Reinforcement Learning
for the real world

Harshad Khadilkar
Tata Consultancy Services Ltd.

Motivation: Why RL?

Core ideas for the future

This is about letting an ecosystem of machines teach itself superhuman capabilities

Why?

“Because it’s there”

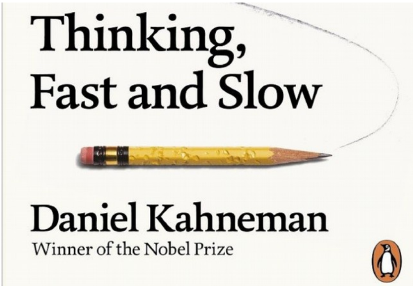
- George Mallory (1923), when asked why he wanted to climb Mt. Everest

RL in the optimization space

	Easy	Hard
Slow	How many eggs for breakfast	How much down-payment on car loan
Fast	Which lane to choose at the toll booth	Packing irregular boxes arriving on a conveyor belt

Pressure increases →

Difficulty increases →



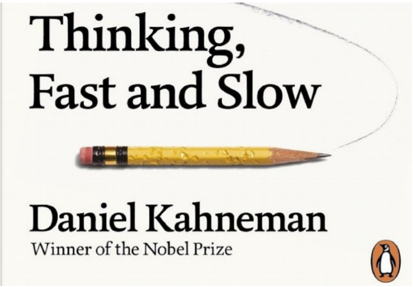
2 Motivation

RL in the optimization space

	Easy	Hard
Slow	Linear programming and its variants	Meta-heuristics
Fast	Rule-based planning Supervised deep learning	Reinforcement learning

Pressure increases

Difficulty increases



When to use RL

Necessary conditions: Answer YES to all of the following

Use for tasks that humans find hard to do (or to do well) → No ideal reference

When time is short → Can’t search or solve in real-time

When the system is hard to define, or complex → No analytical relationships

“The most important training in Unseen University [for wizards] wasn’t how to do magic, but to know when not to use it” - Terry Pratchett

Core ideas for the future

This is about letting an ecosystem of machines teach itself superhuman capabilities

Why?

“Because it’s there”

- George Mallory (1923), when asked why he wanted to climb Mt. Everest

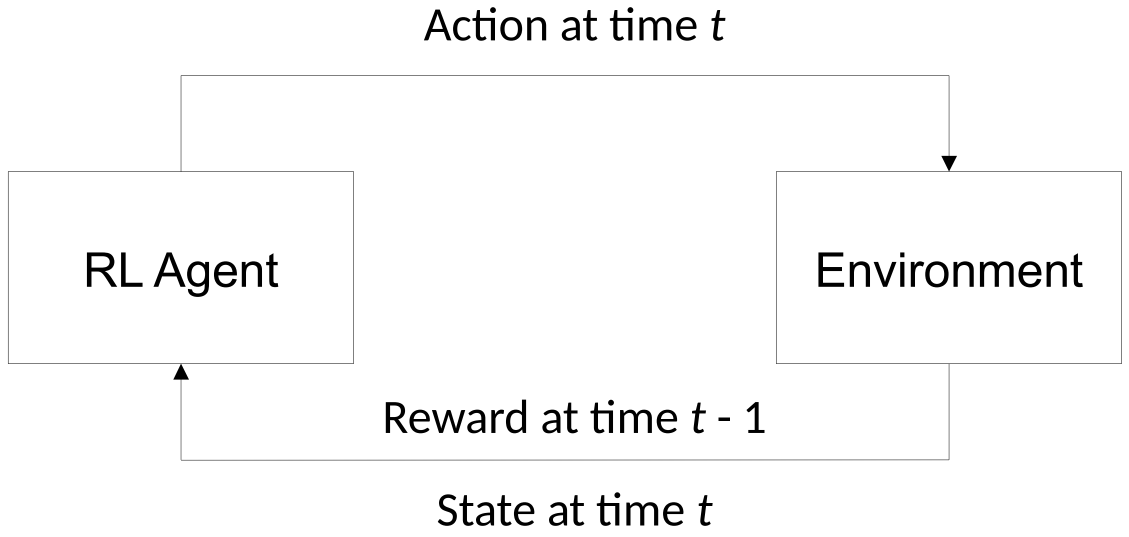
How?

