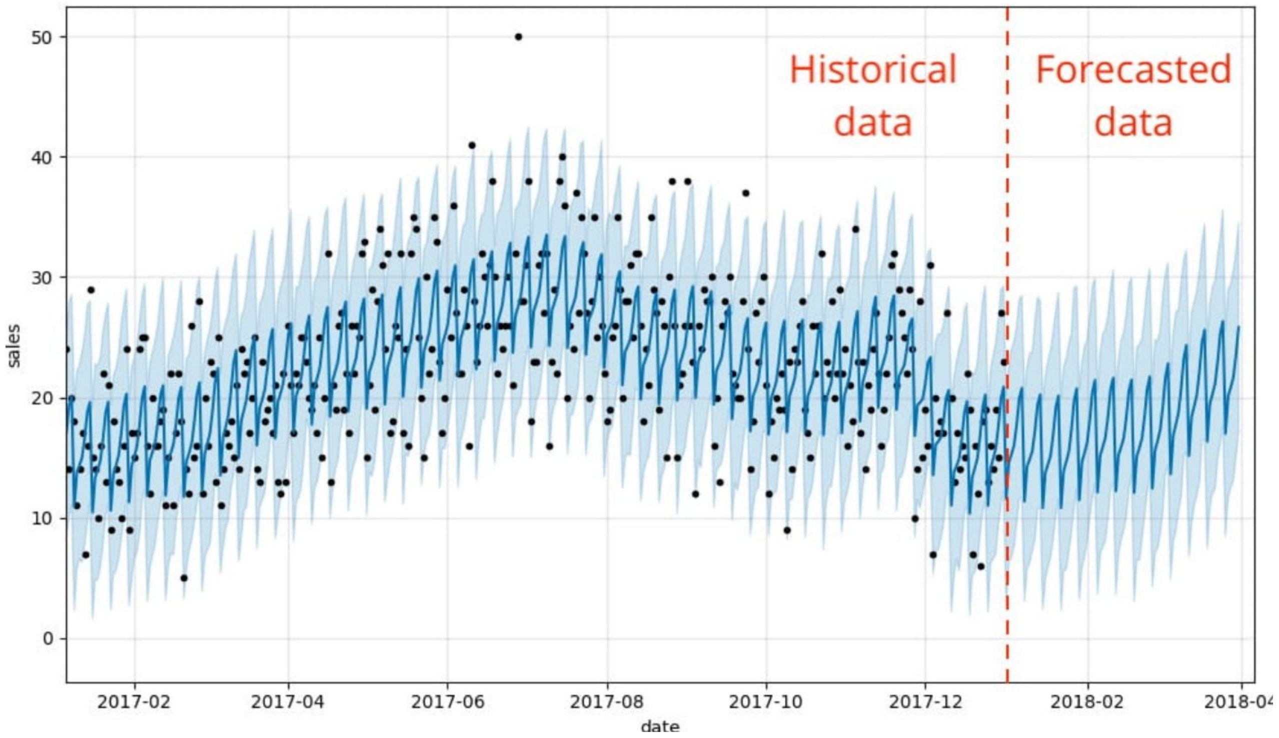
Approximation Algorithms for the Routing and Wavelength Assignment problem — Preliminaries

Jesse Hellendoorn, 16 Dec 2022

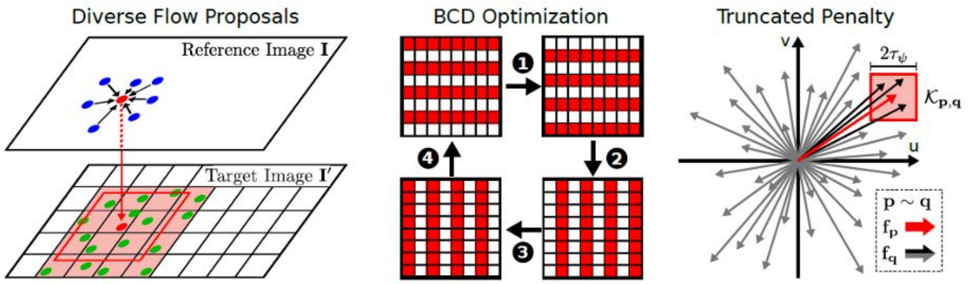
Preliminaries

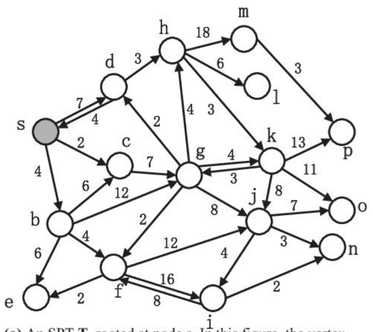
1. Econometrics: Time series and Regression
2. Operations Research: Combinatorial Optimization
3. Non-deterministic polynomial problems: O(n)
4. Approximation bounds: α*OPT

Econometrics

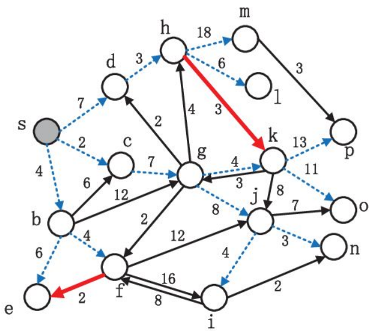


Operations Research

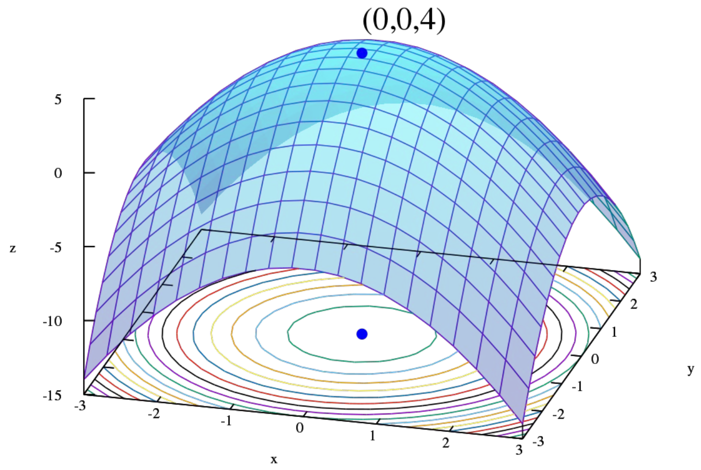

Diverse Flow Proposals
Reference Image $\mathbf{I}$
Target Image $\mathbf{I}'$
BCD Optimization
❶
❷
❸
❹
Truncated Penalty
v
u
$2\tau_\psi$
$\mathcal{K}_{\mathbf{p},\mathbf{q}}$
$\mathbf{p} \sim \mathbf{q}$
$\mathbf{f_p}$
$\mathbf{f_q}$



