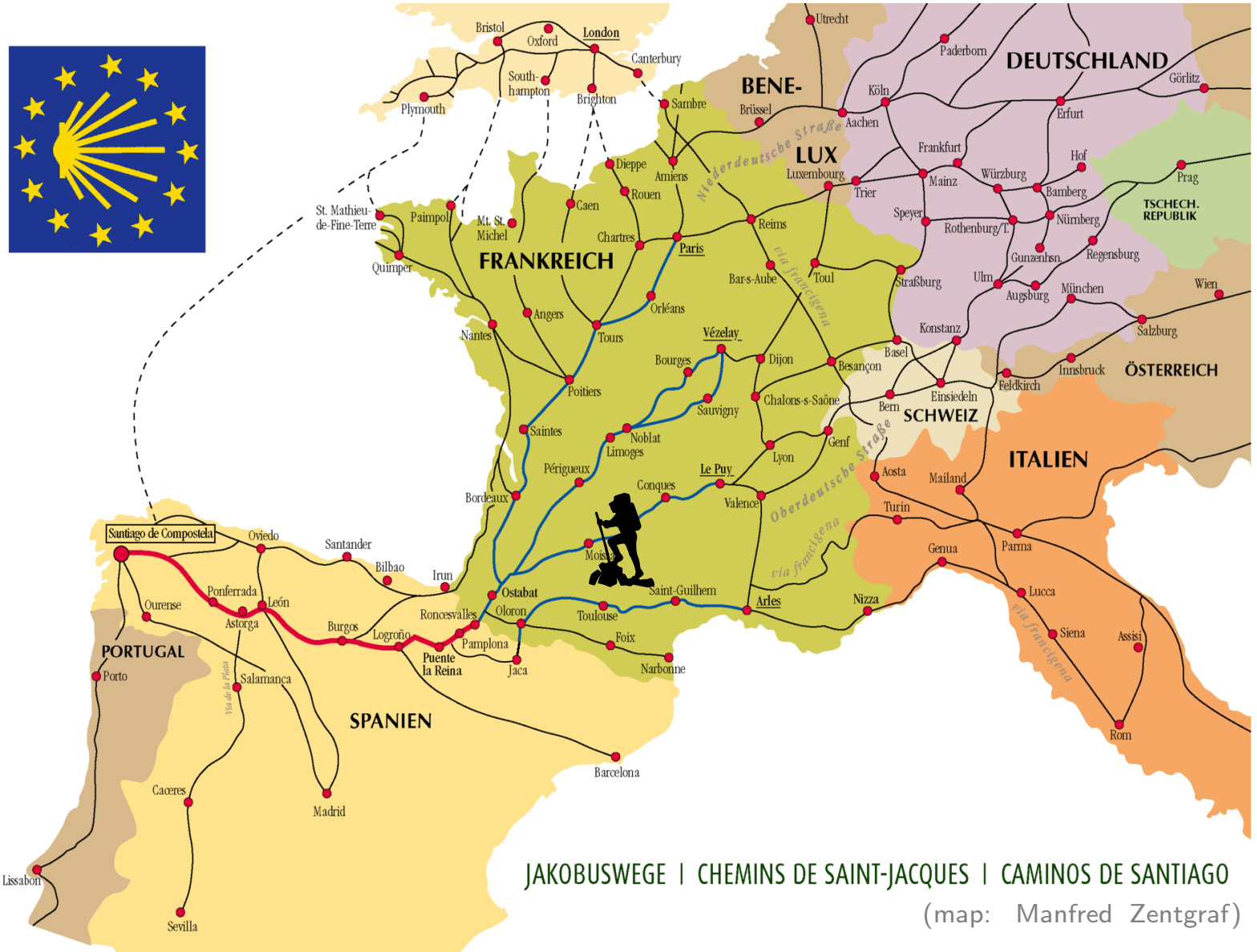
Patrick Ferber[1,2] Malte Helmert[1] Jörg Hoffmann[2]

