Train routing without human-expert knowledge?

Presentation

Author(s):
Jusup, Matej

Publication date:
2024-05

Permanent link:
https://doi.org/10.3929/ethz-b-000676225

Funding acknowledgement:
181210 - DADA - Dynamic data driven Approaches for stochastic Delay propagation Avoidance in railways (SNF)

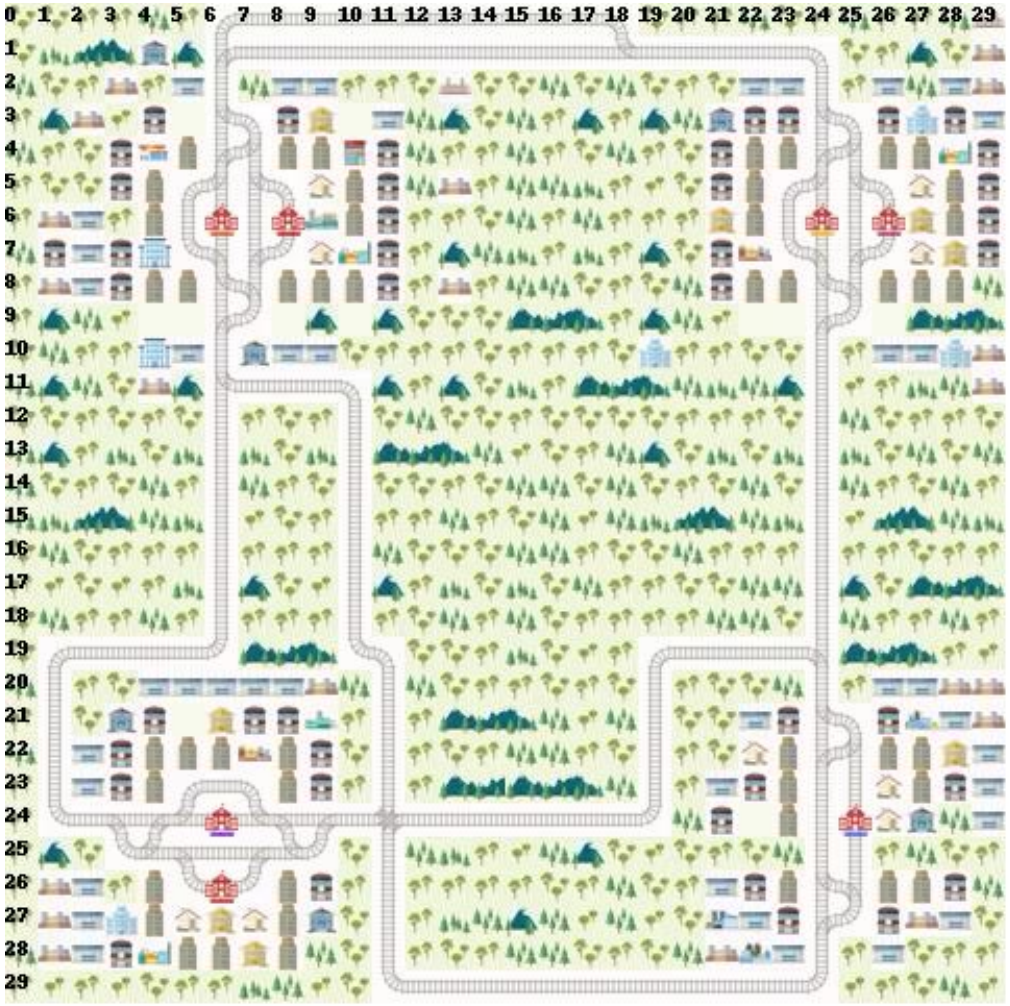
Train Routing Without Human-Expert Knowledge?

Presentation: Matej Jusup

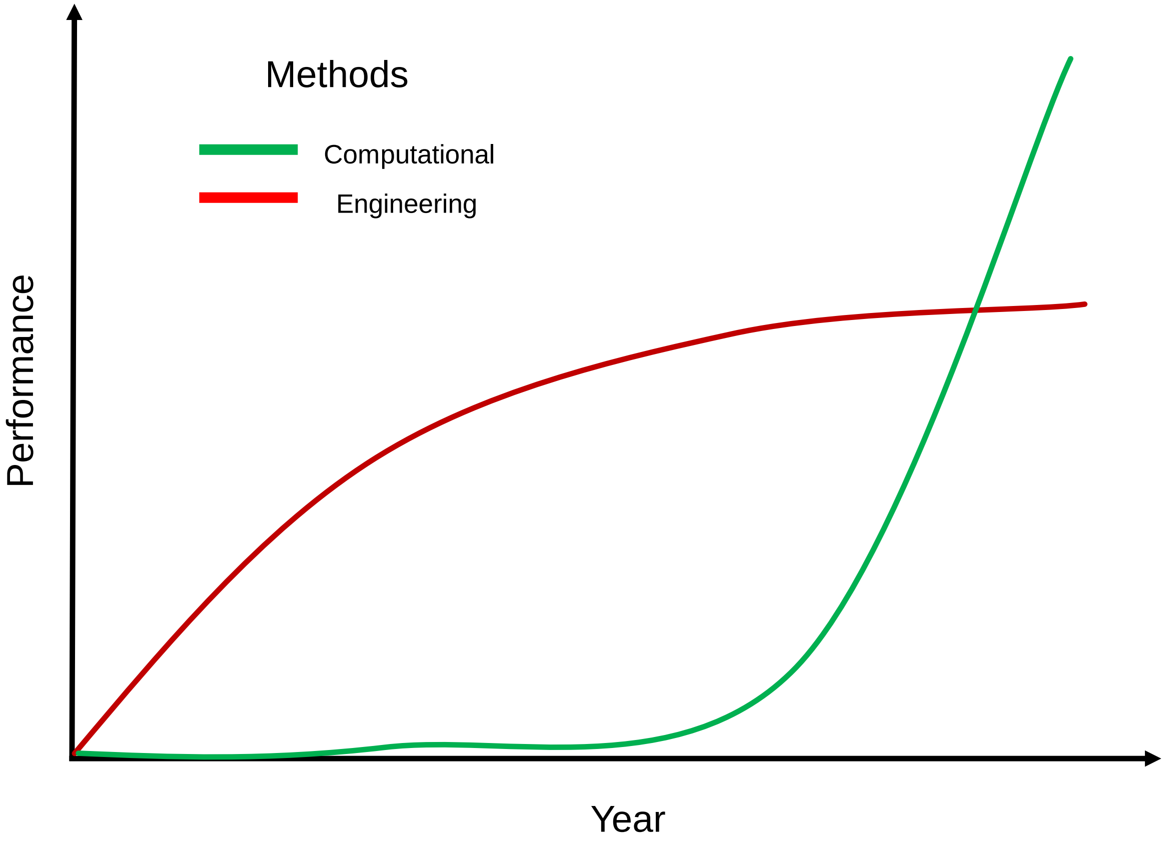
Co-authors: T. Birchler, J. Kirschner, I. Bogunovic, A. Krause, F. Corman

Can we achieve train routing with a minimal expert knowledge, or even without it?