Let the algorithm explore the environment on its own, while learning from experience

Reinforcement learning

Briefly:
What is it?

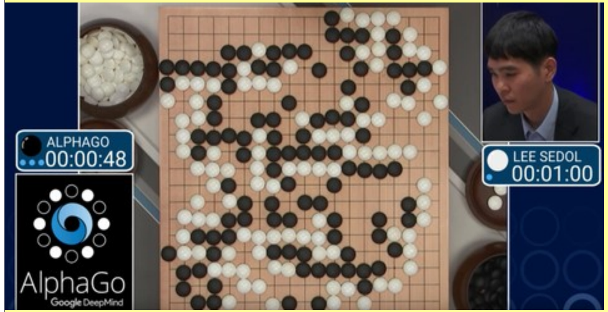
Learning to maximise long-term reward through interaction with the environment

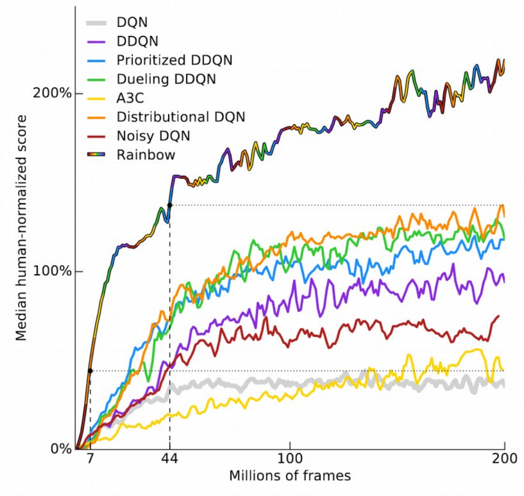


Context: Existing work

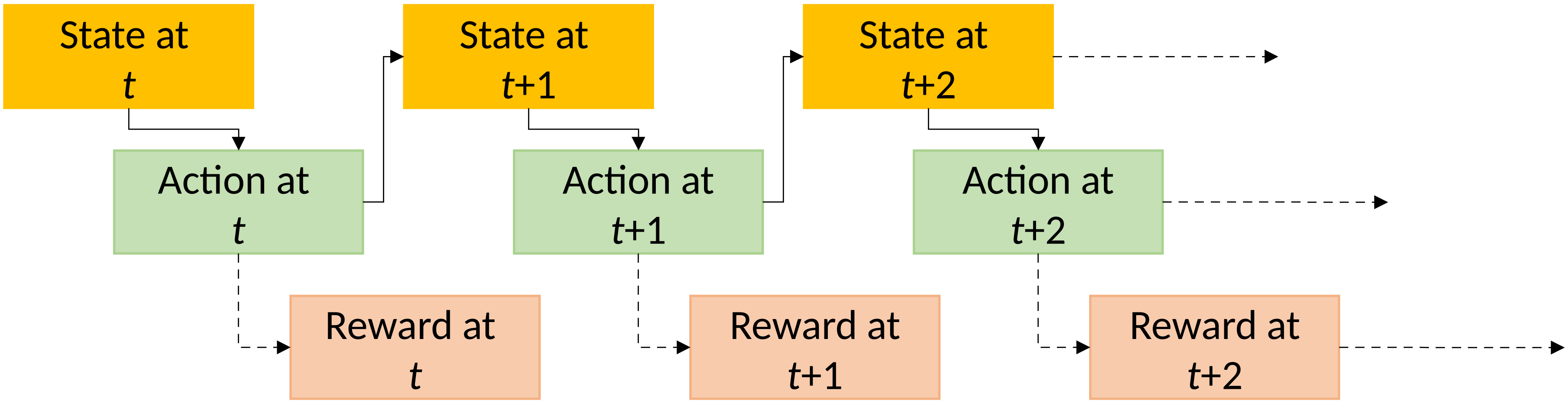
Is this a new idea? Not at all.