[1]University of Basel, Switzerland
[2]Saarland University, Germany

September 1, 2020

Outline

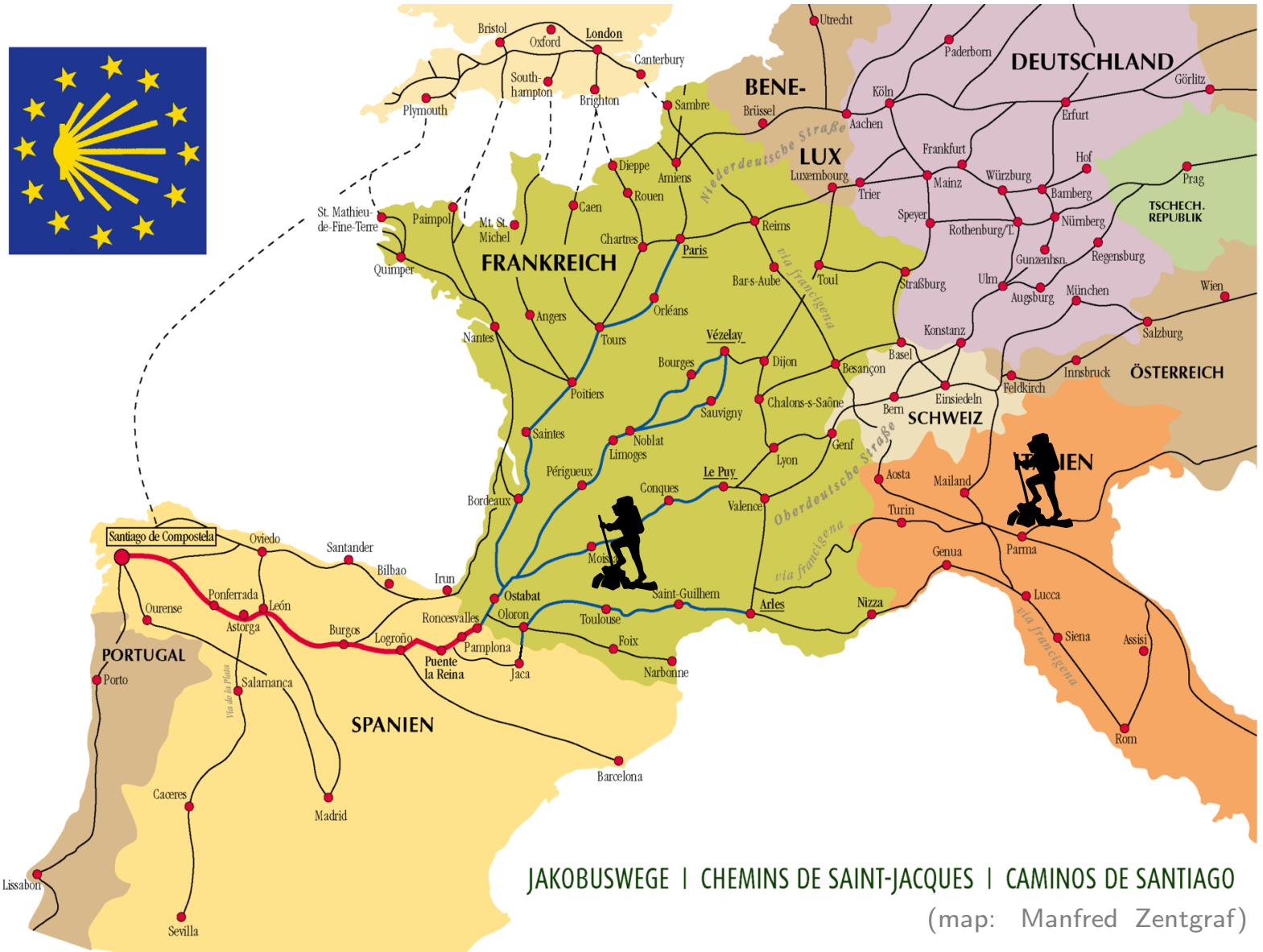
Motivation



JAKOBUSWEGE | CHEMINS DE SAINT-JACQUES | CAMINOS DE SANTIAGO

(map: Manfred Zentgraf)