20 trains trying to reach end station given start station



Engineering vs. computational methods



The image is just a rough interpretation of the actual progress.

What history teaches us?

Richard S. Sutton: The Bitter Lesson[1]

- Humans' tendencies – leverage expert knowledge to achieve marginal improvements
- Methods which focus on human understanding are effective only short-term
- General methods that leverage computation win in the long-run

Can we learn from board games, natural language processing, computer vision and other examples?

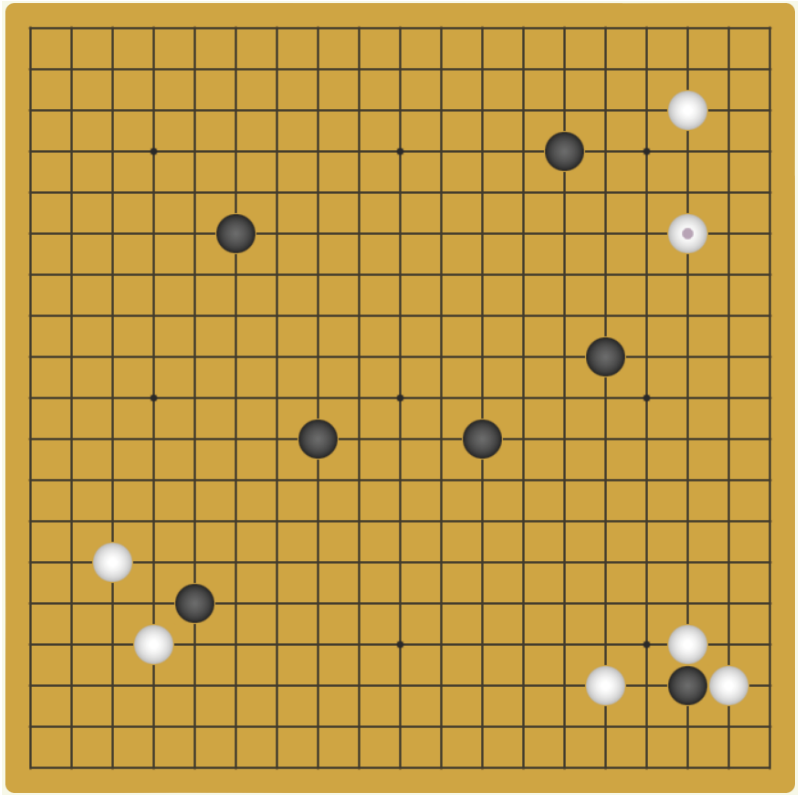
1 http://www.incompleteideas.net/IncIdeas/BitterLesson.html

A ”driver” behind computational methods

Chess

- 8x8 board with 32 pieces at the start of the game
- The goal is checkmating opponents king

Go



- 19x19 empty board at the start of the game
- Black and white stones put on the board in turns
- The goal is occupying most “territory”

A brief history of chess engines

1957

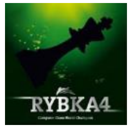
- Hard-coded “human reasoning”

1997

- MCTS
- Human-expert value function
- Specialized optimization techniques

1998-2015

- MCTS
- Improved value function
- Improved optimization



2016

- Neural-MCTS
- NN learn from human-expert games

2017

- Learns via self-play
- No human-experts
- Only game rules are given

A brief history of Go engines

Until 2005

- Hard-coded “human reasoning”
- Achieves amateur level

2005-2015

- MCTS achieved master level
- Common belief – no super-human performance before 2050’s

2016

- Neural-MCTS
- NN learn from human-expert games
- Match win against human professional

2017

- Learns via self-play
- No human-experts
- Only game rules are given
- Match win against the world champion
- Discovered patterns unknown to humans

Not impressed yet?

“AlphaZero” discovered a better matrix multiplication algorithm than humans

- Computational complexity O(n^3)

1812

1969

- Contrary to the established beliefs found O($n^{log_2 7}$) = O($n^{2.8074}$) algorithm

- Found an O($n^{2.373}$) algorithm
- Works only for infinitely big matrices

2020

2022

- Practical O($n^{2.778}$) algorithm
- Uses existing algorithms for exploration
- For the first time in history, a computer discovered a better algorithm than humans!

What do board games, matrix multiplication and railway optimization have in common?

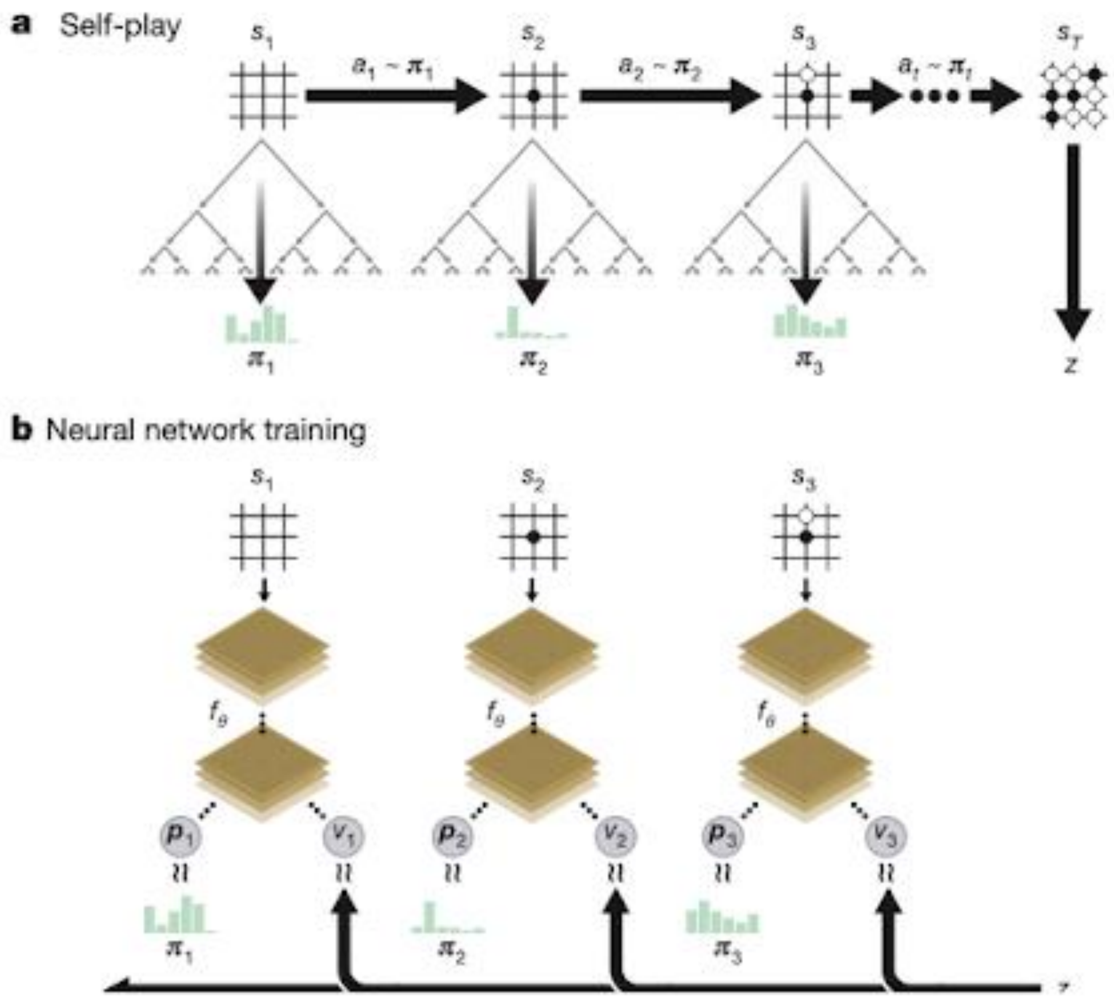
State-of-the-art railway optimization:

- Relies on human-expert knowledge
- Uses highly-specialized, sophisticated optimization techniques
- There is no unified algorithm that solves related problems – routing, scheduling, re-scheduling

My vision:

- Define railway optimization as a simple game
- Use self-play to solve broad class of problems
- Use as little human-expertize as possible – ideally none

Self-play – the core idea behind AlphaZero



Source: Mastering game Go without human knowledge [5]

Neural network (NN): f_θ
NN input: board state s
NN outputs:
- Value/win probability v
- Moves distribution: p

- NN f_θ guides MCTS until the outcome z is reached (win/loss/draw)
- NN f_θ parameters θ are updated

Train routing as a game

INPUT

- Railway infrastructure
  - Lines and stations
  - Represented as a 2D grid

- Players = trains

- Constraints
  - Initial and target station for each train

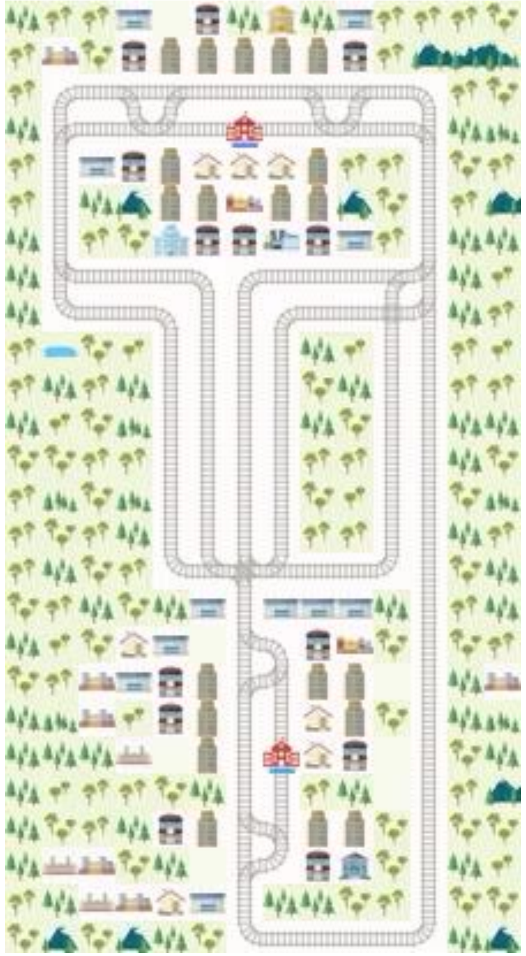
GOAL

- Reaching the target stations
  - Preferably with minimal travel time

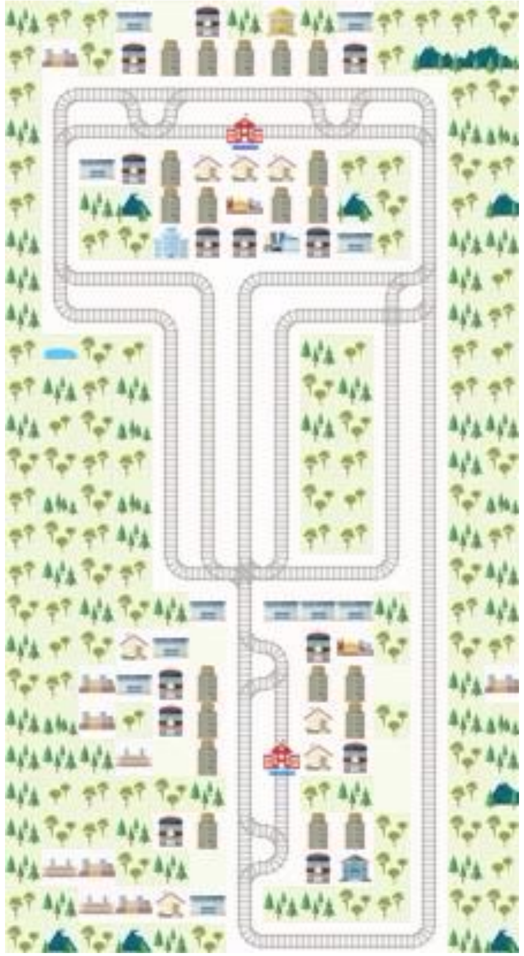
Source: Flatland simulator [4]

Can we solve train routing without human-expert knowledge?