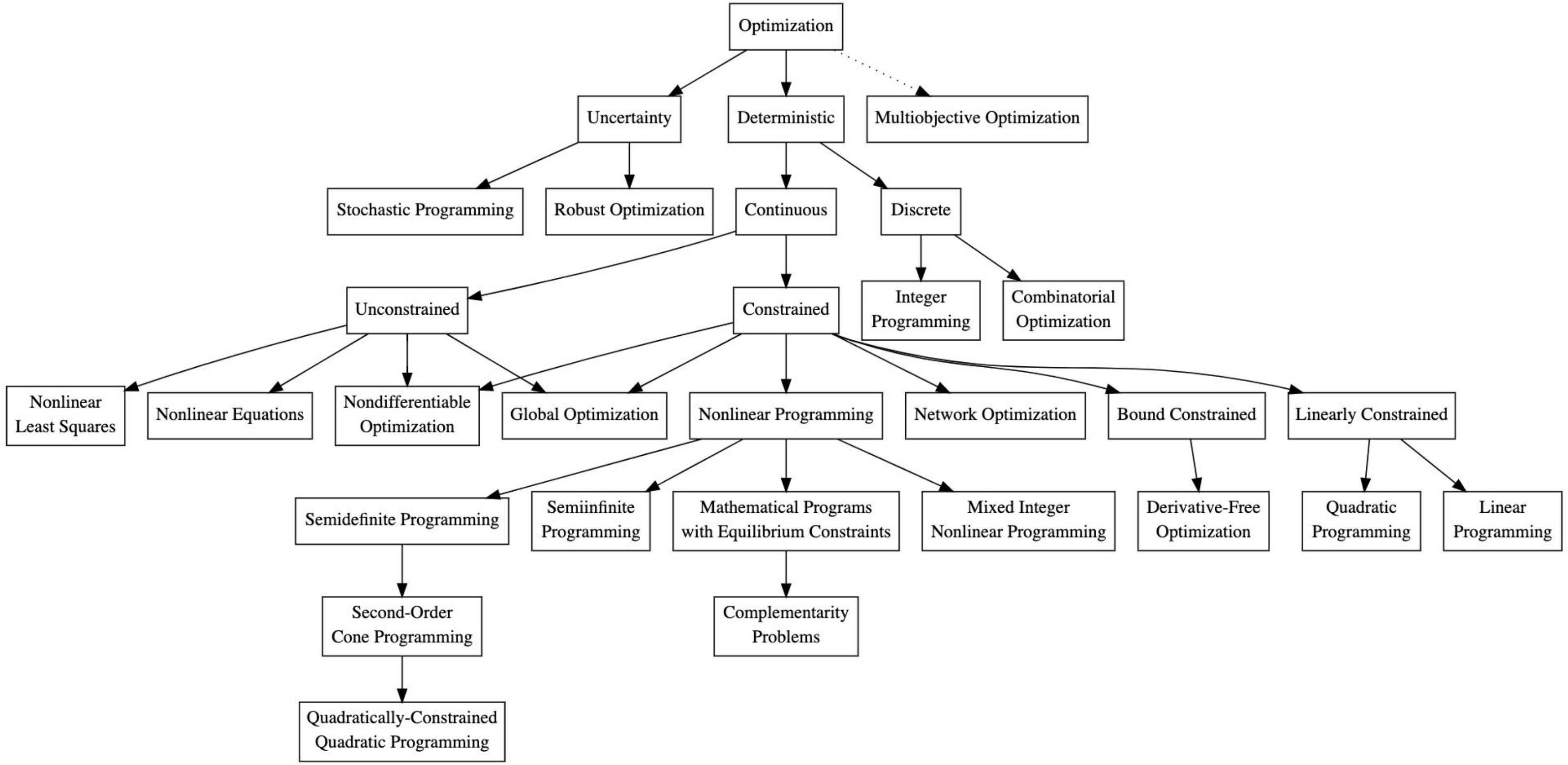
(a) An SPT $\mathbf{T}_s$ rooted at node s. In this figure, the vertex are represented by letters and the numbers along the arcs denote the path length. The arrow from the tail to the head means the direction of the edge.



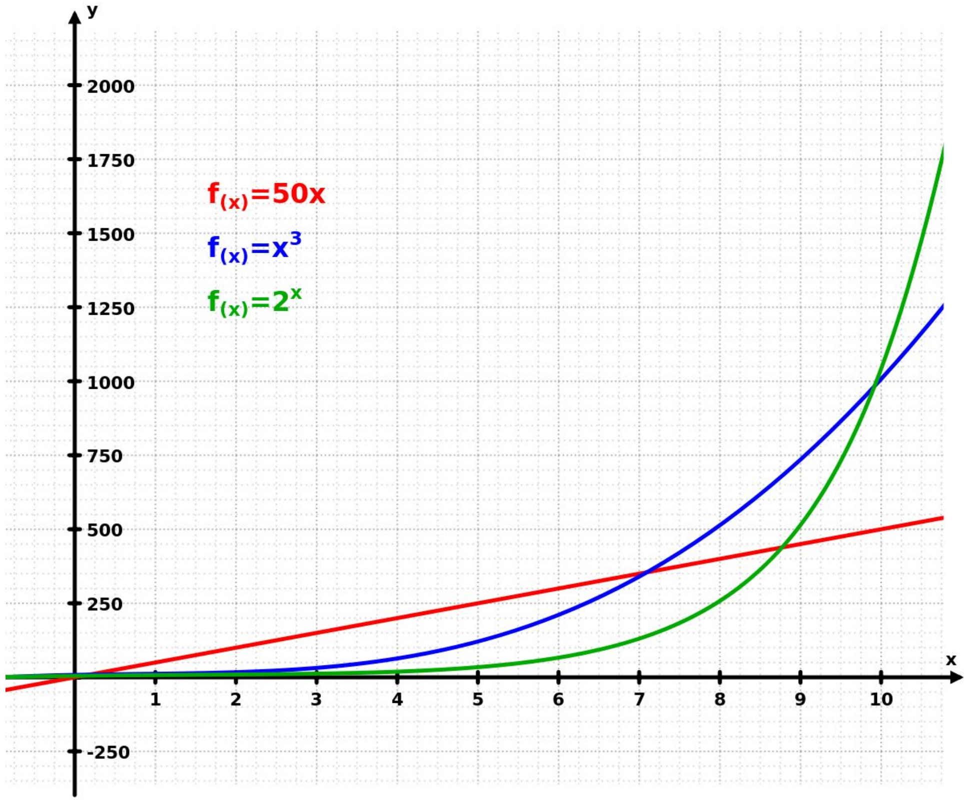
(b) The SPTs rooted at node s constructed by the adaptive amoeba algorithm.