Aerospace: Adaptive control	Ops Research: Dynamic programming	Computer science: Reinforcement learning





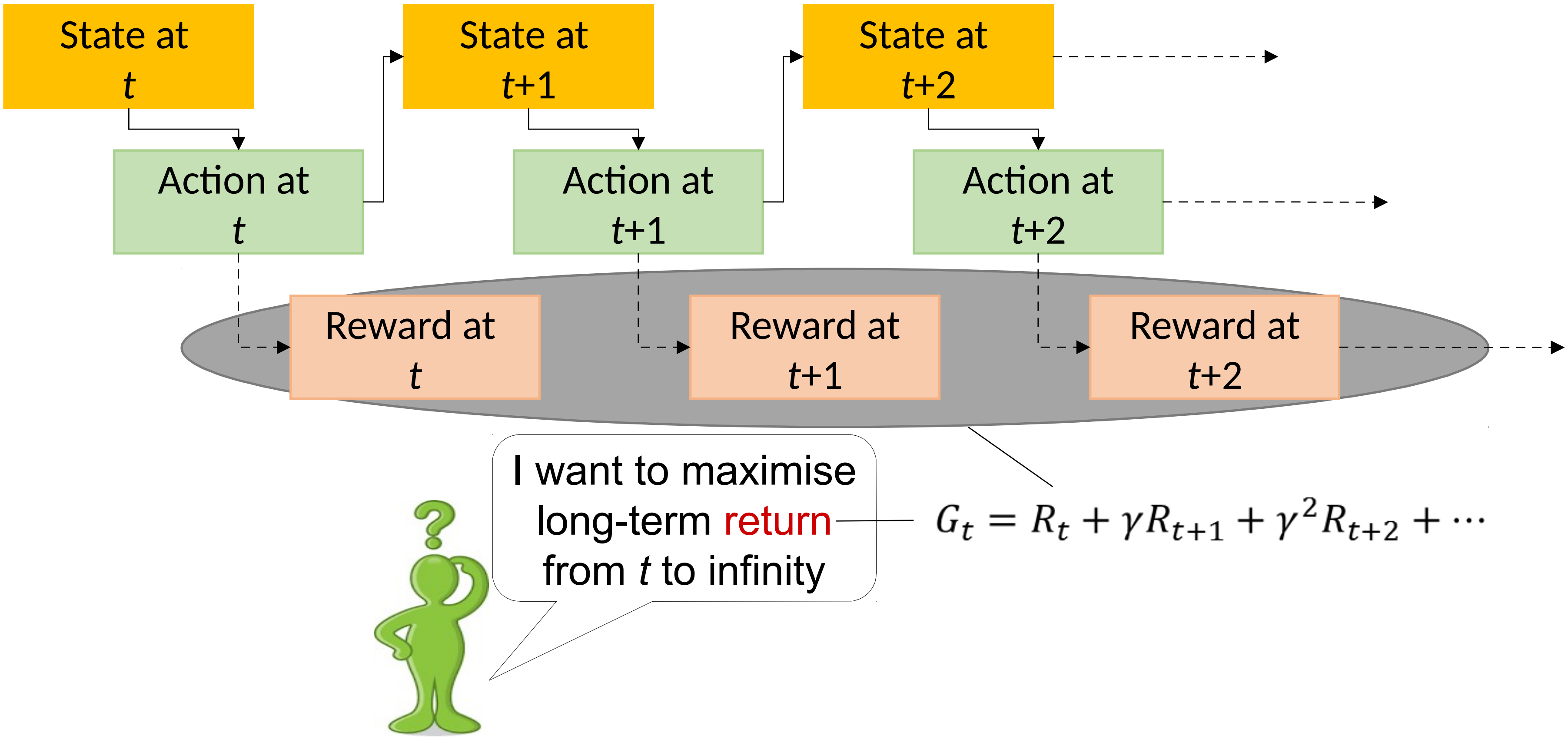
Anatomy of an RL problem



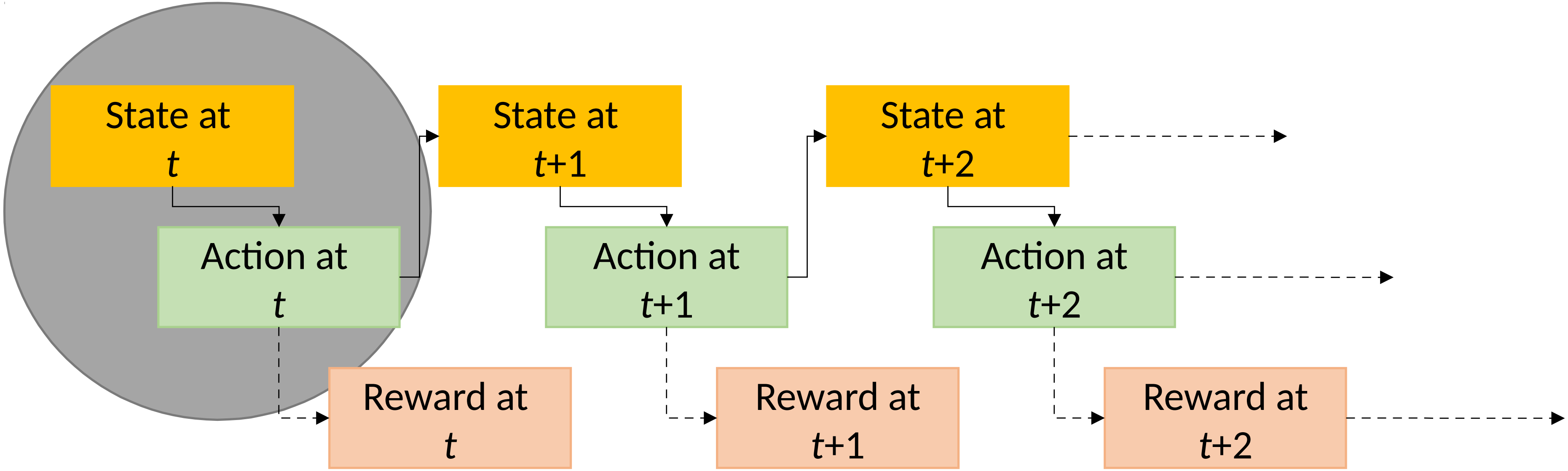
Strictly speaking, must be a Markov Decision Process defined by