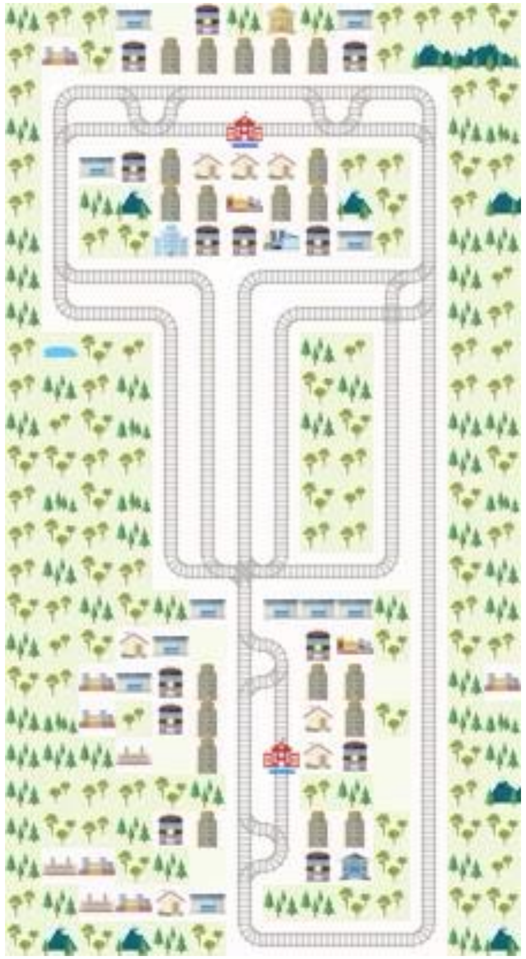
Training round 0 – random policy



Training round 20

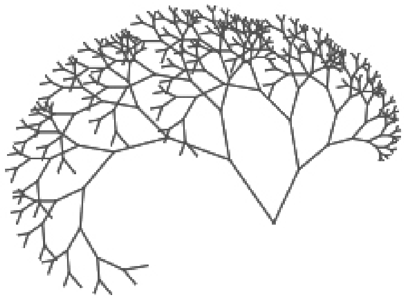


Training round 40



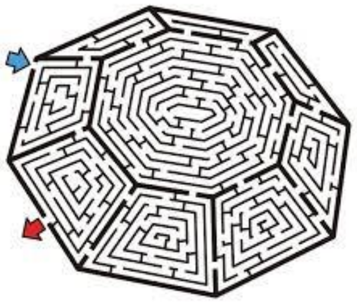
Displayed results are achieved by a pure self-play, i.e., without any expert knowledge.

The main challenges compared to board games and matrix multiplication



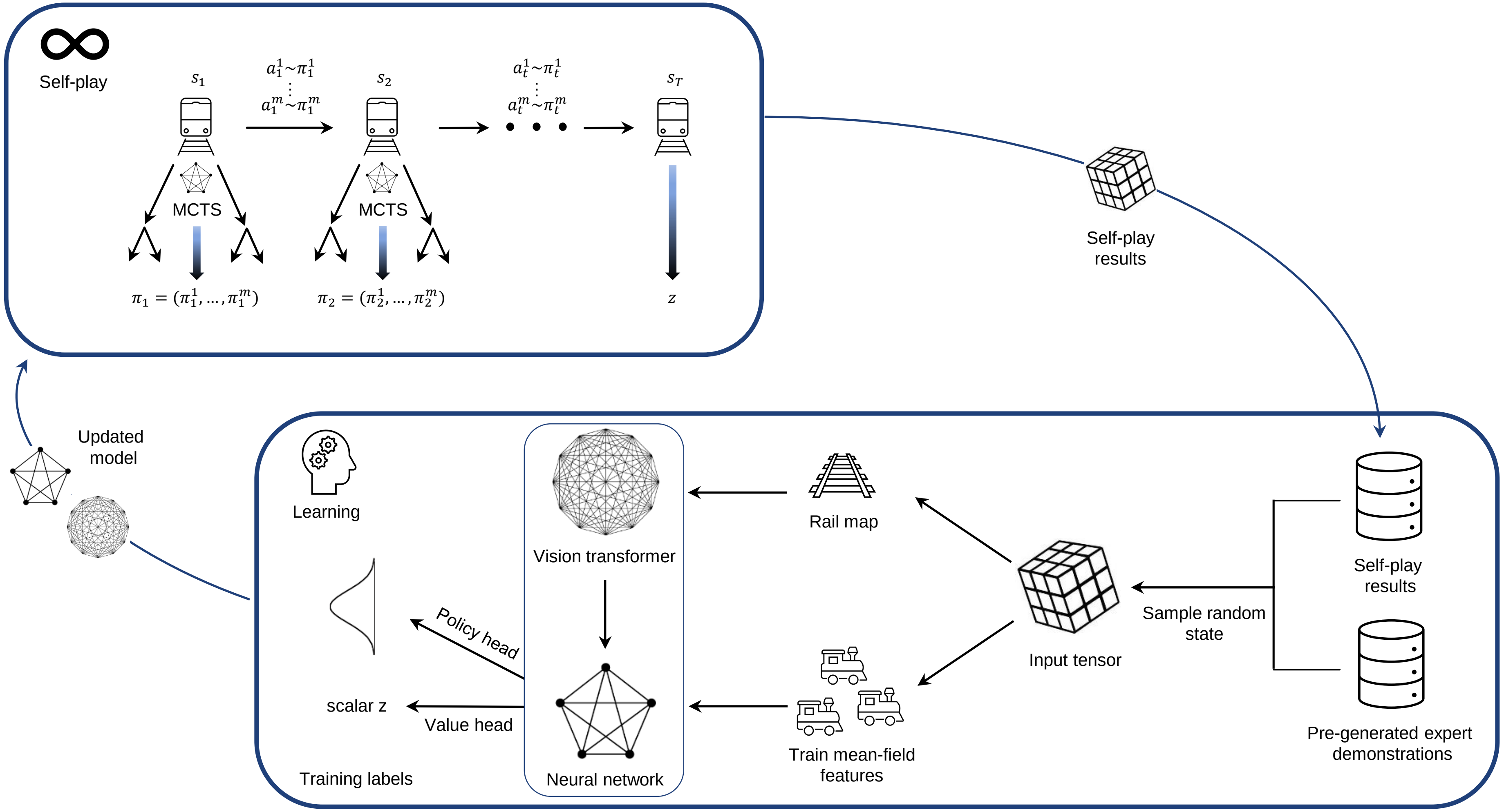
- ➢ Extreme branching factor
  - Multi-agent setting