Discrete Optimization

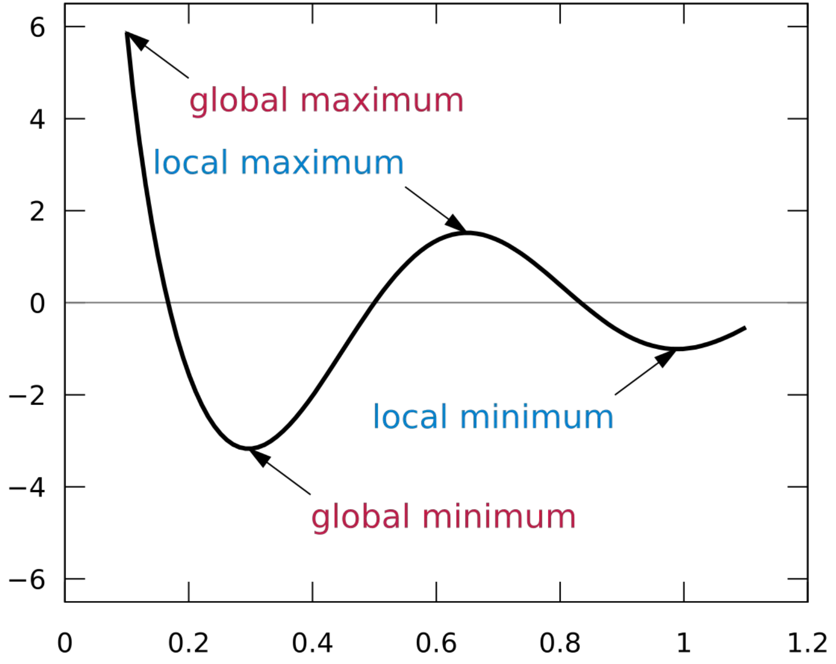


Non-deterministic polynomial problems: O(n)



What happens when we set x to a million?	
50*(10^6)	5e+6
180*(10^6)	18e+6
(10^6)^3	1e+18
(10^6)^5	1e+30
2^(10^6)	1e+301029
Atoms in universe: 1e+82; P vs NP	

Approximation bounds: α*OPT



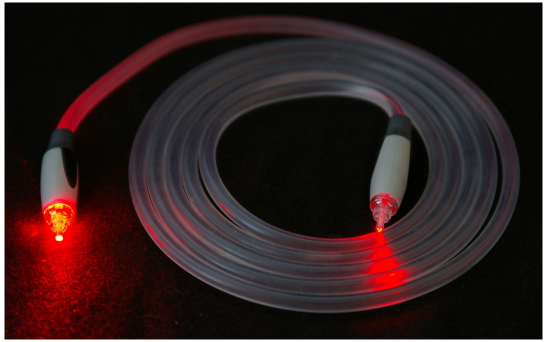
End of preliminaries

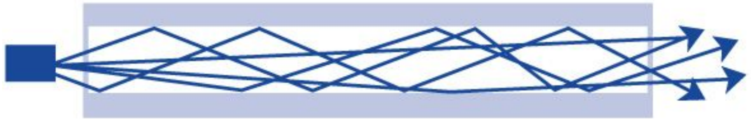
Approximation Algorithms for the Routing and Wavelength Assignment problem — Presentation

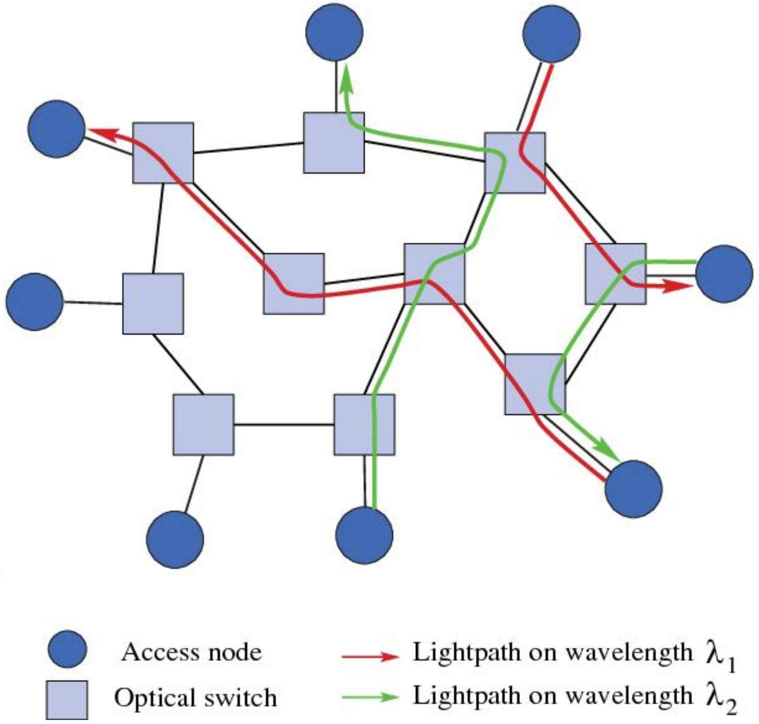
Jesse Hellendoorn, 16 Dec 2022

Contents

The routing and wavelength assignment problem






Access node
Optical switch
Lightpath on wavelength λ_1
Lightpath on wavelength λ_2

Problem formulation

We have a graph G=(V, E) with vertices V and edges E

For some generated set of n demands $K=\{k_1,k_2, \dots , K_n\}$
construct for each demand k a pair of vertices (v_s, v_d) from V

Let $P=\{p_1,p_2, \dots , p_n\}$ be the set of paths in G. A path is a sequence of consecutive edges e from E. A tuple k can use one or more paths from its source v_s to its destination v_d. Let $\Lambda=\{\lambda_1,\lambda_2, \dots , \lambda_n\}$ be the set of wavelengths. A lightpath is a pair (λ, p) corresponding to some wavelength and a path given some demand.

Given K, we seek to maximize the amount of concurrent demands that can be supported in the network while respecting forthcoming constraints.

Problem formulation

Constraint 1: Distinct wavelength assignment (clash)

‘Every edge e from E can at most support 1 demand for every wavelength’

Constraint 2: Wavelength continuity

‘Every path for a given demand can only be assigned to a single wavelength’

Note that a lightpath automatically fulfills this constraint (λ, p)

Optimization function:

Let K^+ be the number of accepted demands with corresponding paths P^+

Let $|E_{p+}|$ be a amount of edges used to construct P^+, this is the sum of path lengths

Then consider $\max \sum_{k \in K^+} x_k + \frac{1}{1 + |E_{p^+}|}$ our optimization function

Problem formulation

Research question:
What approximation ratio can we achieve for the Routing and Wavelength Assignment problem?