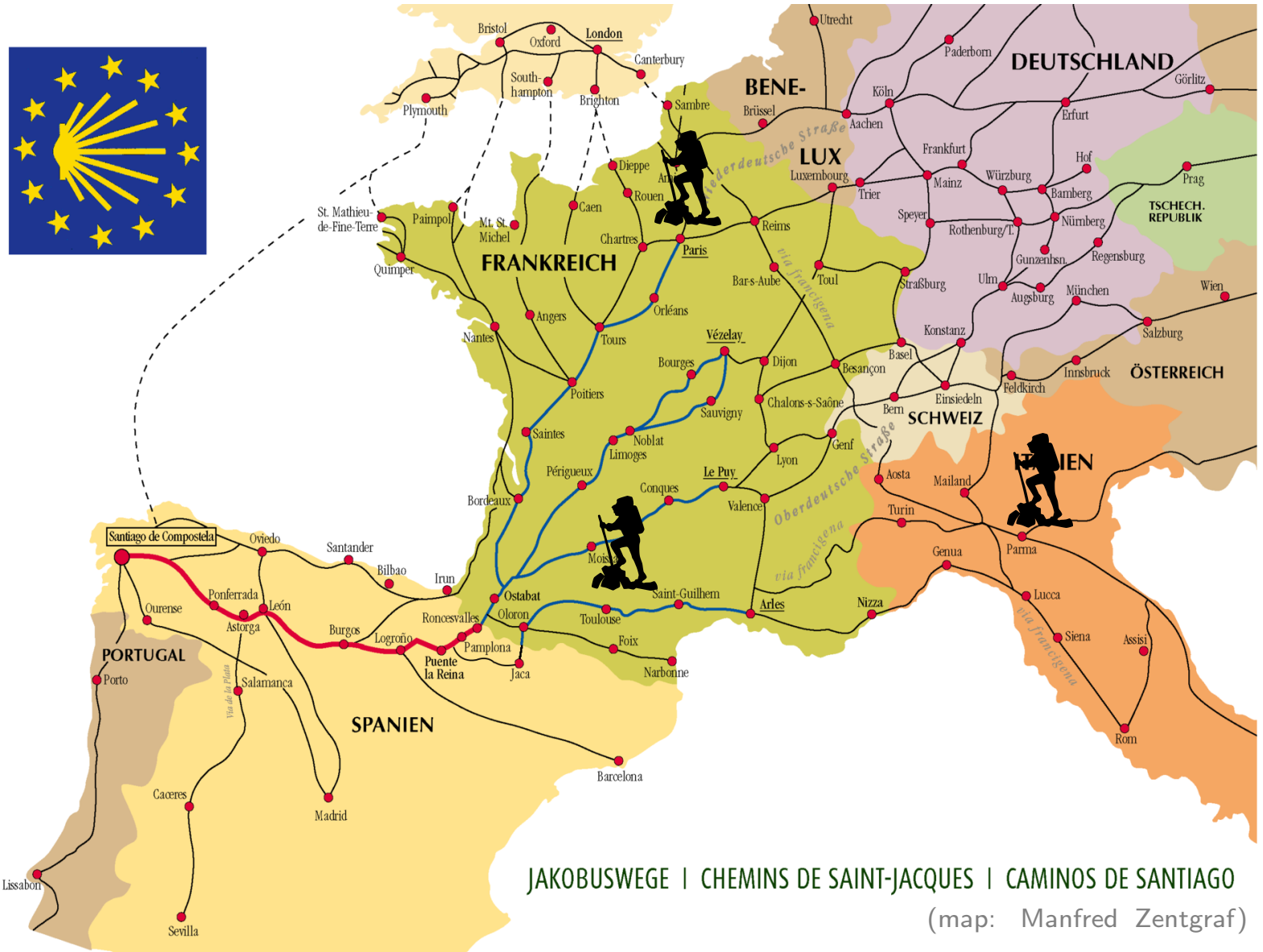
Motivation



JAKOBUSWEGE | CHEMINS DE SAINT-JACQUES | CAMINOS DE SANTIAGO

(map: Manfred Zentgraf)

