CS 747, Autumn 2023: Lecture 20

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2023

Reinforcement Learning

1. Policy search
2. Case study 1: Humanoid robot soccer
3. Case study 2: Railway scheduling

Reinforcement Learning

1. Policy search
2. Case study 1: Humanoid robot soccer
3. Case study 2: Railway scheduling

Black Box Optimisation

- An abstract model:

$$\text{Input } w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow \text{Output } f(w).$$

Black Box Optimisation

- An abstract model:

$$\text{Input } w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow \text{Output } f(w).$$

For what input w is output $f(w)$ maximum?

Black Box Optimisation

- An abstract model:

$$\text{Input } w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow \text{Output } f(w).$$

  For what input w is output $f(w)$ maximum?

- System: chemical manufacturing plant.
  w: process parameters (temperature, ratio of chemical mixture, time duration, pressure, etc.)
  $f(w)$: product yield.

Black Box Optimisation

- An abstract model:

  $$\text{Input } w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow \text{Output } f(w).$$

  For what input w is output $f(w)$ maximum?

- System: chemical manufacturing plant.
  w: process parameters (temperature, ratio of chemical mixture, time duration, pressure, etc.)
  $f(w)$: product yield.

- System: Your MDP!
  w: parameters defining your policy.
  $f(w)$: expected long-term reward (as a scalar).

Black Box Optimisation

- An abstract model:

$$\text{Input } w = (w_1, w_2, \ldots, w_d) \longrightarrow \boxed{\text{System}} \longrightarrow \text{Output } f(w).$$

  For what input w is output $f(w)$ maximum?

- System: chemical manufacturing plant.
  w: process parameters (temperature, ratio of chemical mixture, time duration, pressure, etc.)
  $f(w)$: product yield.

- System: Your MDP!
  w: parameters defining your policy.
  $f(w)$: expected long-term reward (as a scalar).

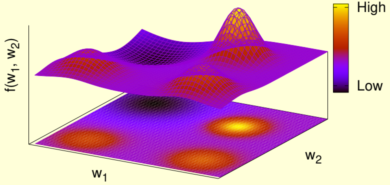
- Is finding the optimal w easy? Why is this approach called black box optimisation?

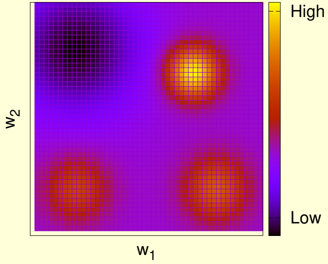
Typical Context for Black Box Optimisation

- Little/nothing known/assumed about f—can be discontinuous, non-linear, "erratic".
- Given w, evaluating $f(w)$ is relatively efficient.
- Calculating $f(w)$ usually involves a computer simulation.

Typical Context for Black Box Optimisation

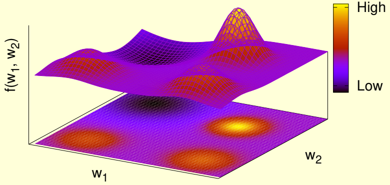
- Little/nothing known/assumed about f—can be discontinuous, non-linear, "erratic".
- Given w, evaluating $f(w)$ is relatively efficient.
- Calculating $f(w)$ usually involves a computer simulation.

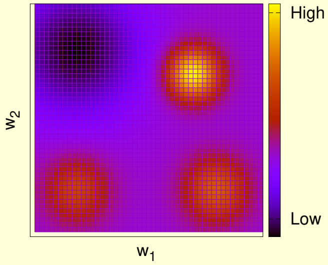




Typical Context for Black Box Optimisation

- Little/nothing known/assumed about f—can be discontinuous, non-linear, "erratic".
- Given w, evaluating $f(w)$ is relatively efficient.
- Calculating $f(w)$ usually involves a computer simulation.

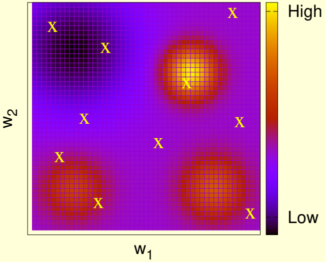




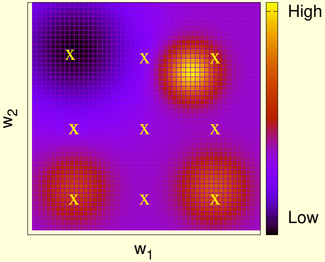
- How to find a "relatively good" w?