(States, Actions, Rewards, Transitions, Discount factor)

Anatomy of an RL problem



Anatomy of an RL problem



Use neural network, with RHS of equation providing labels

Value-based **Deep RL**

Can be any function from (state, action) → scalar

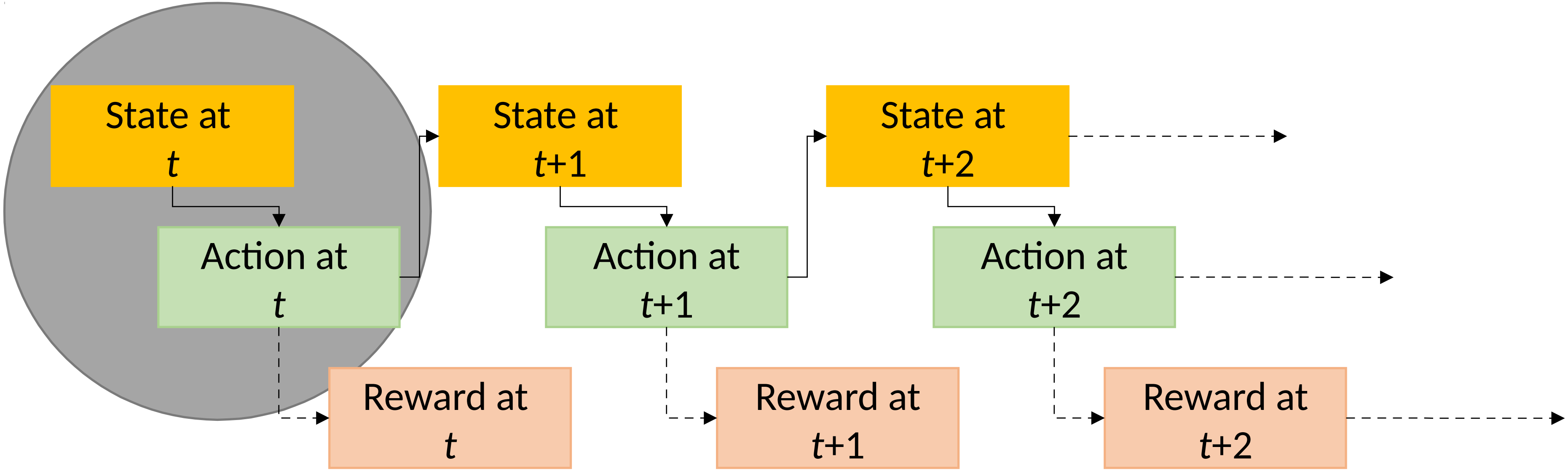
This is unknown at *t*, but is 'known' at *t+1*