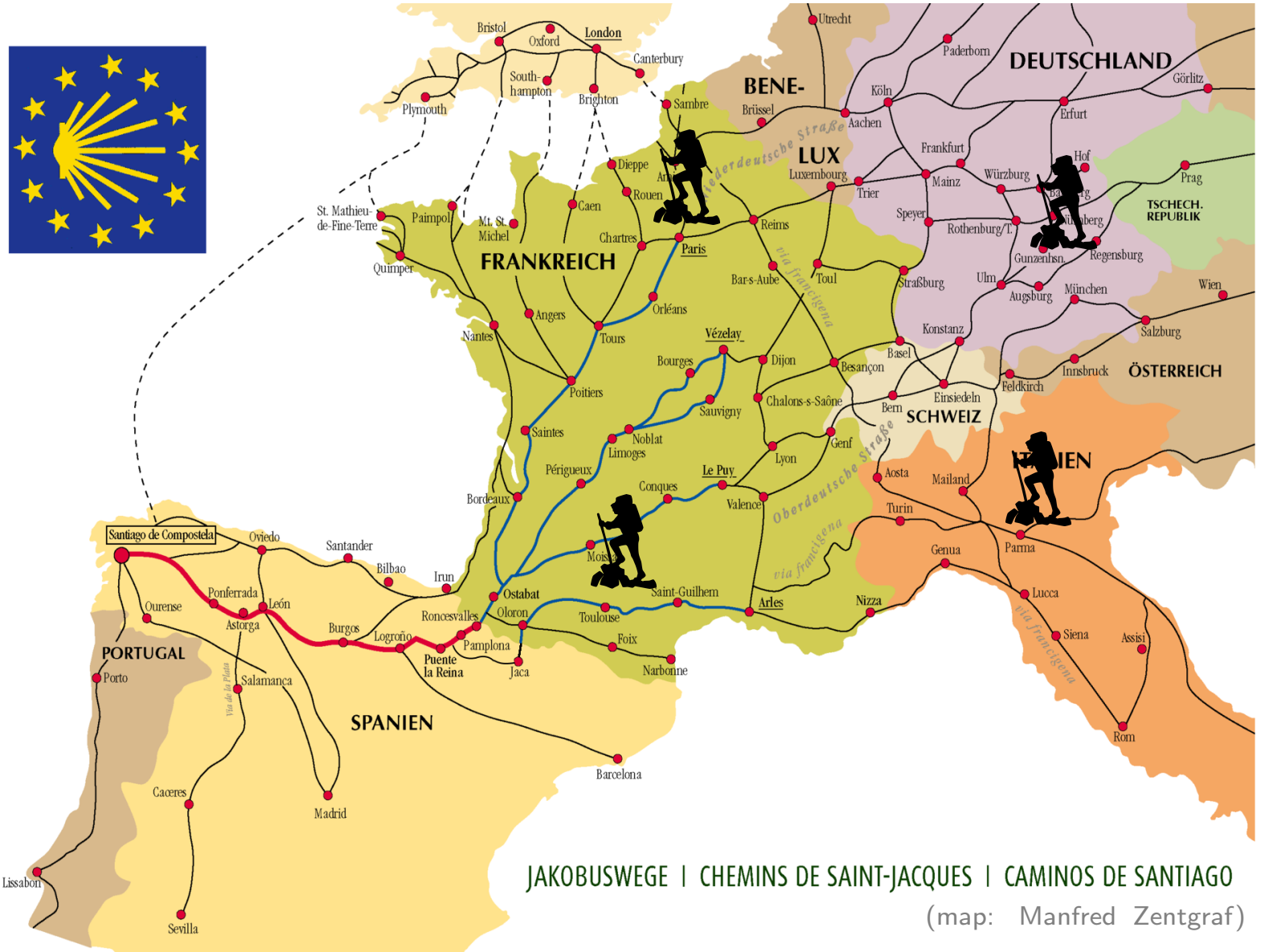
Motivation



JAKOBUSWEGE | CHEMINS DE SAINT-JACQUES | CAMINOS DE SANTIAGO

(map: Manfred Zentgraf)