Some Natural Approaches

Random weight guessing

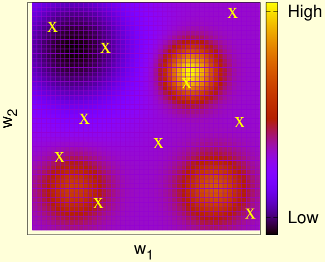


Grid search

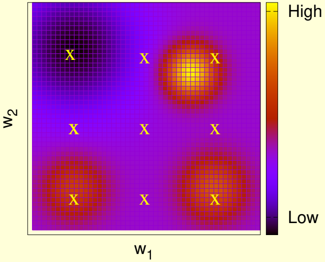


Some Natural Approaches

Random weight guessing



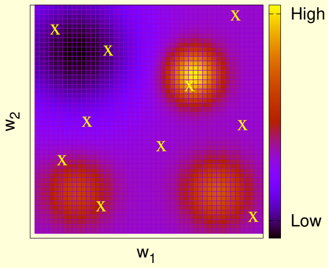
Grid search



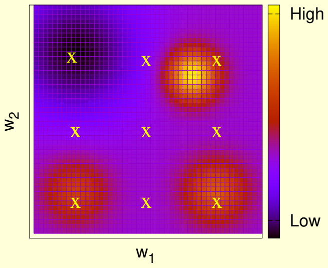
- These approaches work for small dimensions d (say $d \leq 5$).

Some Natural Approaches

Random weight guessing



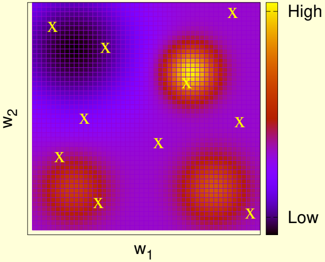
Grid search



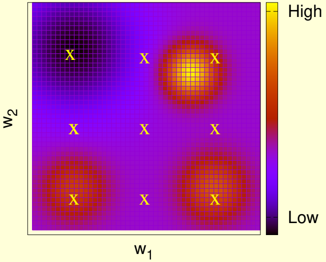
- These approaches work for small dimensions d (say $d \leq 5$).
- No method can be expected to work well for very large d (1000's or higher).

Some Natural Approaches

Random weight guessing



Grid search



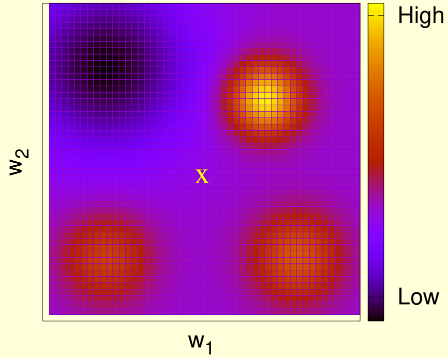
- These approaches work for small dimensions d (say $d \leq 5$).
- No method can be expected to work well for very large d (1000's or higher).
- Local search works for intermediate d (10's, 100's).

Local Search

- Illustrative method: Hill Climbing.

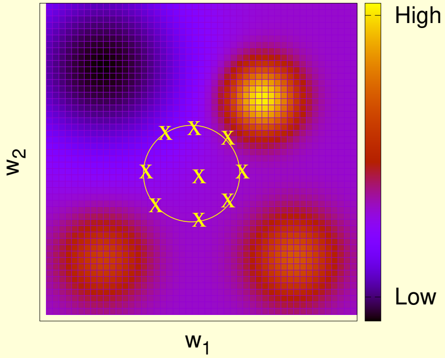
Local Search

- Illustrative method: Hill Climbing.



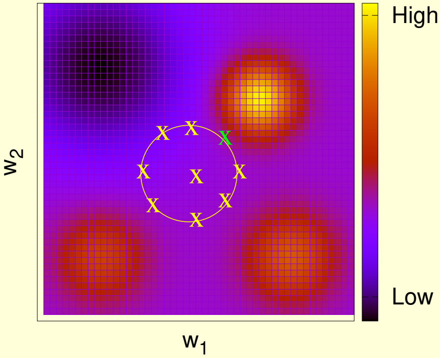
Local Search

- Illustrative method: Hill Climbing.



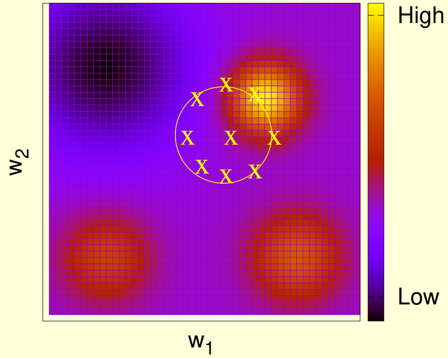
Local Search

- Illustrative method: Hill Climbing.



Local Search

- Illustrative method: Hill Climbing.



Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...

  No decided winner among these (depends on problem, not much guidance).

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...

  No decided winner among these (depends on problem, not much guidance).

- All instances of the generate and test paradigm.

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...

  No decided winner among these (depends on problem, not much guidance).

- All instances of the generate and test paradigm.

  − Ignores the structure of f.

  + Highly parallelisable.

  + Allow easy integration of domain knowledge.

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...

  No decided winner among these (depends on problem, not much guidance).

- All instances of the generate and test paradigm.

  $-$ Ignores the structure of f.

  $+$ Highly parallelisable.

  $+$ Allow easy integration of domain knowledge.

  Validation/successes primarily empirical; not much theoretical justification.

Local Search

- Several other local search variants: simulated annealing, ant-colony optimisation, particle swarm optimisation, evolutionary algorithms, ...