All value based approaches

$$Q(s_t, a_t) = R_t + \gamma Q(s_{t+1}, a_{t+1})$$

3 How RL works

Anatomy of an RL problem



Use neural network, with gradients driving the training

Policy-based **Deep RL**

Gradient of expected reward with respect to each element of

Goal: Compute parameters θ that maximize reward

All policy based approaches

$$\Pi_\theta : s_t \to a_t$$

Alternative approach

3 How RL works

Practical challenges

The bad news

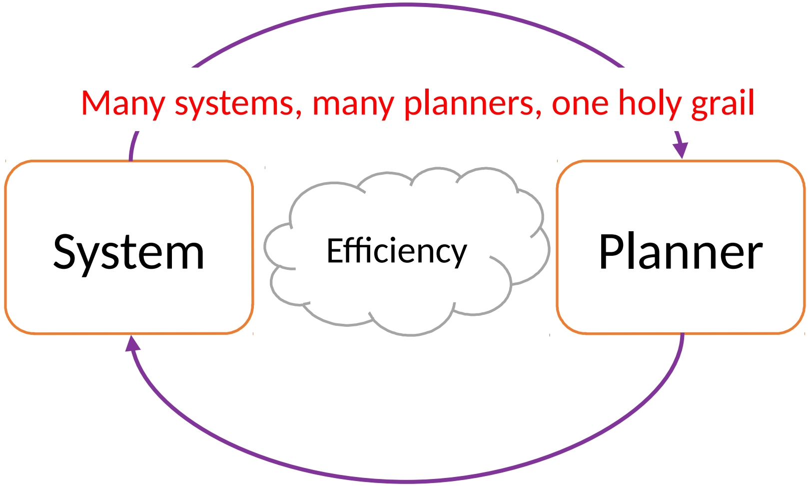
These ideas work brilliantly in games,
but not in real life

Why not?

1. Large scale
2. Variable scale
3. Complexity
4. Limited compute
5. Explainability requirement

RL in the real world

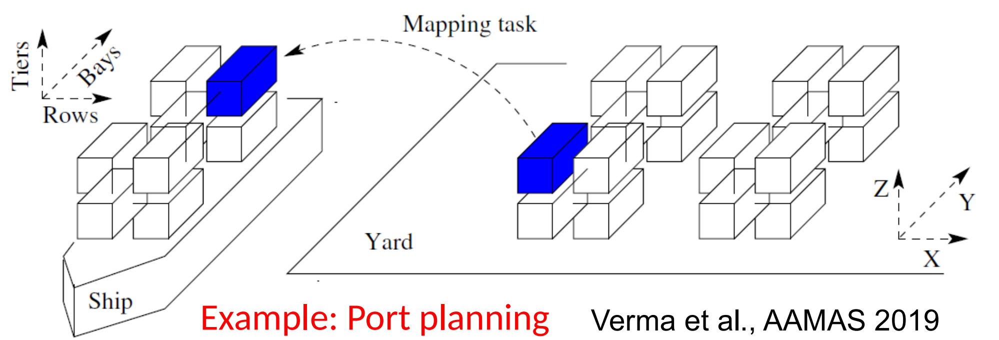
One-slide summary of past work



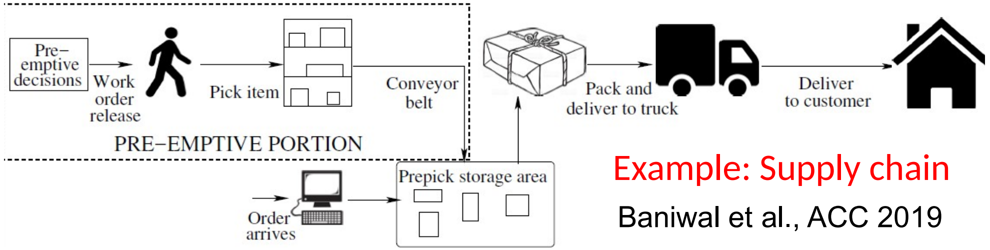
Problem: Systems do not operate in silos …

… but planners/controllers do

Goal: Build optimal planning & control algorithms that,
1. Operate in real-time (online)
2. Work without human-labelled historical data
3. Adapt automatically to changes in the environment