- Can reinforcement learning have a contribution to the approximation ratio?
- How close does our estimated upper bound come to estimated smallest upper bound?

In report:

- What is the most appropriate amount of wavelengths given specific demand and graph?
- What is the correlation of the amount of traversals per edge and the edge betweenness?
- What is the correlation of the demand acceptance rate per endpoint and their eigenvector centrality?

Simulation assumptions and configuration

Assumption 1: We consider multicast connections

Assumption 2: Network edges support demand symmetrically and bidirectionally

Assumption 3: Network vertices are gateways

Assumption 4: Demand not supported is lost, not queued or stored $K \backslash K^{+} = \varnothing$

Assumption 5: We consider the static routing and wavelength problem

Assumption 6: Our solution uses path/wavelength establishment decomposition

Assumption 8: Circuits and loops are invalid $v_s \neq v_d$

Simulation assumptions and configuration

Assumption 9: Edges represent unit distance single-fiber optic links

Assumption 10: Graphs are fully connected

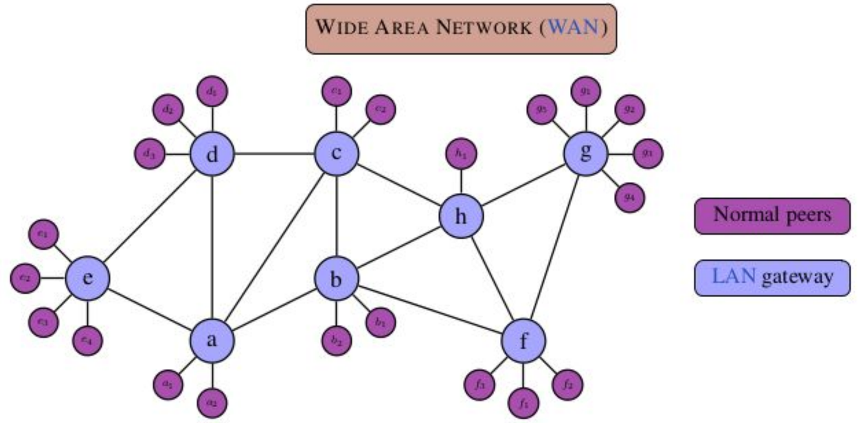
Assumption 11: Demands are continuous, unit time (no delay) and not faultless

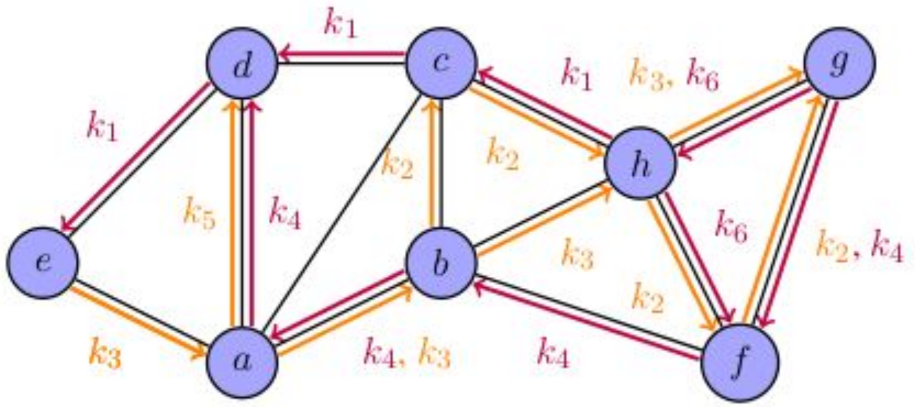
Assumption 12: Edge capacities are homogeneous and constant

Assumption 13: We consider a shortest path depth of 25 paths at most

Assumption 14: $P \neq NP$ and the routing and wavelength assignment problem is NP-Complete (ILP yields optimum of RWA and vise versa)

Simulation assumptions and configuration





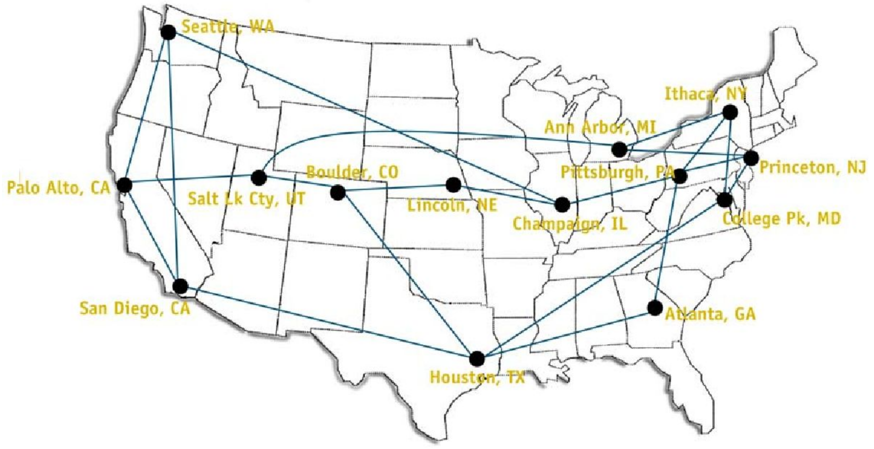
We have 2 wavelengths, 6 demands:

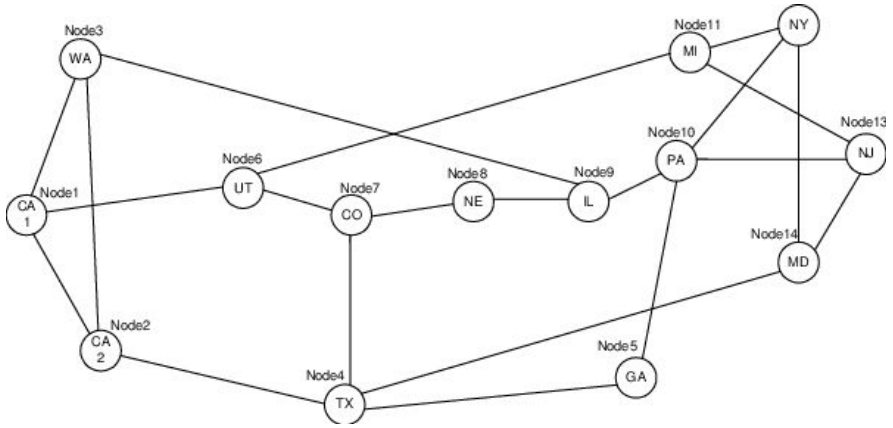
1. h-e (h, c, d, e) with lambda 1
2. b-g (b, c, h, f, g) with lambda 2
3. e-g (e, a, b, h, g) with lambda 2
4. g-d (g, f, b, a, d) with lambda 1
5. a-d (a, d) with lambda 1
6. g-f (g, h, f) with lambda 1