  No decided winner among these (depends on problem, not much guidance).

- All instances of the generate and test paradigm.

  − Ignores the structure of f.

  + Highly parallelisable.

  + Allow easy integration of domain knowledge.

  Validation/successes primarily empirical; not much theoretical justification.

- Called policy search when applied on the RL problem.

Reinforcement Learning

1. Policy search
2. Case study 1: Humanoid robot soccer

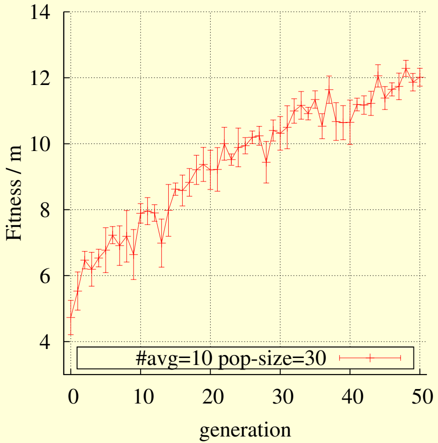
   Joint work with Patrick MacAlpine, Yinon Bentor, Daniel Urieli, and Peter Stone (UT Austin Villa robot soccer team).
3. Case study 2: Railway scheduling

Gait Optimisation: Policy Parameters

Notation	Description
maxStep^*_i	Maximum step sizes allowed for x, y, and θ
y^*_{shift}	Side to side shift amount with no side velocity
z^*_{torso}	Height of the torso from the ground
z^*_{step}	Maximum height of the foot from the ground
f^*_g	Fraction of a phase that the swing foot spends on the ground before lifting
f_a	Fraction that the swing foot spends in the air
f^*_s	Fraction before the swing foot starts moving
f_m	Fraction that the swing foot spends moving
ϕ^*_{length}	Duration of a single step
δ^*	Factors of how fast the step sizes change
y_{sep}	Separation between the feet
x^*_{offset}	Constant offset between the torso and feet
x^*_{factor}	Factor of the step size applied to the forwards position of the torso
err^*_{norm}	Maximum COM error before the steps are slowed
err^*_{max}	Maximum COM error before all velocity reach 0

Design and Optimization of an Omnidirectional Humanoid Walk: A WinningApproach at the RoboCup 2011 3D Simulation Competition. Patrick MacAlpine, Samuel Barrett, Daniel Urieli, Victor Vu, and Peter Stone. In Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012), AAAI Press, 2012.

Progress of Forward Speed Optimisation



On Optimizing Interdependent Skills: A Case Study in Simulated 3D Humanoid Robot Soccer. Daniel Urieli, Patrick MacAlpine, Shivaram Kalyanakrishnan, Yinon Bentor, and Peter Stone. In Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), pp. 769–776, IFAAMAS, 2011.

RoboCup 2011 3D Simulation Competition

- UT Austin Villa combined score: 136–0 (over 24 games).

Rank	Team	Goal Difference
3	apollo3d	1.45 (0.11)
5-8	boldhearts	2.00 (0.11)
5-8	robocanes	2.40 (0.10)
2	cit3d	3.33 (0.12)
5-8	fcportugal3d	3.75 (0.11)
9-12	magmaoffenburg	4.77 (0.12)
9-12	oxblue	4.83 (0.10)
4	kylinsky	5.52 (0.14)
9-12	dreamwing3d	6.22 (0.13)
5-8	seuredsun	6.79 (0.13)
13-18	karachikoalas	6.79 (0.09)
9-12	beestanbul	7.12 (0.11)

UT Austin Villa 2011: A Champion Agent in the RoboCup 3D Soccer Simulation Competition. Patrick MacAlpine, Daniel Urieli, Samuel Barrett, Shivaram Kalyanakrishnan, Francisco Barrera, Adrian Lopez-Mobilia, Nicolae Ştiurcă, Victor Vu, and Peter Stone. In Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), pp. 129–136, IFAAMAS, 2012.

Reinforcement Learning

1. Policy search

2. Case study 1: Humanoid robot soccer

3. Case study 2: Railway scheduling

   Joint work with Rohit Prasad, Harshad Khadilkar.

Railways and the Economy

[1]

- Indian Railways, 20000+ trains, 9.1 billion yearly ridership.
- Delay incurs economic costs

[1] https://pixabay.com/photos/transportation-system-travel-vehicle-3351330/.