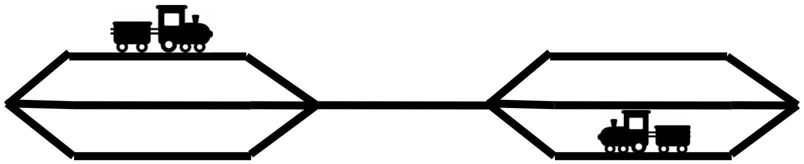


Example: Port planning Verma et al., AAMAS 2019



Example: Supply chain

Baniwal et al., ACC 2019



Example: Railway rescheduling Khadilkar, IEEE ITS 2019

Key takeaways from past work

1. Use domain knowledge
    - to divide the problem into a sequence of tasks
    - to define how system performance is measured

2. Define tasks that can be repeatedly performed to achieve goals (constant I/O size)

3. Build the right fidelity of simulation to compute the effect of actions on the system

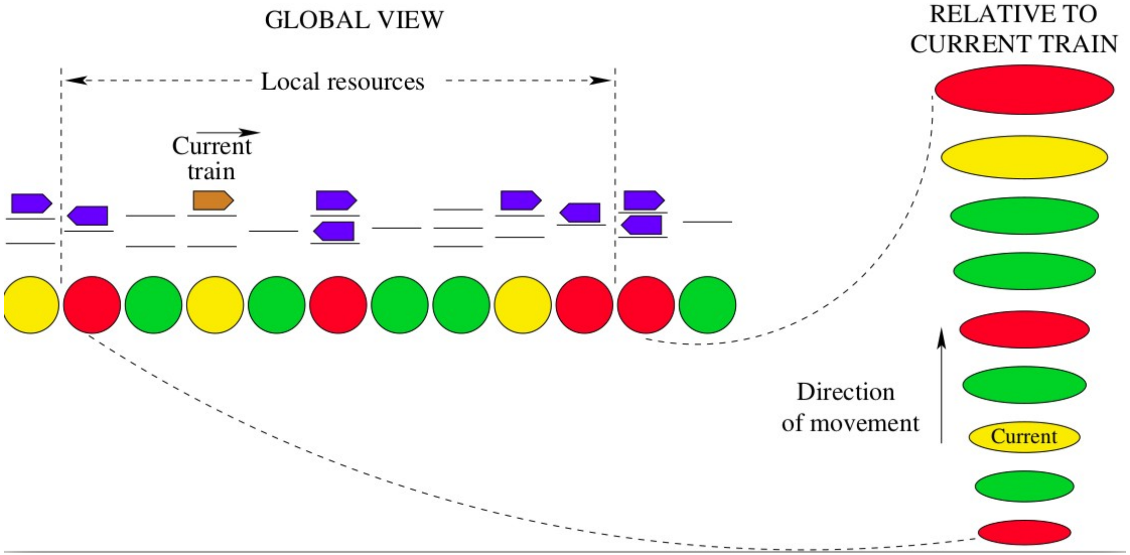
4. Use RL only for decisions where the ‘correct’ ones are not obvious

5. Wherever feasible, speed up RL training by seeding with existing heuristics

Concrete example: Railway scheduling

Goal: Minimise knock-on effects along the railway line, when recovering from a delayed state