Simulation assumptions and configuration

We use N=100 simulations for 12 network topologies

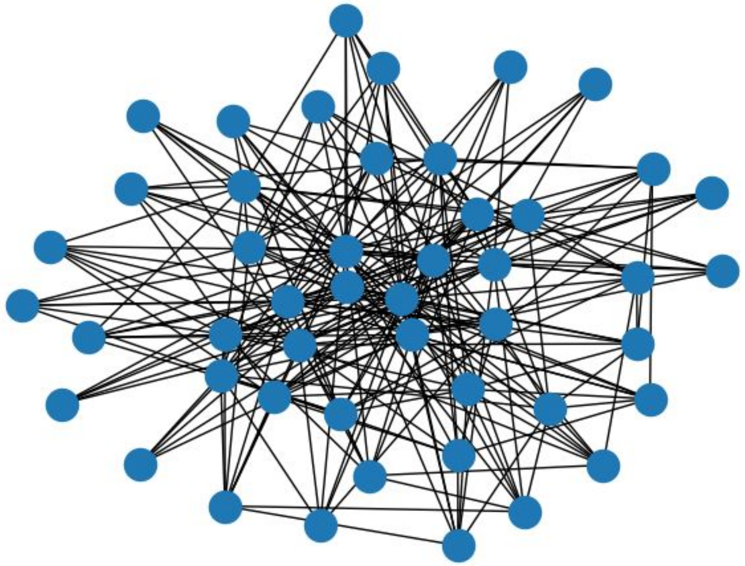
In this presentation: National Science Foundation Network (NSFNET)



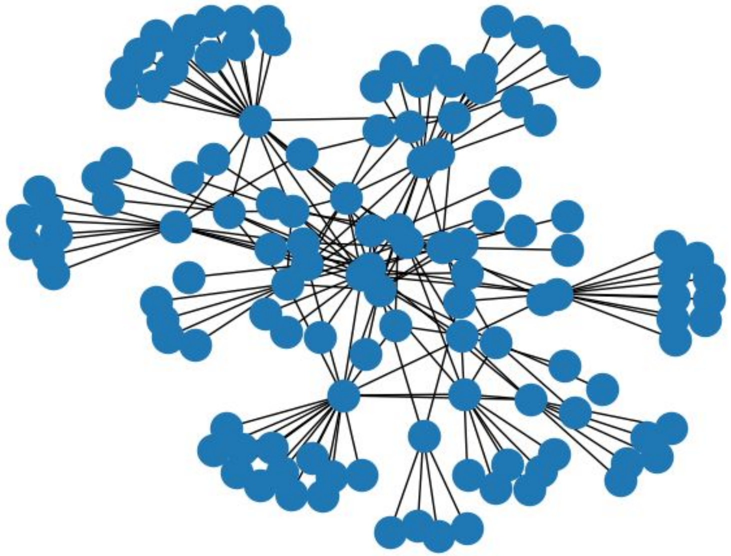


Simulation assumptions and configuration

Mycielsky(6)



Random Internet (128, 64)



Simulation assumptions and configuration

We generate demand using the following function:

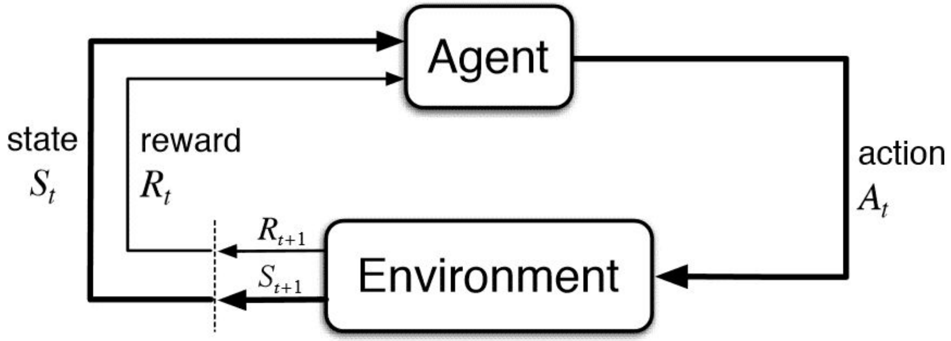
1. Amount of wavelengths (constant during simulations)
   $$\left\lceil \frac{|E|}{|V|} * \frac{\sqrt{|E|}}{2} \right\rceil$$
2. Amount of demands (for every simulation)
   Sample identically and independently distributed sequence of numbers from the uniform distribution $U(|V|^{1/2} * |\Lambda|, |V| * |\Lambda|)$

3. Demand pairs
   Permutation over the pairs of vertices, abandon if
   $|\Lambda| * \delta(v) < K \setminus \{(v_s, v_d) \mid v_s \neq v,\ v_d \neq v\}$ for arbitrary v

$${}_nP_r = \frac{n!}{(n-r)!}$$

Environment

The environment is suited for reinforcement learning
- Contains logic and an action space + state/observation space
- Holds our solution object (which is a superset of collections per wavelength for paths)