Railway Rescheduling Problem

Train	Station	Timetable Arrival Time	Timetable Departure Time	Minimum Halt Time	Minimum Run Time	Scheduled Arrival Time	Scheduled Departure Time	Delay
101	Alpha	4:30	4:40	10	30			
102	Alpha	4:20	4:40	20	10			
601	Echo	9:15	9:16	1	60			
401	Delta	3:20	3:35	15	20			

Given an initial timetable
compute a feasible timetable
subject to resource allocation and time constraints
minimising

Priority-weighted departure delay

$$= \sum_{\text{Train } t, \text{ Resource } r} \frac{\text{Delay of train } t \text{ at resource } r}{\text{Priority of train } t}.$$

Railway Rescheduling Problem

Train	Station	Timetable Arrival Time	Timetable Departure Time	Minimum Halt Time	Minimum Run Time	Scheduled Arrival Time	Scheduled Departure Time	Delay
101	Alpha	4:30	4:40	10	30			
102	Alpha	4:20	4:40	20	10			
601	Echo	9:15	9:16	1	60			
401	Delta	3:20	3:35	15	20			

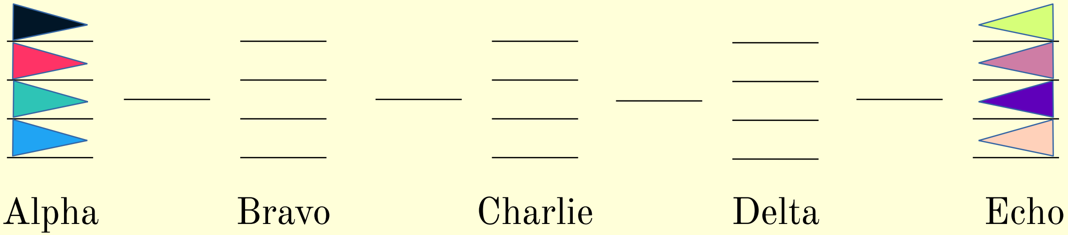
Given an initial timetable
compute a feasible timetable
subject to resource allocation and time constraints
minimising

Priority-weighted departure delay

$$= \sum_{\text{Train } t, \text{ Resource } r} \frac{\text{Delay of train } t \text{ at resource } r}{\text{Priority of train } t}.$$

Illustration

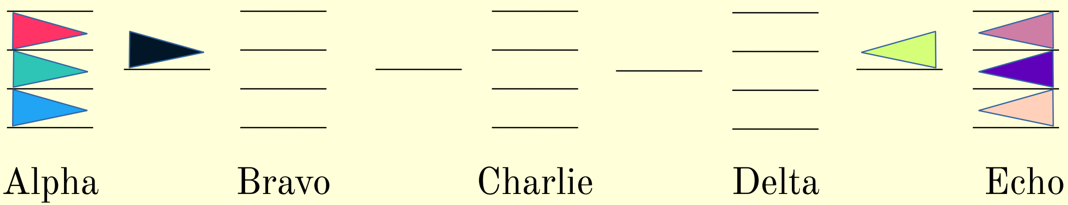
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

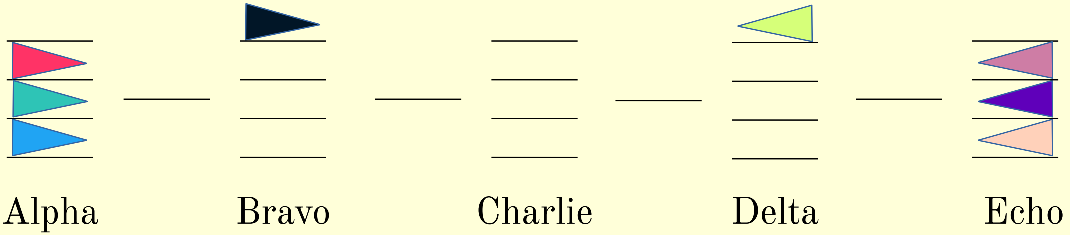
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

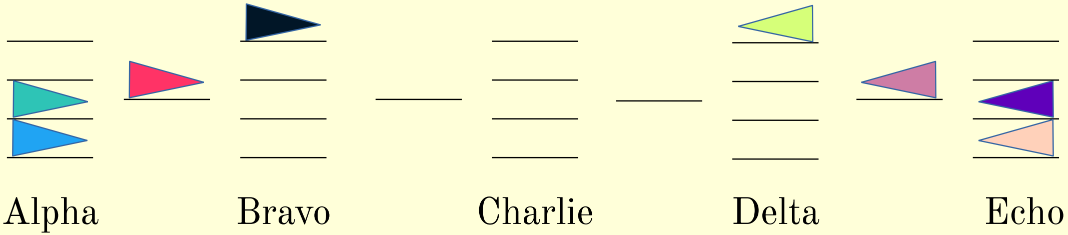
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

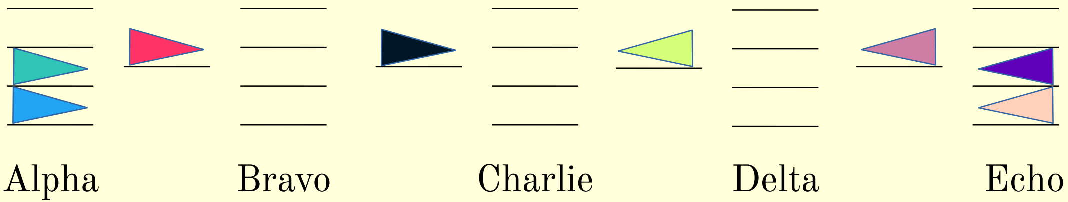
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