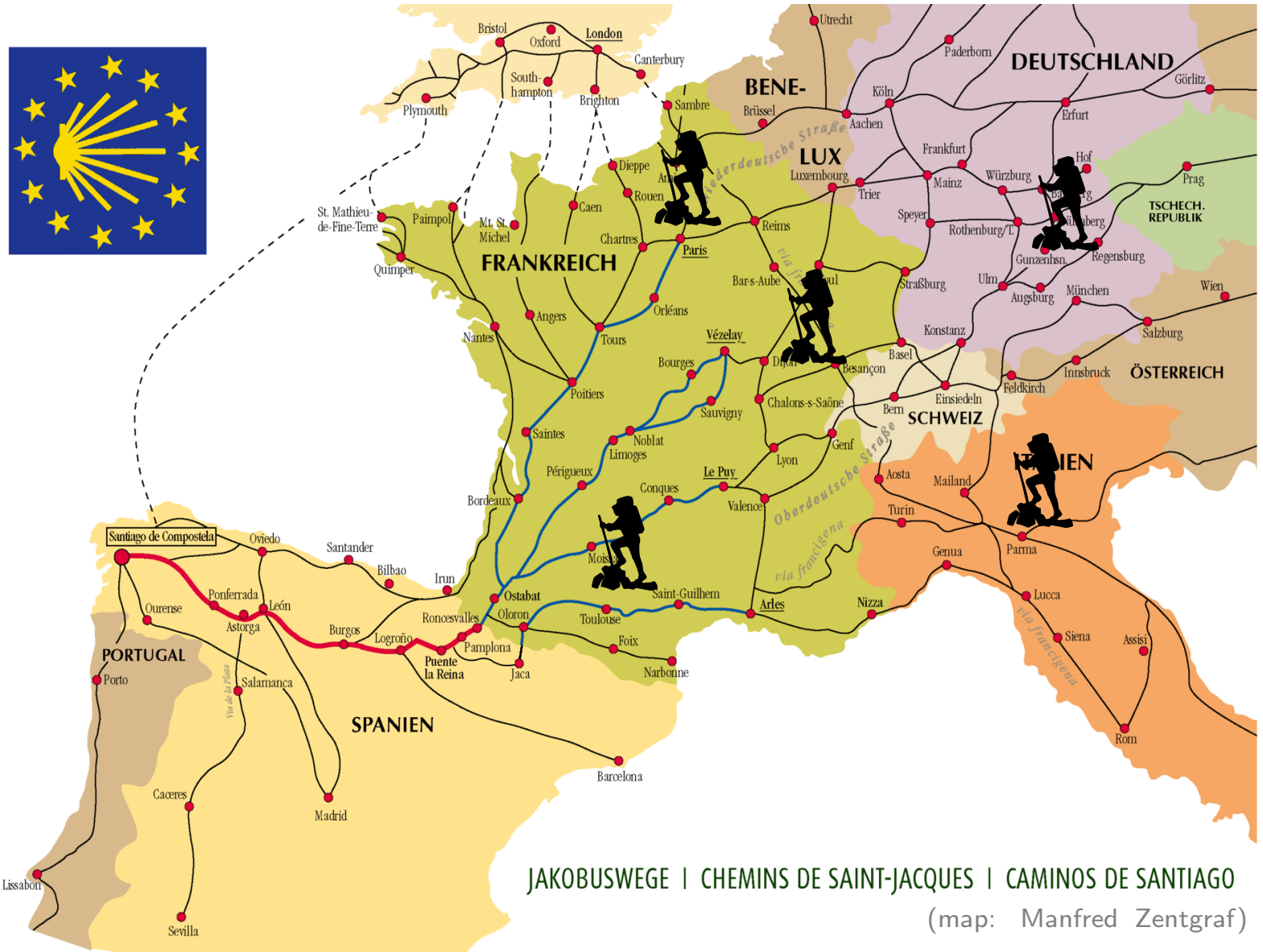
Motivation



JAKOBUSWEGE | CHEMINS DE SAINT-JACQUES | CAMINOS DE SANTIAGO

(map: Manfred Zentgraf)