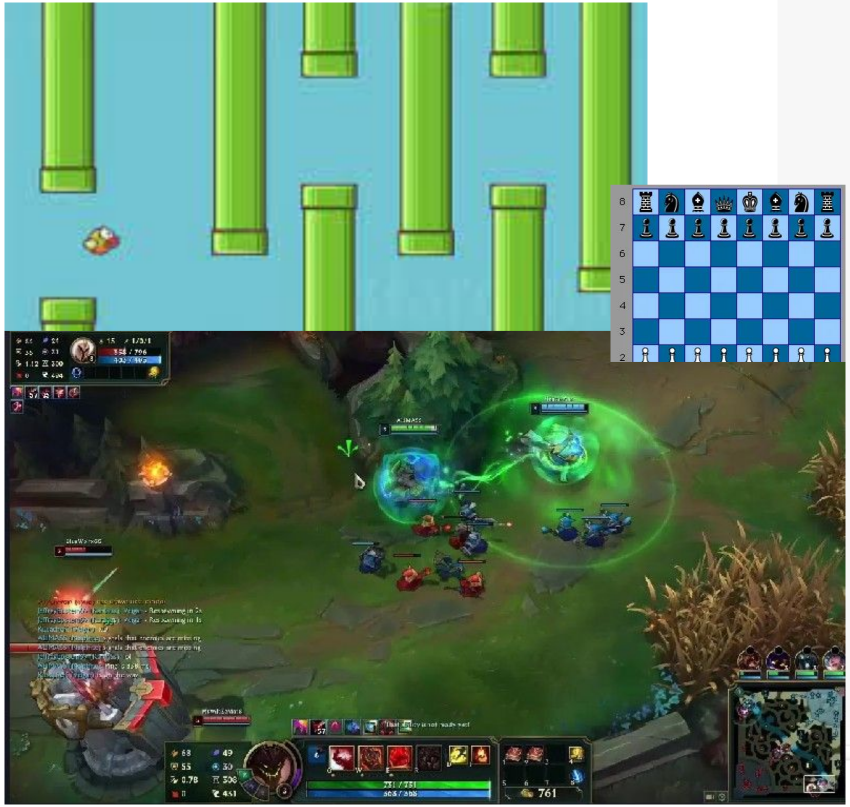


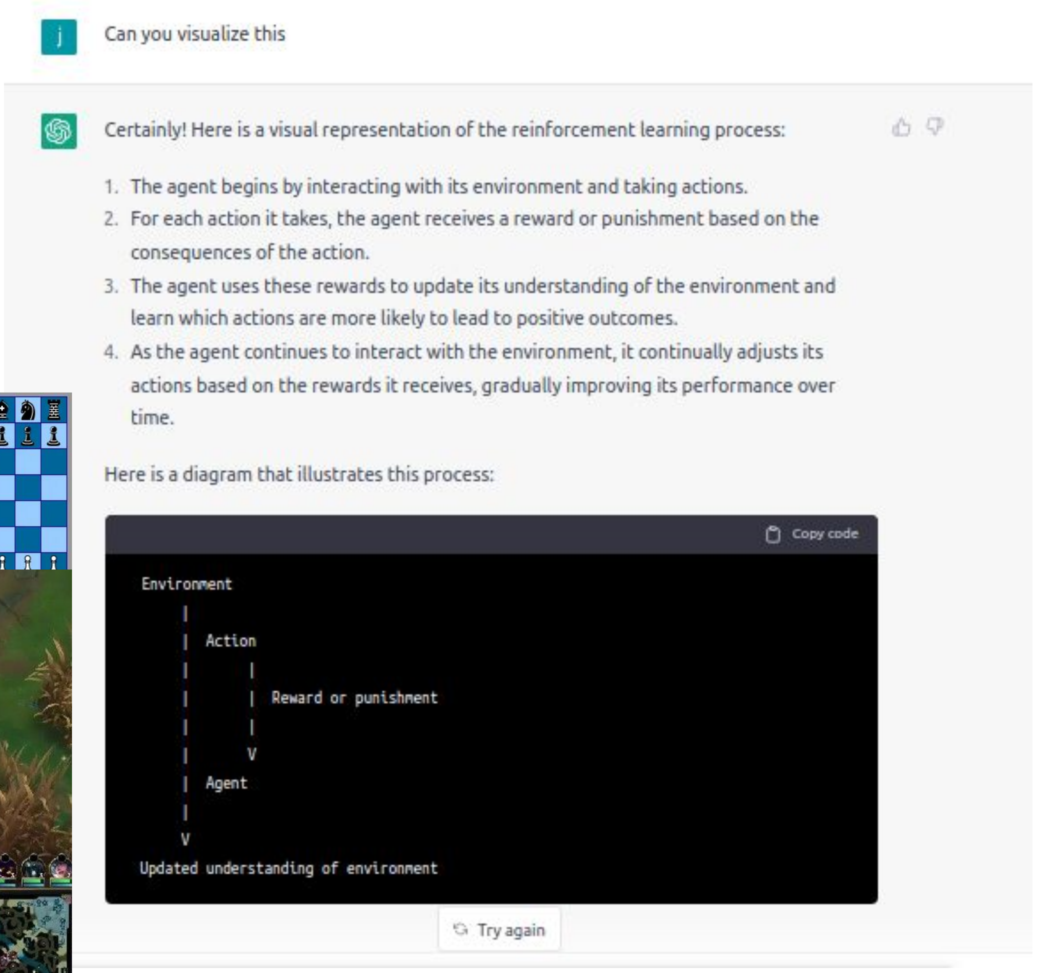
Environment has three actions
1. Remove path from solution

2. Add path to solution optimally

3. Add path to solution probabilistically (empirically distributed by normalized weights from inverse path lengths)

Environment: DRL





j Can you visualize this

Certainly! Here is a visual representation of the reinforcement learning process:

1. The agent begins by interacting with its environment and taking actions.
2. For each action it takes, the agent receives a reward or punishment based on the consequences of the action.
3. The agent uses these rewards to update its understanding of the environment and learn which actions are more likely to lead to positive outcomes.
4. As the agent continues to interact with the environment, it continually adjusts its actions based on the rewards it receives, gradually improving its performance over time.

Here is a diagram that illustrates this process:

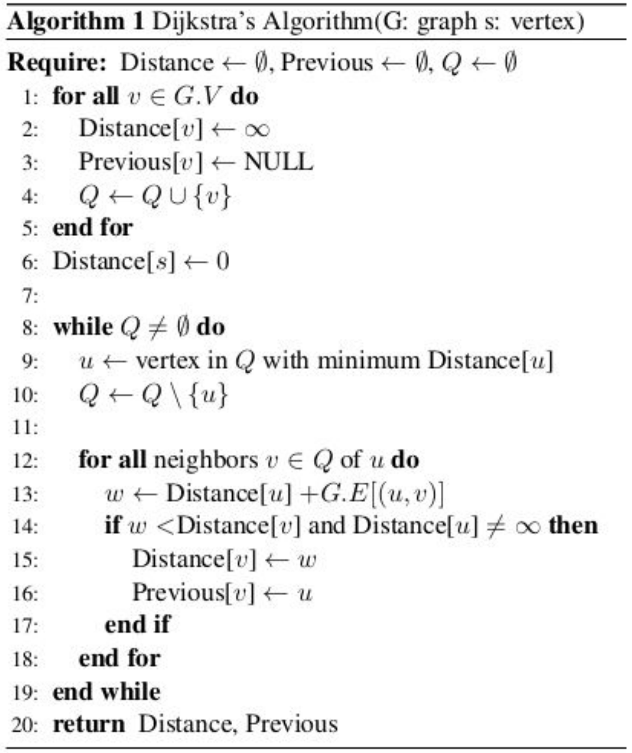
```
Environment
     |
     |  Action
     |      |
     |      |  Reward or punishment
     |      |
     |      V
     |  Agent
     |
     V
Updated understanding of environment
```

Environment

Decomposed routing and wavelength problem:

Yen’s algorithm
(extension of Dijkstra’s shortest path algorithms)
to yield the shortest paths