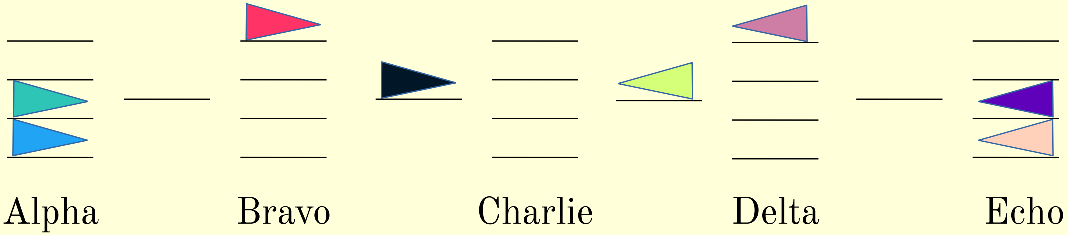
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

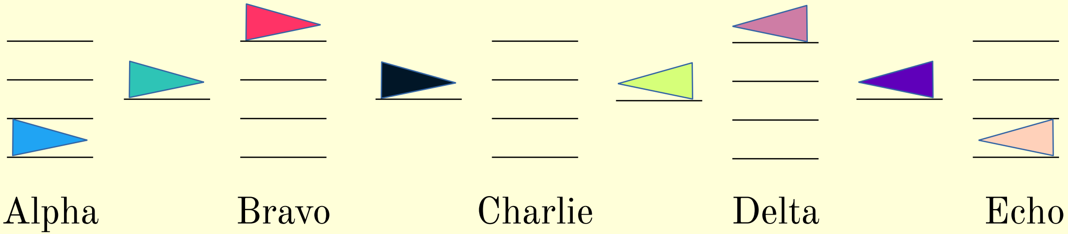
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

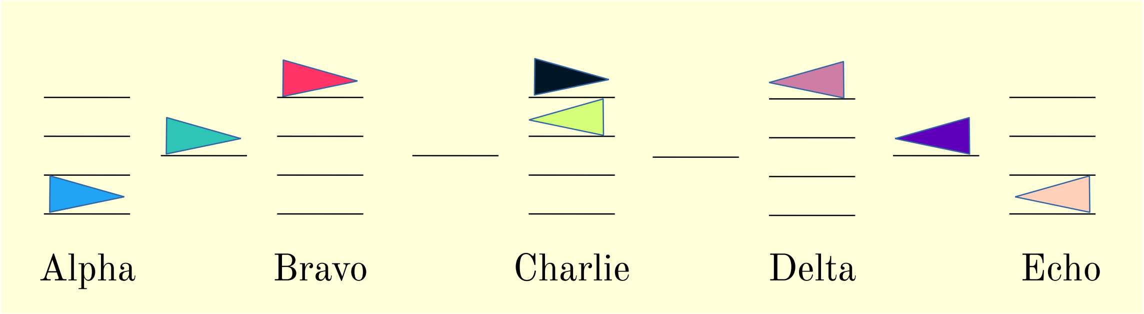
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

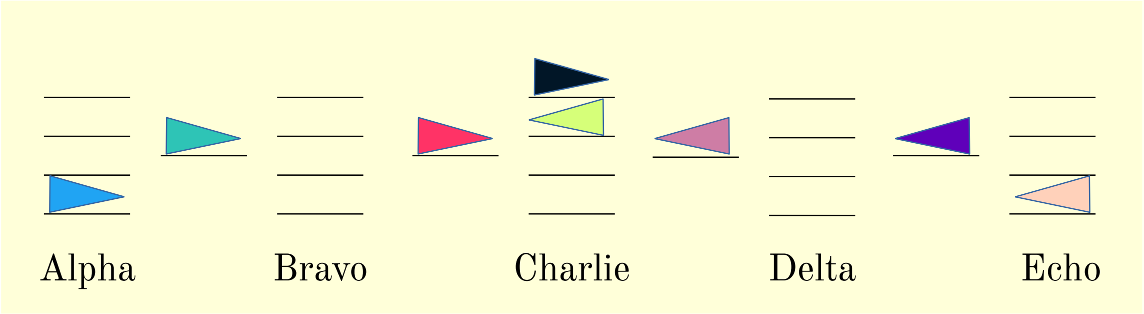
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

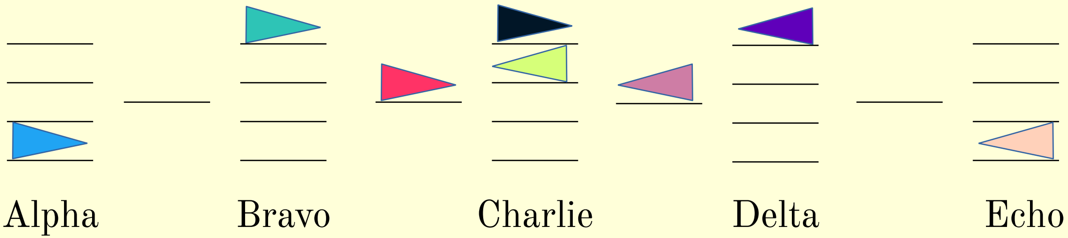
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