- ➢ Actions are irreversible
  - Safety considerations – no deadlocks allowed

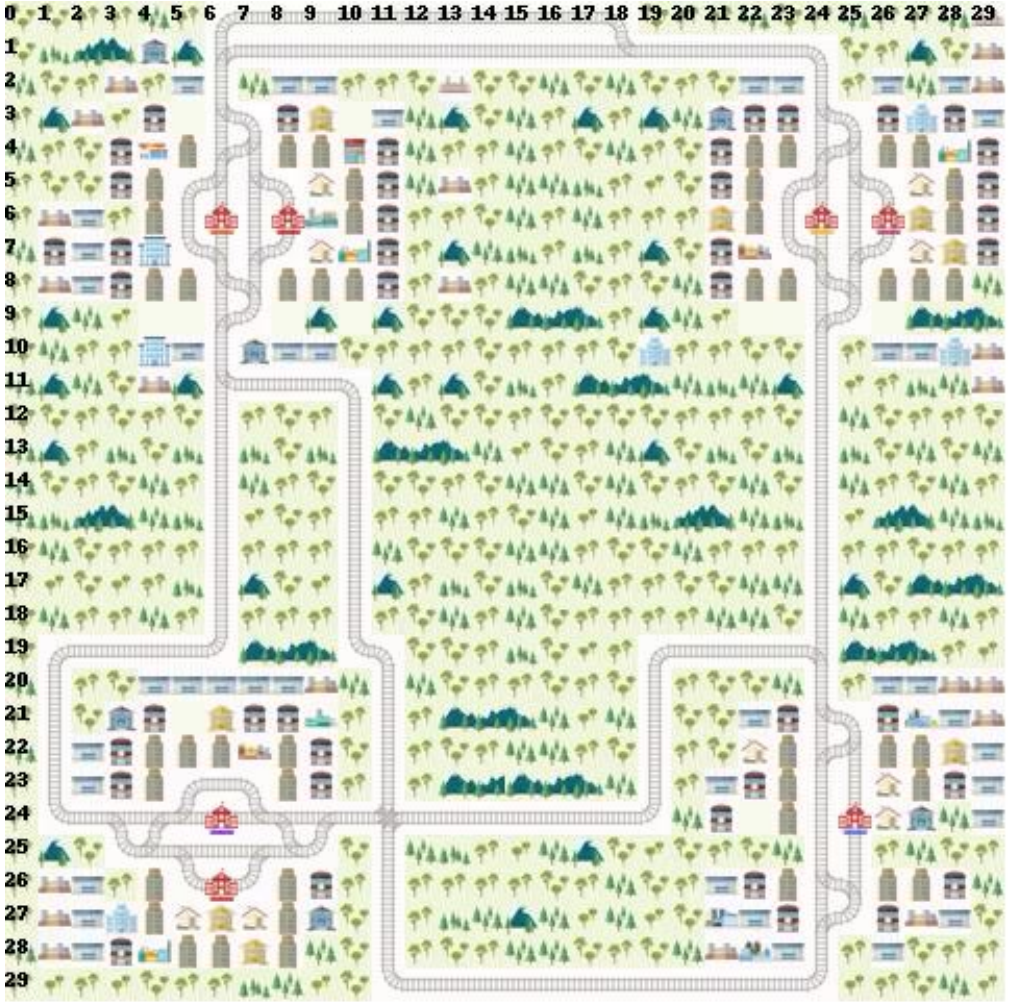


- ➢ Heavy exploration
  - Extremely difficult to get positive examples

Preliminary model architecture



Minimal expert guidance helps with challenging networks



- Initial policy uses a few expert examples
- Self-play for the rest of the training
- Results after only 3 training rounds

Initial results

We match the best solver's optimal results on the networks of up to 10 agents!

	Number of agents	**Proposed setup**	No planning	No expert data	Monte-Carlo tree search	Shortest path	Old Driver
Planning rollout		NeuralNet	NeuralNet	NeuralNet	Heuristic		
No. rollout simulations		32	0	32	32		
Expert demonstrations		50%	50%	0%	0%		
Percentage of successfull arrivals	1	100%	100%	100%	100%	100%	100%
	5	100%	67%	53%	33%	53%	100%
	10	100%	90%	27%	40%	37%	100%
	Avg	*100%*	*86%*	*60%*	*58%*	*63%*	*100%*
Average travel time	1	32.33	32.33	33.67	34.00	32.33	32.33
	5	33.80	deadlock	deadlock	deadlock	deadlock	33.80
	10	38.67	deadlock	deadlock	deadlock	deadlock	38.67
	Avg	*34.93*	*deadlock*	*deadlock*	*deadlock*	*deadlock*	*34.93*

Our model

Best solver

Summary

- Railway optimization problems can be naturally modelled as a game
  - Train routing
  - Train scheduling
  - Train rescheduling
- Self-play could lead to a unified algorithm
- Preliminary results suggest it is a promising research direction!

Contact

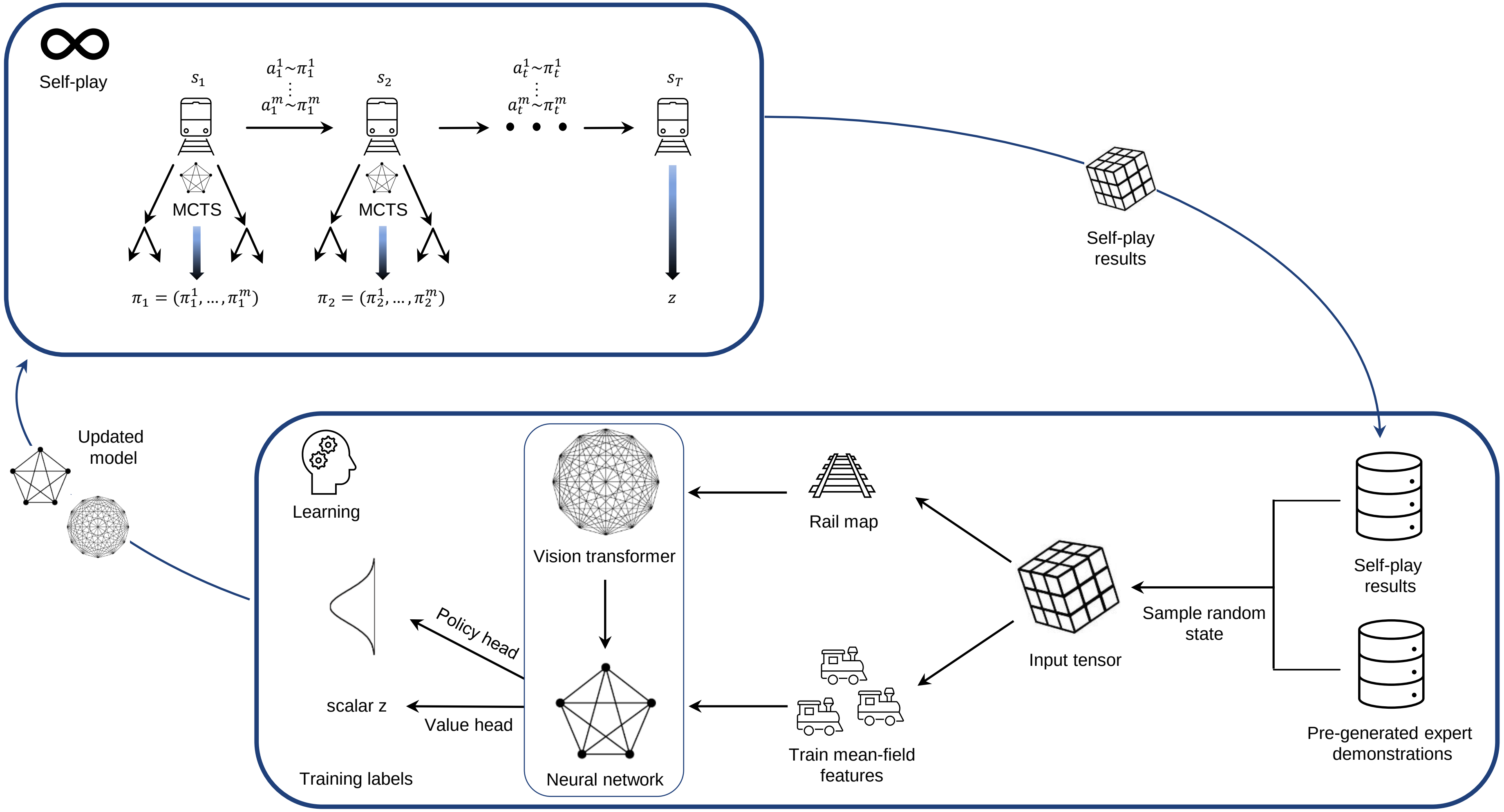
Questions?