Motivation



JAKOBUSWEGE | CHEMINS DE SAINT-JACQUES | CAMINOS DE SANTIAGO

(map: Manfred Zentgraf)

Motivation

often solve tasks with the same goal and state space

search algorithms start from scratch always

learn a heuristic once $\Rightarrow$ reduce search effort always

Finite-Domain Representation (FDR)

Definition

An FDR planning task is a tuple $\Pi = (V, A, I, G)$ with:

- V: a set of multi-valued variables
- A: a set of actions
- I: an initial state
- G: a partial variable assignment

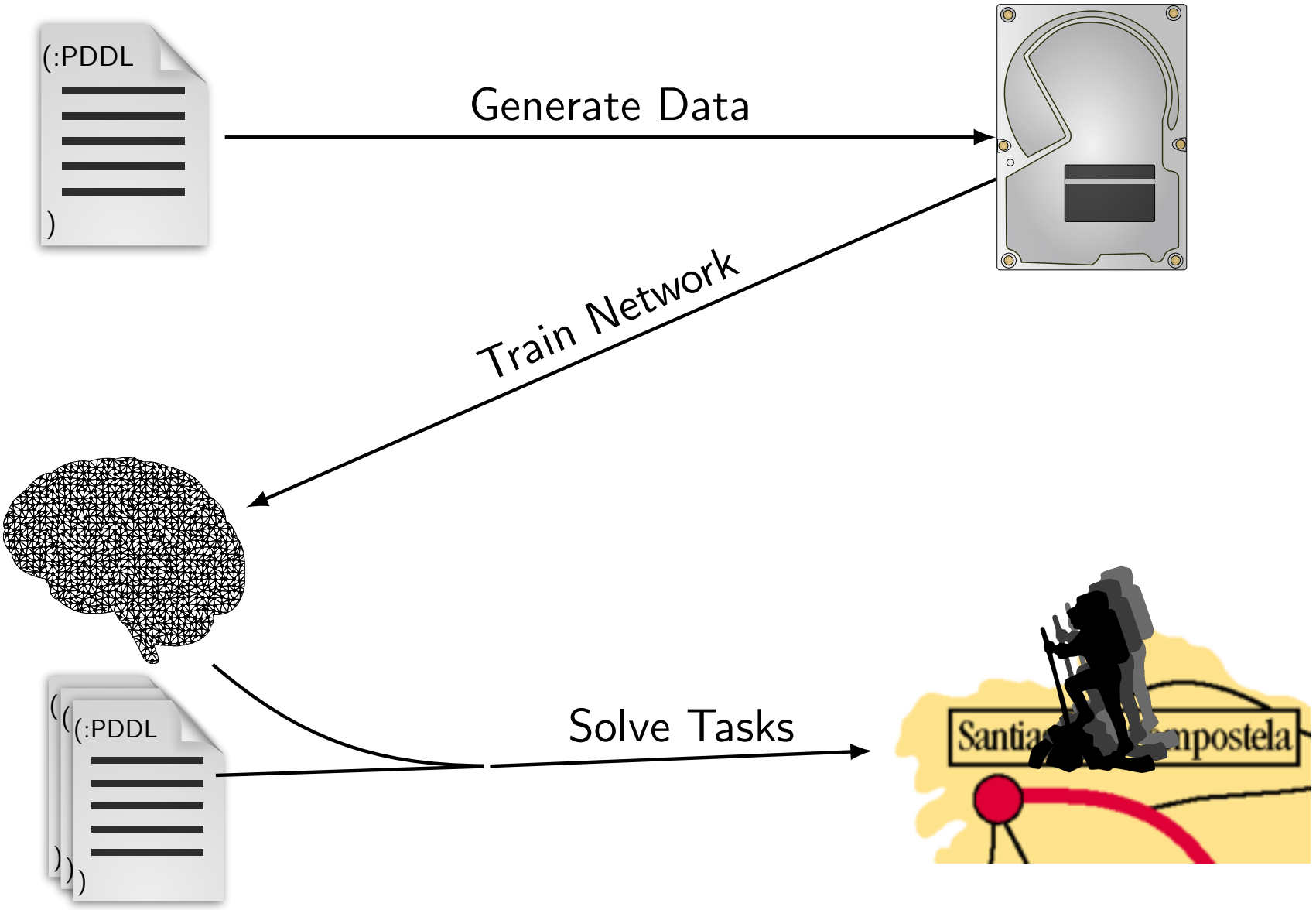
Finite-Domain Representation (FDR)