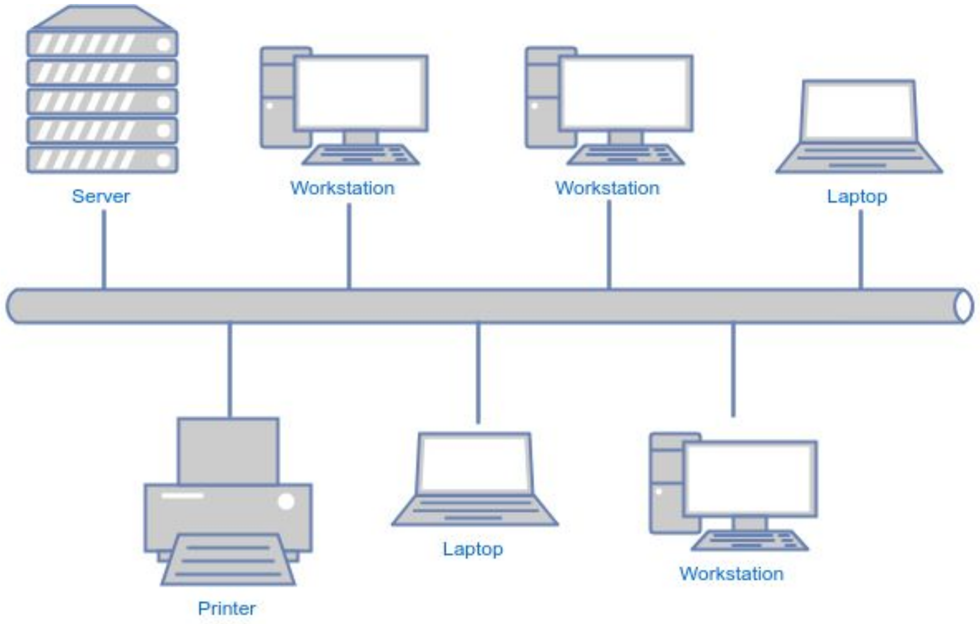
Is a polynomial time algorithm to solve k-shortest path algorithm



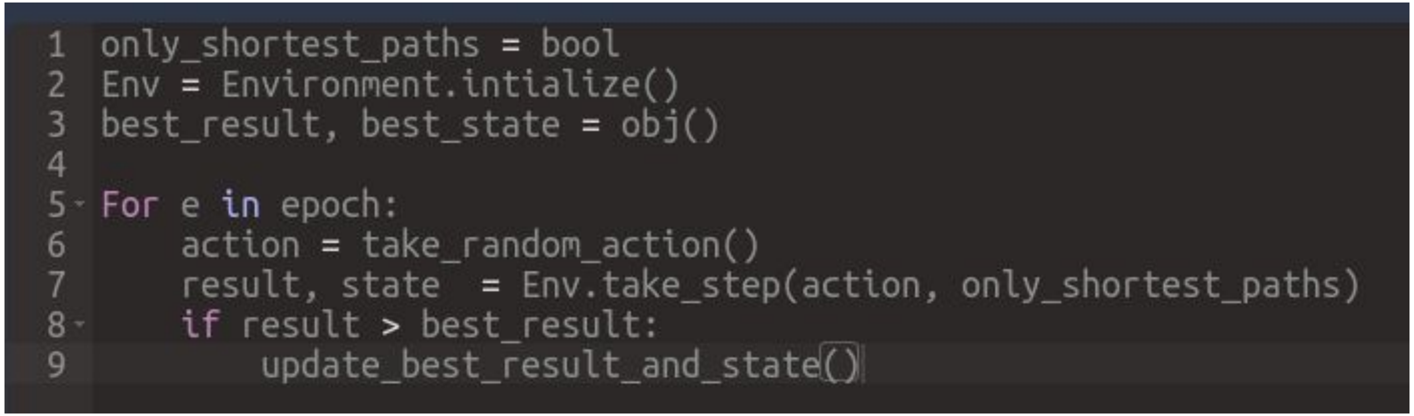
Algorithm 1 Dijkstra’s Algorithm(G: graph s: vertex)

```
Require: Distance ← $\emptyset$, Previous ← $\emptyset$, $Q \leftarrow \emptyset$
for all $v \in G.V$ do
    Distance[$v$] ← $\infty$
    Previous[$v$] ← NULL
    $Q \leftarrow Q \cup \{v\}$
end for
Distance[$s$] ← 0

while $Q \neq \emptyset$ do
    $u$ ← vertex in $Q$ with minimum Distance[$u$]
    $Q \leftarrow Q \setminus \{u\}$

    for all neighbors $v \in Q$ of $u$ do
        $w$ ← Distance[$u$] $+G.E[(u,v)]$
        if $w$ <Distance[$v$] and Distance[$u$] $\neq \infty$ then
            Distance[$v$] ← $w$
            Previous[$v$] ← $u$
        end if
    end for
end while
return Distance, Previous
```

Environment: greedy strategies



Lightpath establishment algorithms: Random, SP



```
only_shortest_paths = bool
Env = Environment.intialize()
best_result, best_state = obj()

For e in epoch:
    action = take_random_action()
    result, state  = Env.take_step(action, only_shortest_paths)
    if result > best_result:
        update_best_result_and_state()
```

Lightpath establishment algorithms: (deep) Q-learning

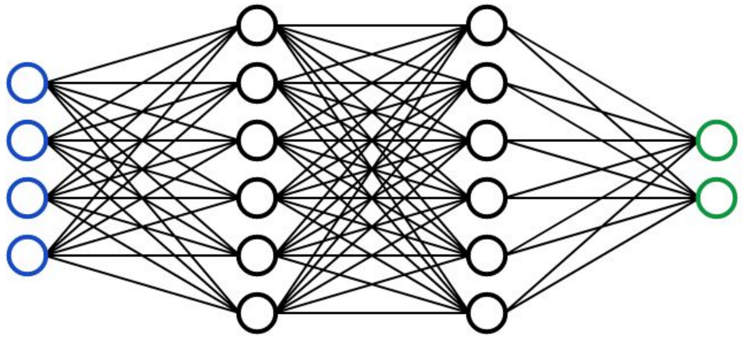
Hyperparameters epsilon=1, linear epsilon demeaning, discount factor 0.93 and learning rate 0.93 and 150 epochs

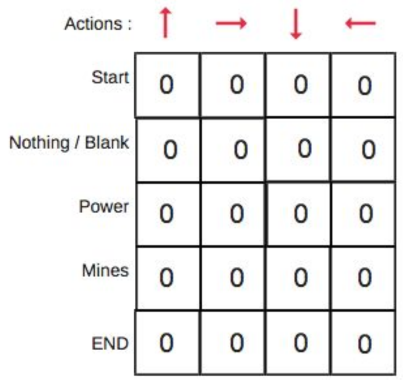
For Q learning, maintain matrix Q and update with Bellman function

$$Q_*(s,a) = \sum_{s'} P_{ss'}^{a}\left(r(s,a) + \gamma \max_{a'} Q_*(s',a')\right)$$