References

1. Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre,D. Kumaran, T. Graepelet et al.(2018) A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,Science,362(6419) 1140–1144
2. Li, J., Z. Chen, Y. Zheng, S.-H. Chan, D. Harabor, P. J. Stuckey, H. Ma and S. Koenig(2021) Scalable rail planning and replanning: Winning the 2020 flatland challenge, paper presented at the Proceedings of the International Conference on Automated Planning and Scheduling, vol. 31, 477–485.
3. Jusup, M., A. Trivella and F. Corman (2021) A review of real-time railway and metro rescheduling models using learning algorithms, paper presented at the30th International Joint Conference on Artificial Intelligence (IJCAI-21).
4. SBB and AICrowd (2022) Flatland,https://flatland.aicrowd.com/intro.html.
5. Silver, D., Schrittwieser, J., Simonyan, K. et al. Mastering the game of Go without human knowledge. Nature 550, 354–359 (2017)
6. Rosin, C. D. (2011). Multi-armed bandits with episode context. Annals of Mathematics and Artificial Intelligence, 61(3), 203-230.
7. Fawzi, A., Balog, M., Huang, A. *et al.* Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **610**, 47–53 (2022).
8. Chaslot, G. M. J., Winands, M. H., Herik, H. J. V. D., Uiterwijk, J. W., & Bouzy, B. (2008). Progressive strategies for Monte-Carlo tree search. New Mathematics and Natural Computation, 4(03), 343-357.
9. Bertsekas, D. (2021). Multiagent reinforcement learning: Rollout and policy iteration. *IEEE/CAA Journal of Automatica Sinica*, *8*(2), 249-272.

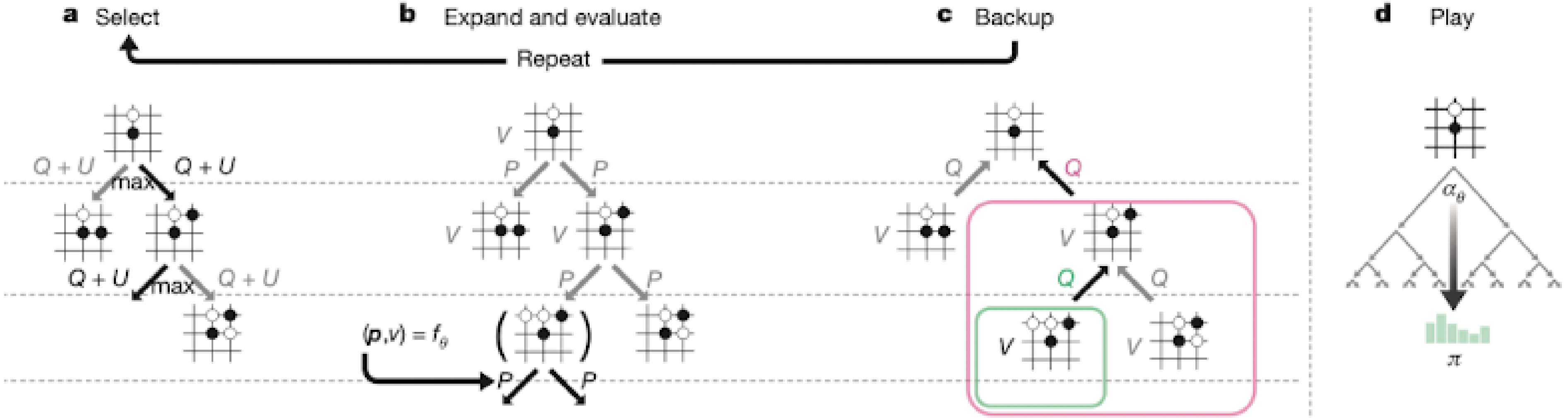
APPENDIX

Model architecture



Self-play – the core idea behind AlphaZero

- Computer plays the games againts itself from which it learns value function approximation and policy for choosing moves during MCTS



Source: Mastering game Go without human knowledge [5]

Back to board games…
Why did chess engines achieve human-level performance in late 90's while we needed to wait for another 20 years for the same results in computer Go?

- Go has higher branching factor
  - Breadth of 250 moves each turn and average game depth of 150 moves results in 10^{360} possible moves
  - Compared to chess with 10^{120} possible moves

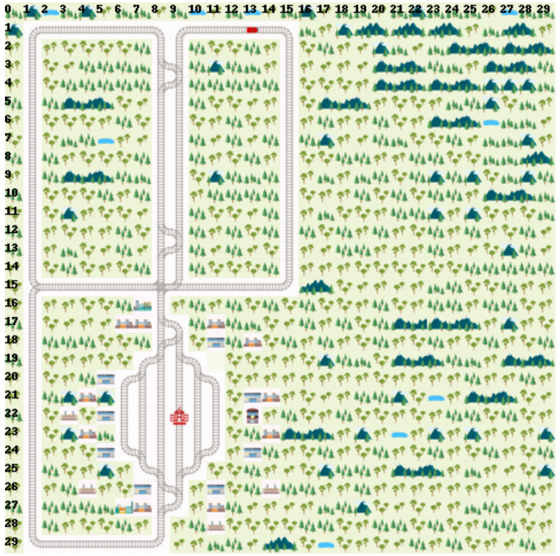
- Due to Go's simplicity, it is hard/impossible to write a simple closed-form value function
  - In chess it is much easier to write good enough value function due to more complex rules and piece interconnection
  - E.g., assign value to each piece, control over the center, king's safety, control of light/dark squares, number of protected pawns, number of loose pawns…

What is more valuable, optimal, but practically infeasible theoretical result or very good practical results?