Definition

An FDR planning task is a tuple $\Pi = (V, A, I, G)$ with:

- V: a set of multi-valued variables
  $\{p \mid p \in pilgrims\}$ with $dom(p) = \{Irun, Bilbao, ...\}$
- A: a set of actions — defined by roads
- I: an initial state — $\{p \mapsto home\ town \mid p \in V\}$
- G: a partial variable assignment — $\{p \mapsto Santiago \mid p \in V\}$

Overview



Generate Data

```
1  def generate_data(
2      task,
3      timeout,
4      walk_length,
5      teacher_search,
6  ):
7      while time() < timeout:
8          new_task = random_walk(task, walk_length)
9          plan = solve(new_task, teacher_search)
10         //plan = <a_1, a_2, ..., a_n>
11         state = new_task.initial_state
12         store(state, |plan|)
13         for i in 1..|plan|:
14             state = apply(state, a_i)
15             store(state, |plan| - i)
```

Generate Data

```
1  def generate_data(
2      task,
3      timeout = 400 hrs,
4      walk_length = 200,
5      teacher_search = GBFS with h^FF,
6  ):
7      while time() < timeout:
8          new_task = random_walk(task, walk_length)
9          plan = solve(new_task, teacher_search, 30m, 3.8GB)
10         //plan = <a_1, a_2, ..., a_n>
11         state = new_task.initial_state
12         store(state, |plan|)
13         for i in 1..|plan|:
14             state = apply(state, a_i)
15             store(state, |plan| - i)
```

parallelized on 200 Intel Xeon E5-2600 cores

Train Network

Reminder

A state of an FDR task is a complete assignment to its variables.

var1	var2	...
val1 | val2 | ...	val1 | val2 | ...	...

Train Network

Reminder

A state of an FDR task is a complete assignment to its variables.

var1			var2			...
0	1	...	1	0	...	...

Train Network

Reminder

A state of an FDR task is a complete assignment to its variables.

var1			var2			...
0	1	...	1	0	...	...

Train Network

Reminder

A state of an FDR task is a complete assignment to its variables.

var1		var2		...	
0	1	...	1	0	...	...

Feed-Forward Network

Regression:	2			
Classification (one-hot):	0	0	1	0
Classification (unary):	1	1	1	0

10 fold cross-validation, 4 cores, 12GB memory & 24 hrs time limit

Solve Tasks

Use neural network as heuristic in a greedy best-first search.

- evaluate one state after another
- 1 core
- 30 min time & 3.8 GB memory limit

Domains

Blocksworld, Depots, Grid, NPuzzle, Pipesworld-NoTankage, Rovers, Scanalyzer, Storage, Transport, VisitAll

- deadend-free
- select tasks with 1s $<$ teacher search solution time $<$ 900s
- generate 200 unseen states for each selected task

Classification vs Regression

Domain	cls_{OH}	cls_{U}	reg
blocksworld	93.4	**97.2**	65.3
depots	**87.7**	76.2	77.3
grid	44.8	**93.2**	71.0
pipes-nt	84.3	**89.6**	82.0
scanalyzer	**96.2**	94.6	80.3
storage	14.5	95.5	**98.5**
transport	92.2	**99.1**	88.9
Average	73.3	**92.2**	80.5

Classification vs Regression

Domain	cls_{OH}	cls_{U}	reg
blocksworld	93.4	**97.2**	65.3
depots	**87.7**	76.2	77.3
grid	44.8	**93.2**	71.0
pipes-nt	84.3	**89.6**	82.0
scanalyzer	**96.2**	94.6	80.3
storage	14.5	95.5	**98.5**
transport	92.2	**99.1**	88.9
Average	73.3	**92.2**	80.5