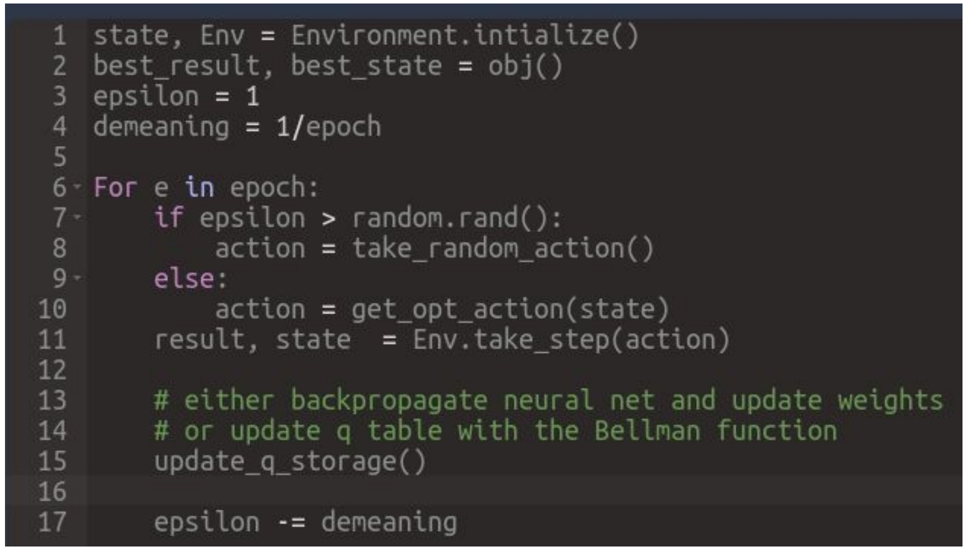
For deep-Q learning we use Stablebaselines 3/Pytorch. It maintains a neural network to estimate the Q values

Lightpath establishment algorithms: (deep) Q-learning





Actions :	↑	→	↓	←
Start	0	0	0	0
Nothing / Blank	0	0	0	0
Power	0	0	0	0
Mines	0	0	0	0
END	0	0	0	0



```
state, Env = Environment.intialize()
best_result, best_state = obj()
epsilon = 1
demeaning = 1/epoch

For e in epoch:
    if epsilon > random.rand():
        action = take_random_action()
    else:
        action = get_opt_action(state)
    result, state  = Env.take_step(action)

    # either backpropagate neural net and update weights
    # or update q table with the Bellman function
    update_q_storage()

    epsilon -= demeaning
```

Lightpath establishment algorithms: PPO, 2AC

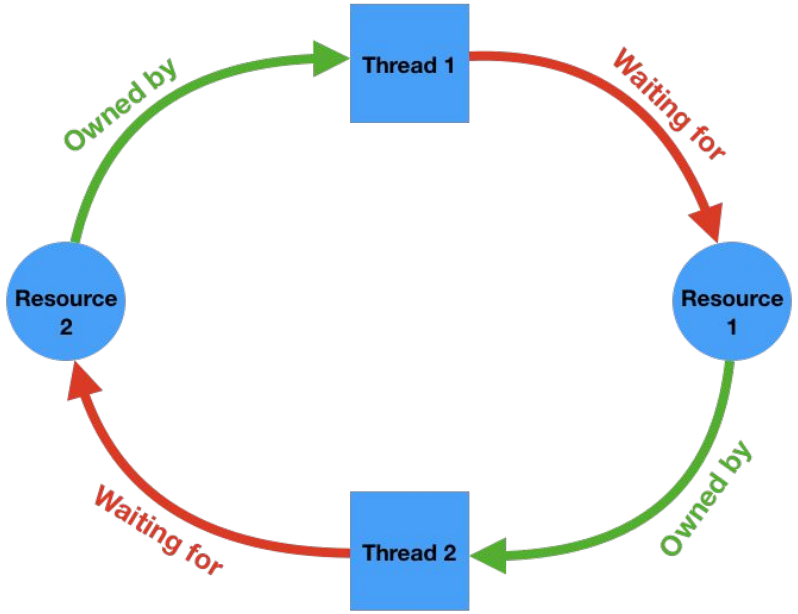
Deep reinforcement learning algorithms (Using deep neural networks)

2AC: Advantage actor critic algorithm
Actor: control policy evaluation of agent
Critic: maintains value evaluation of agent
These two workers are neural network functions that maintain an adversarial relation

PPO: Proximal Policy Optimization algorithm
A2c, but after update, take new step that closely resembles previous step
Clipping state and actions to exploit beneficial behavior

Lightpath establishment algorithms: SDG

Inspired on resolutions for death locks for operating system processes



- Add lightpaths in increasing order of amount of blocked lightpaths of other demands.
- Use sampling and random blocking to explore alternative solutions

Lightpath establishment algorithms: ILP

Integer Linear programming (using GEKKO library).

- System of linear equations.
- Guaranteed optimal solution given 25-shortest path depth if it converges.
- Is 15-50x slower than some of the aforementioned methods
- Is our estimated smallest upper bound.

Greatest upper bound can be obtained when

- Assuming all demands are provisioned using the shortest paths
- Note that this can be infeasible for the RWA problem instance

Results

- Can reinforcement learning have a contribution to the approximation ratio?
  Yes, A2c and DQL scale very well and often achieve the best ratios for larger instances. Depending on the graph type, the ratio is different. However, for smaller graphs (NSFNET) we see the best performance by SDG.

- How close does our estimated upper bound come to the upper bounds?
  Average estimated smallest upper bound approximation α: 1.16 (NSFNET)
  Greatest upper bound approximation α: 2.14 (NSFNET)

Discussion and reflection

Are the approximation ratios good?

How applicable is the simulation?

- Edge and demand assumptions are most unrealistic

How is the routing and wavelength problem relevant to optical networking?

- Economic benefit of wavelength division multiplexing

What can we expect from optical networking ?

Thank you for your attention!

In short, thesis contributions:

- Demonstrated the usage of (deep) reinforcement learning for the routing and wavelength assignment problem
- Established a scalable, simple environment for the decomposed routing and wavelength assignment problem which can used by several optimization algorithms
- Accumulated several divergent research papers and background literature and reduced it into one coherent product

Visit https://gitlab.com/jsmaxim/rl4rwa for the thesis, source code and slides