- Theoreticians say that deep, novel concepts is the only way forward. Even though they are not practically feasible today, if the ideas are here they might become feasible in the future

- Practitioners say that having small improvements today are of utmost value because 10-20% faster computation means 10-20% more productive time, less energy consumption, and more computational resources for other tasks

How multi-agent AlphaZero works?



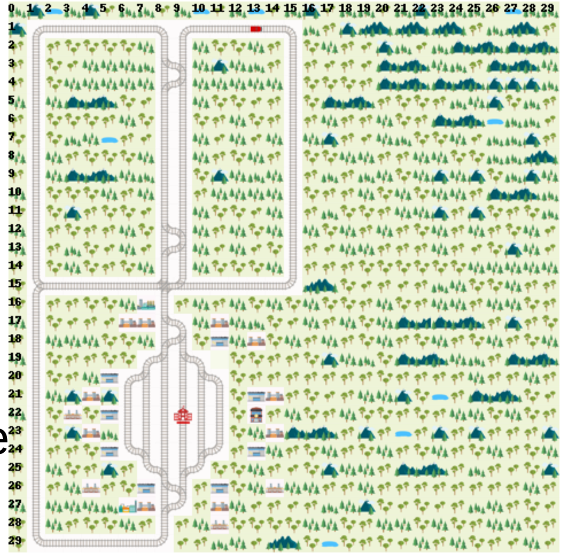
Extract features from the network

We can imagine it as a brain of the model

Neural network (NN)

1. Predict value of the state/position on the map
2. Predict actions distribution

v = 0.8
p = [0, 0.6, 0.2, 0.1]
left, straight, right, stop



NN is the core of the model. Practice shows that MCTS will very likely be successful if NN can make high-quality “position evaluation”

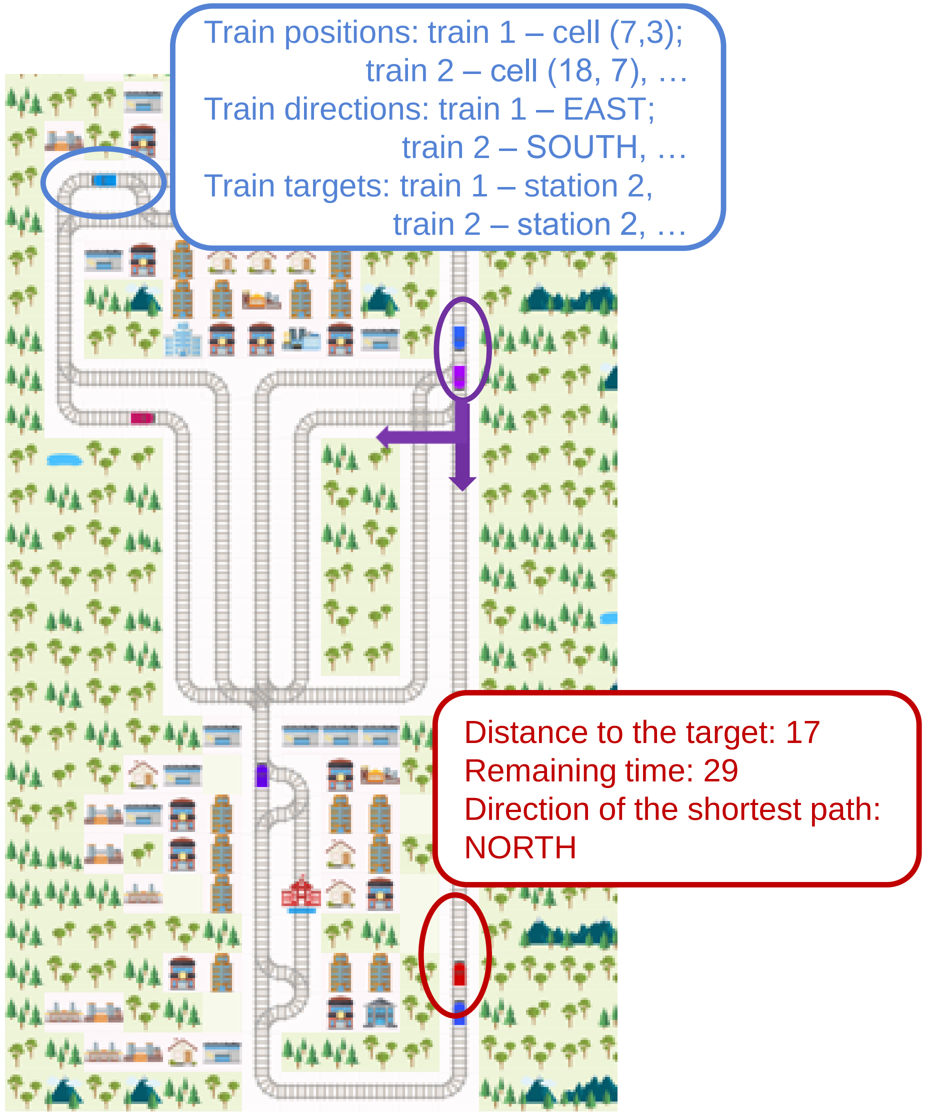
Some details on AlphaRail mechanics

- Controller is aware of a **global observation**, i.e., railway infrastructure snapshot that consists of rail lines, train positions and directions, and train targets