Sample Selection Strategies

$$s_{\mathcal{I}} \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} s_n$$

Domain	$s_{\mathcal{I}}$	$s_{\mathcal{R}}$	π
blocksworld	0.0	**97.2**	81.6
depots	28.6	**76.2**	71.2
grid	12.2	**93.2**	48.0
pipes-nt	**90.6**	89.6	75.8
scanalyzer	16.8	**94.6**	79.4
storage	13.5	**95.5**	52.0
transport	17.8	**99.1**	93.7
Average	25.6	**92.2**	71.7

Sample Selection Strategies

$$s_{\mathcal{I}} \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} s_n$$

Domain	$s_{\mathcal{I}}$	$s_{\mathcal{R}}$	π
blocksworld	0.0	**97.2**	81.6
depots	28.6	**76.2**	71.2
grid	12.2	**93.2**	48.0
pipes-nt	**90.6**	89.6	75.8
scanalyzer	16.8	**94.6**	79.4
storage	13.5	**95.5**	52.0
transport	17.8	**99.1**	93.7
Average	25.6	**92.2**	71.7

Sample Selection Strategies

$$s_{\mathcal{I}} \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} s_n$$

π: all

Domain	$s_{\mathcal{I}}$	$s_{\mathcal{R}}$	π
blocksworld	0.0	**97.2**	81.6
depots	28.6	**76.2**	71.2
grid	12.2	**93.2**	48.0
pipes-nt	**90.6**	89.6	75.8
scanalyzer	16.8	**94.6**	79.4
storage	13.5	**95.5**	52.0
transport	17.8	**99.1**	93.7
Average	25.6	**92.2**	71.7

Sample Selection Strategies

$s_{\mathcal{R}}$: any

$$s_{\mathcal{I}} \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} s_n$$

π: all

Domain	$s_{\mathcal{I}}$	$s_{\mathcal{R}}$	π
blocksworld	0.0	**97.2**	81.6
depots	28.6	**76.2**	71.2
grid	12.2	**93.2**	48.0
pipes-nt	**90.6**	89.6	75.8
scanalyzer	16.8	**94.6**	79.4
storage	13.5	**95.5**	52.0
transport	17.8	**99.1**	93.7
Average	25.6	**92.2**	71.7

Sample Selection Strategies

$s_{\mathcal{R}}$: any

$$s_{\mathcal{I}} \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} s_n$$

π: all

Domain	$s_{\mathcal{I}}$	$s_{\mathcal{R}}$	π
blocksworld	0.0	**97.2**	81.6
depots	28.6	**76.2**	71.2
grid	12.2	**93.2**	48.0
pipes-nt	**90.6**	89.6	75.8
scanalyzer	16.8	**94.6**	79.4
storage	13.5	**95.5**	52.0
transport	17.8	**99.1**	93.7
Average	25.6	**92.2**	71.7

Coverage

Domain	NN	h^{FF}	Lama
blocksworld	97.2	**100**	**100**
depots	76.2	92.1	**100**
grid	93.2	93.2	**100**
pipes-nt	89.6	63.6	**98.7**
scanalyzer	94.6	97.8	**100**
storage	95.5	94.5	**96**
transport	99.1	**100**	**100**
Average	92.2	91.6	**99.2**

Expansions

Domain	NN	h^{FF}	Lama
blocksworld	1381	12665	**465**
depots	**182**	47731	8734
grid	493	2439	**150**
pipes-nt	**354**	832	1676
scanalyzer	269	482	**208**
storage	4208	**2089**	24712
transport	3002	4586	**192**
Average	**1413**	10118	5162

Unsuccessful Domains

	coverage	
Domain	NN	h^{FF}
npuzzle	0.0	**97.3**
rovers	53.0	**73.9**
visitall	0.2	**94.1**
Average	17.7	**88.4**

Summary

Framework to train heuristics for Classical Planning

- generate training data
- use supervised learning to train networks
- use network as heuristic in any search algorithm

Explored a rich set of hyperparameters

Can outperform the state of the art in informedness and runtime

Outlook: Reinforcement Learning