Solution: Divide the problem into a sequence of moves

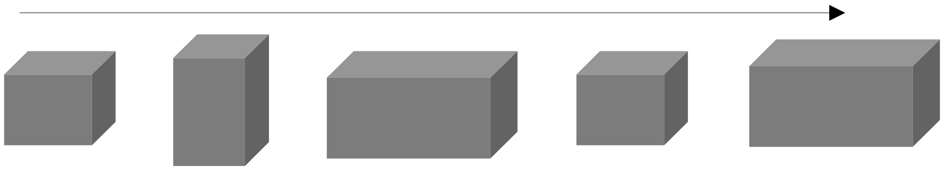


Current work

5 Current work
Planning for robotic parcel loading

ONLINE 3D BIN PACKING

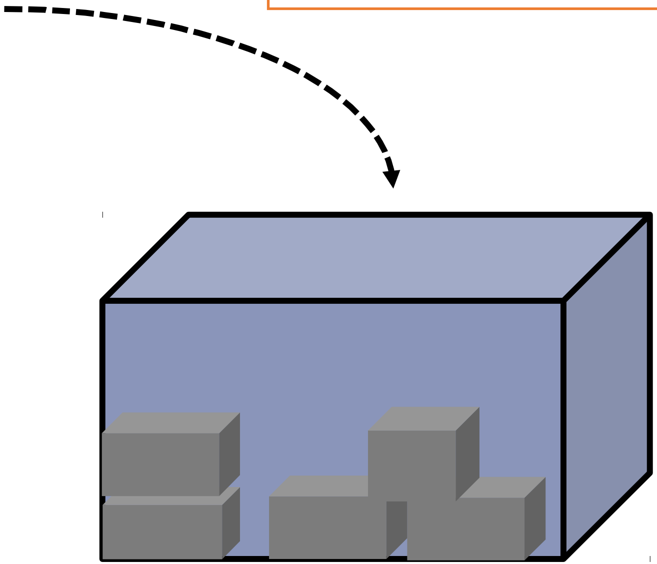
Stream of incoming boxes



Current box

Stable arrangement

Robot stackable



Container

Goal: Maximise the volume packed in containers, using boxes appearing on a conveyor belt

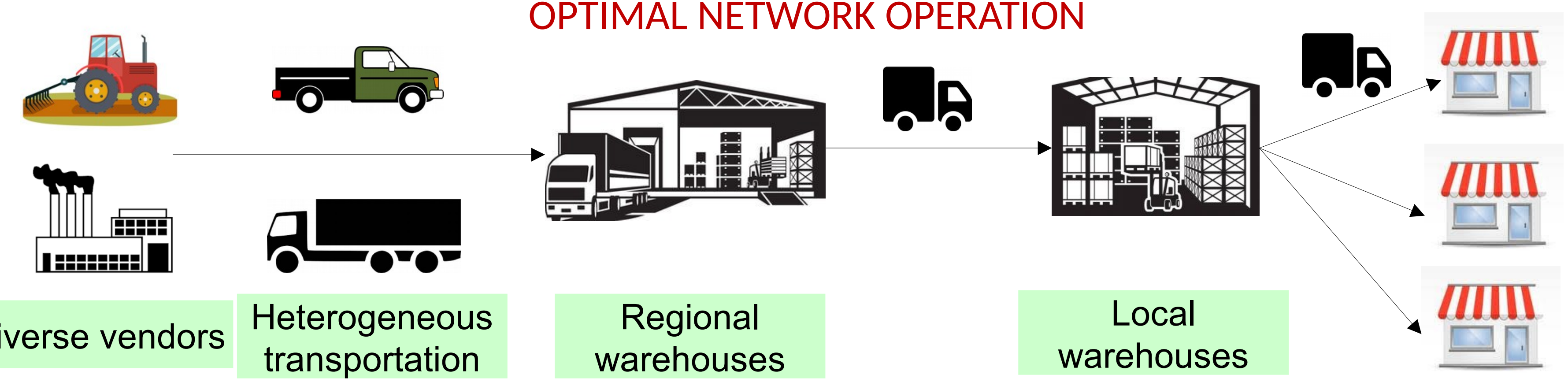
Rotate the box?

Where to place?

Skip current box?

Supply chain replenishment

OPTIMAL NETWORK OPERATION



Goal: Minimise supply chain operating costs while maximising key performance indicators

State of the art: Requirements flow from right to left (upstream), while products flow from left to right (downstream)

Solution: Multi-agent reinforcement learning at each node of the supply chain, for automated adaptive response to demands

Multi-Agent Reinforcement Learning (MARL)

Generalisation of Markov Decision Processes to Stochastic Games

Homogeneous Stochastic Games: (n, S, A, T, R, γ)

Number of agents

Transitions

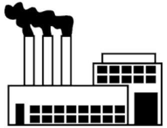
Discount factor

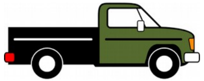
States

Actions

Rewards

Heterogeneous Stochastic Games:





Can this set of participants in a *system of systems* collaborate effectively?

5 Concluding remarks

Reinforcement learning = Use of machine learning for decision-making problems

Should be used when it is the best tool for the job:

1. Fast response
2. Systems *simulatable* but not analytically describable
3. Unknown ‘optimal’ decisions
4. Sequence-dependent rewards

Making RL work for you in real life:

1. Make sure you can simulate your problem, for training
2. Divide large problems into a sequence of repeated tasks
3. Use domain expertise rather than throw it away
4. Build solutions with explanations, not black boxes