- Model generates train **actions[1] dynamically at discrete timesteps**

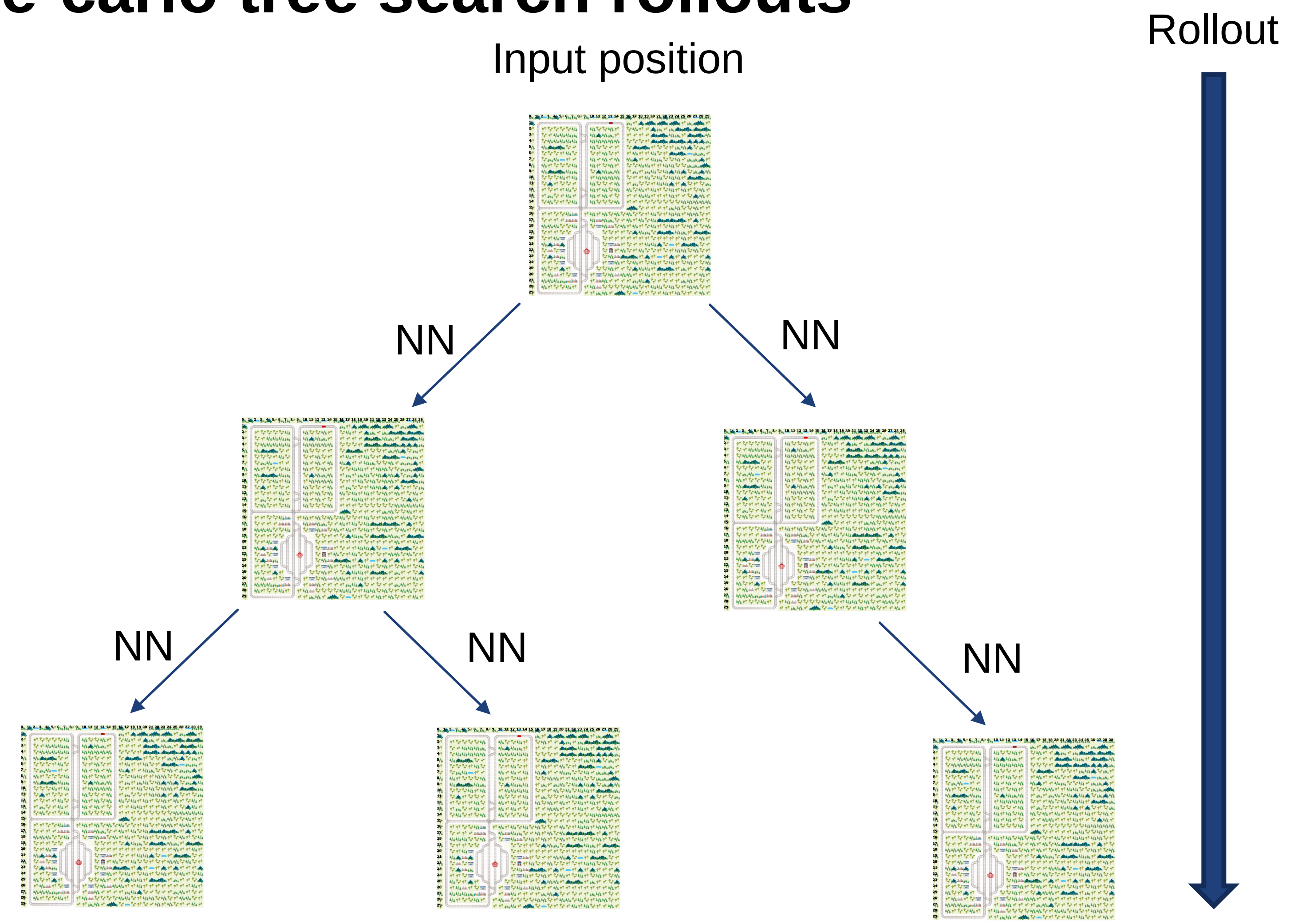
- Training is done by **self-play reinforcement learning**, i.e., model learns from its own mistakes

- At the end of each self-play round the model is evaluated by a simple (normalized discounted) delay cost function



1 The possible actions are going forward, turning left, turning right or stop moving, i.e., turning direction is not allowed.

Monte-carlo tree search rollouts



Use some method to determine the most promising branch.

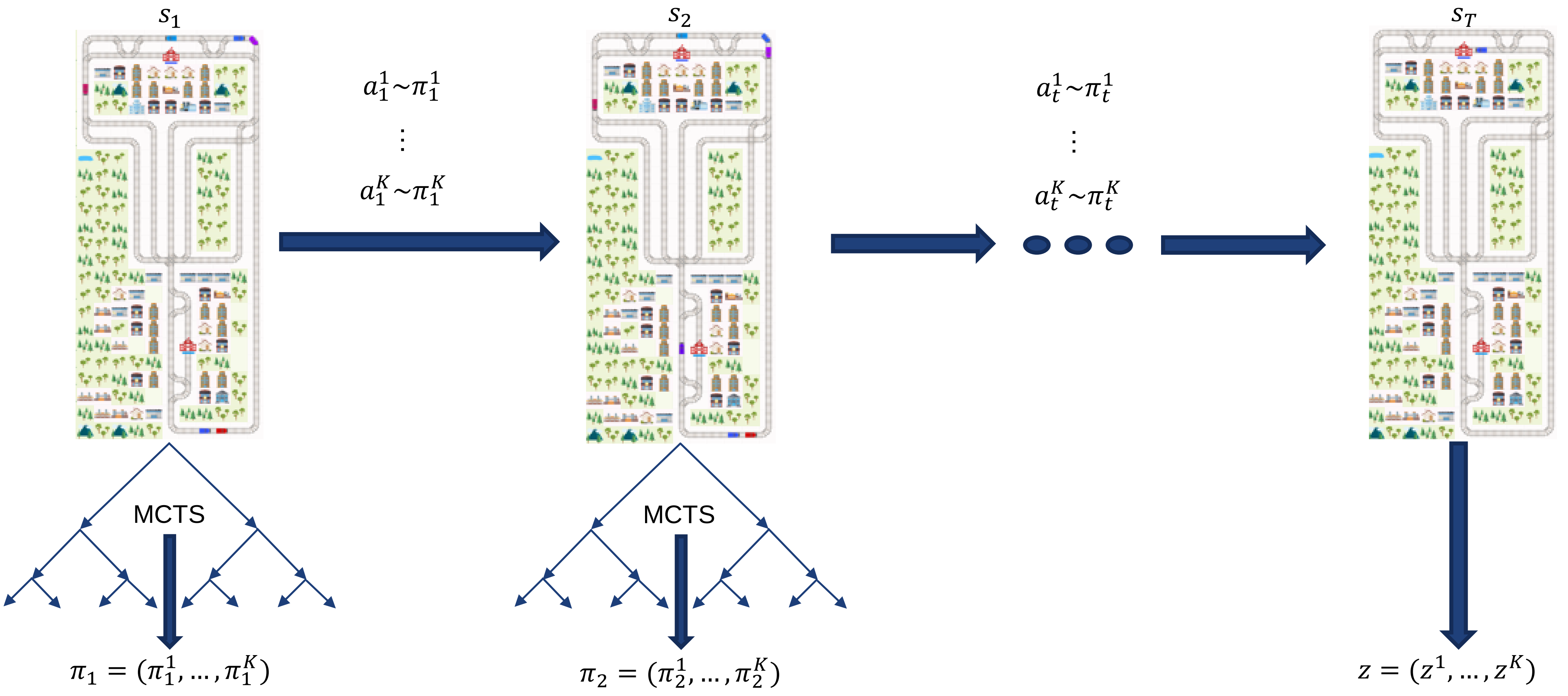
We use upper-confidence bound (UCB) to guide the rollout and determine the actions in the end.

If NN has high-quality evaluations, it will lead the MCTS search process to the correct conclusion.

Critical question: How does the NN gets better?

Multi-agent AlphaZero architecture at a glance!

A self-play generates the game outcomes used for neural-network f_θ parameters update[1]



MCTS = Monte-Carlo tree search

1 Neural-network is used for actions generation within MCTS.