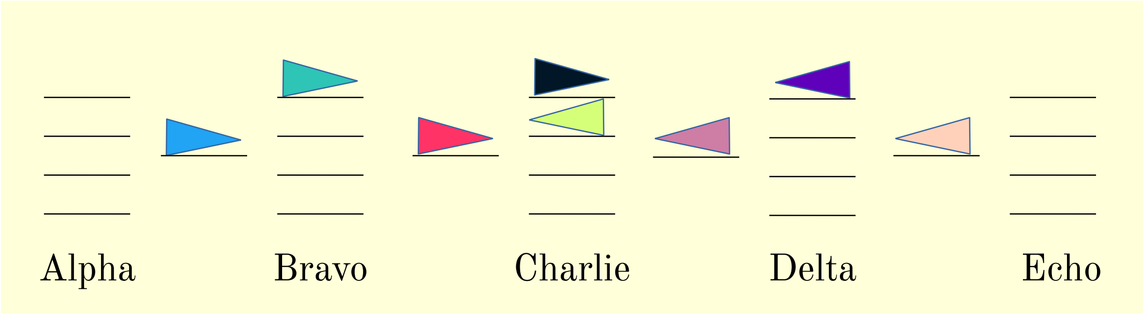
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

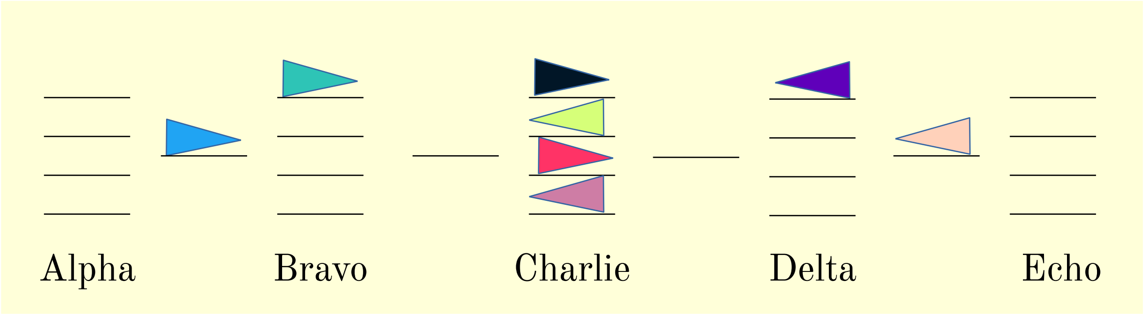
A hypothetical railway line with 8 trains and 5 stations



Moving trains

Illustration

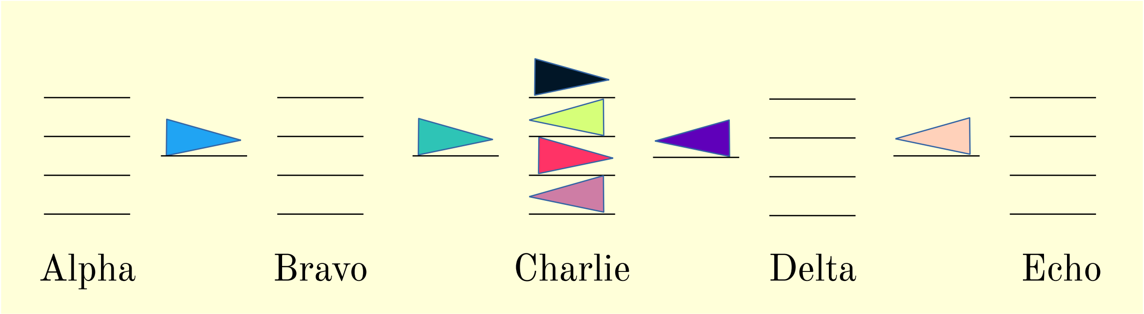
A hypothetical railway line with 8 trains and 5 stations



Intermediate state

Illustration

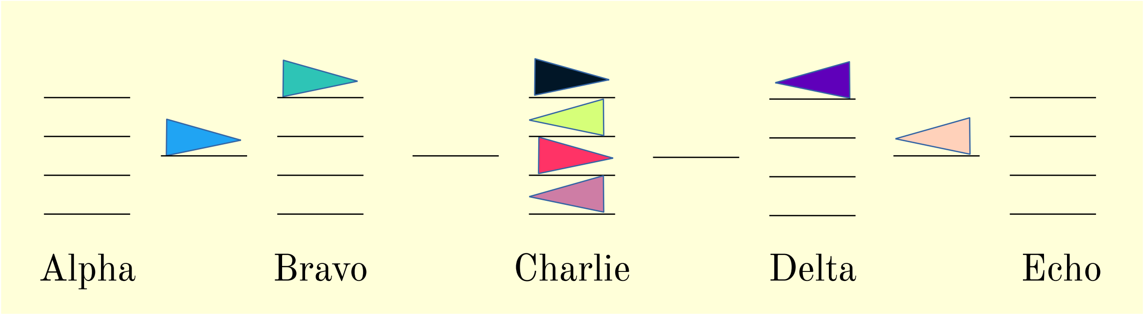
A hypothetical railway line with 8 trains and 5 stations



Deadlock!

Illustration

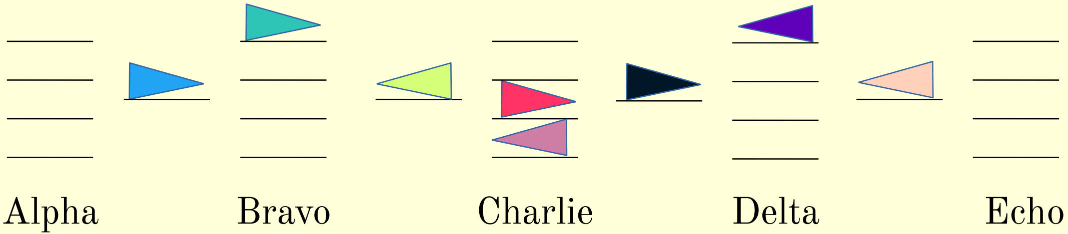
A hypothetical railway line with 8 trains and 5 stations



Intermediate state

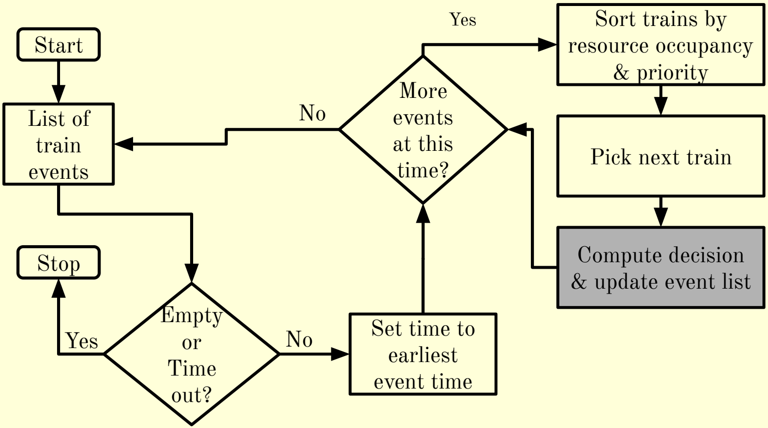
Illustration

A hypothetical railway line with 8 trains and 5 stations



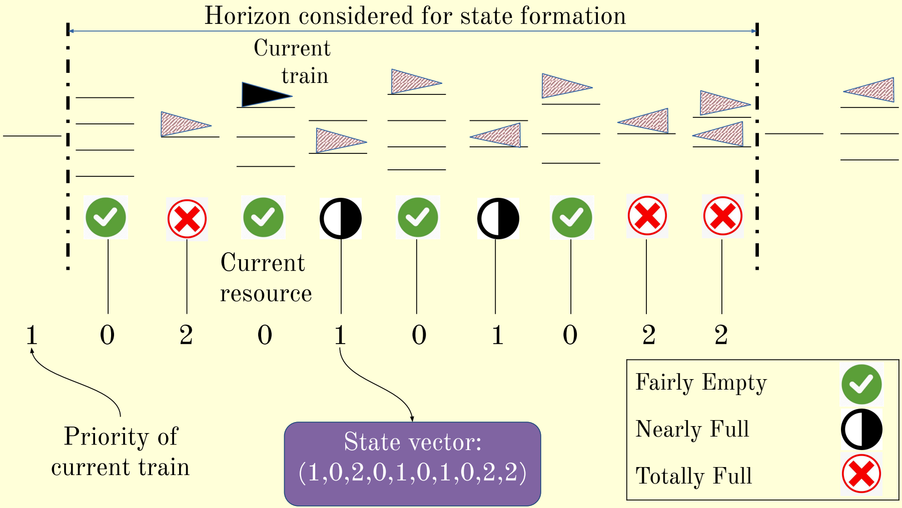
No Deadlock!

Our Solution



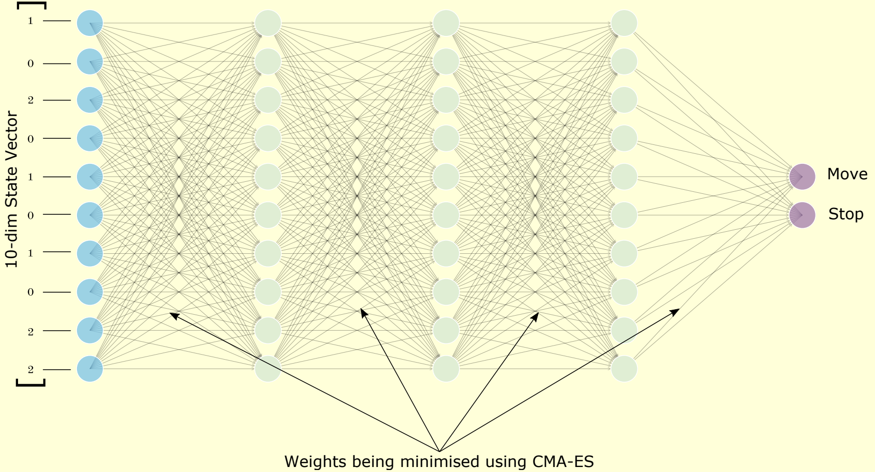
- The wrapper layer picks a potential train to move.
- We optimise the module that decides MOVE / STOP.

State Space



Horizon considered for state formation
Current
train
Current
resource
1 0 2 0 1 0 1 0 2 2
Priority of
current train
State vector:
(1,0,2,0,1,0,1,0,2,2)
Fairly Empty
Nearly Full
Totally Full

Policy Representation

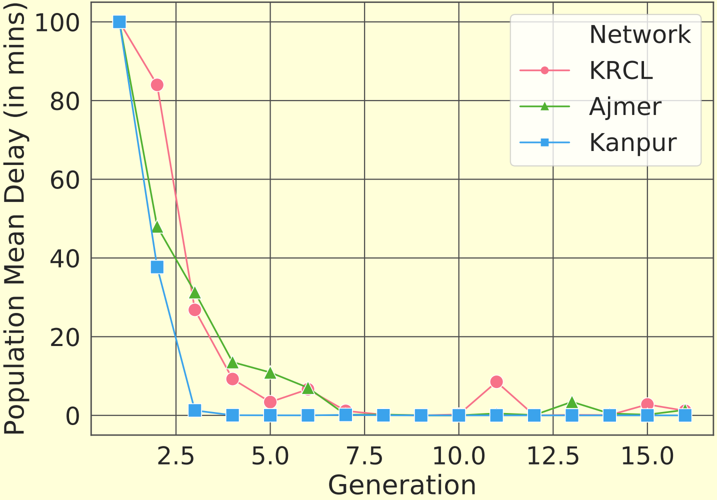


$$d = 352.$$

Benchmark Railway Lines

Scenario	Type	Stations	Trains	Events	Timetable span
HYP-2	Line	11	60	1320	4 hours
HYP-3	Line	11	120	2640	7 hours
KRCL	Line	59	85	5418	3 days
Kanpur	Line	27	190	7716	3 days
Ajmer	Line	52	444	26258	7 days

Results


Population Mean Delay (in mins)
100
80
60
40
20
0
2.5
5.0
7.5
10.0
12.5
15.0
Generation
Network
KRCL
Ajmer
Kanpur

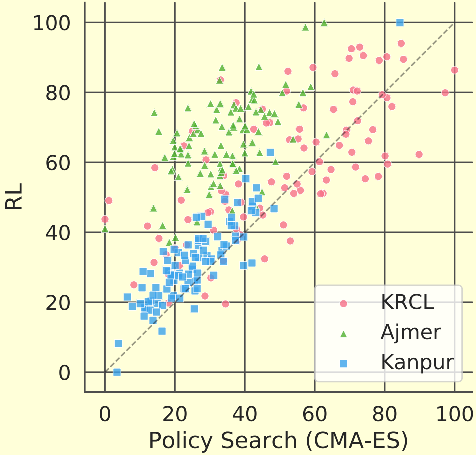
Summary of Results

Priority-weighted departure delay.

	Policy search	RL	TAH-FP	TAH-CF	Naive	GWP	PTD
HYP-2	**4.28 (0)**	4.78 (0)	4.58 (0)	5.93 (0)	11.16 (2)	4.35 (0)	714.00 (0)
HYP-3	**15.50 (0)**	18.54 (0)	61.89 (97)	140.14 (95)	- (100)	16.35 (0)	2003.98 (0)
KRCL	**42.34 (0)**	43.04 (0)	46.41 (8)	47.02 (0)	- (100)	42.40 (0)	4714.08 (0)
Ajmer	**3.92 (0)**	4.65 (0)	10.76 (3)	5.99 (0)	9.25 (76)	3.99 (3)	8304.84 (0)
Kanpur	**1.54 (0)**	1.66 (0)	2.19 (0)	2.28 (0)	1.85 (0)	1.54 (0)	313.60 (0)

Comparison with RL (Q-learning)

Priority-weighted departure delay.



Summary

- Initial lectures assumed finite state spaces.

  Seldom seen in practice!
- Need to generalise over states (sometimes actions).

  Function approximation has many empirical successes, yet is often problematic, especially for control.
- Quality of features/representation determines the validity of Markovian assumption.
- Policy search ignores Markovian structure—which sometimes works to its advantage!

  Conceptually simple; also has many empirical successes.

Summary

- Initial lectures assumed finite state spaces.

  Seldom seen in practice!
- Need to generalise over states (sometimes actions).

  Function approximation has many empirical successes, yet is often problematic, especially for control.
- Quality of features/representation determines the validity of Markovian assumption.
- Policy search ignores Markovian structure—which sometimes works to its advantage!

  Conceptually simple; also has many empirical successes.

- **Next class**: policy gradient methods.