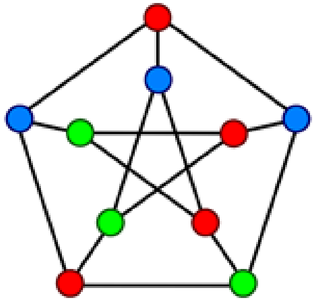
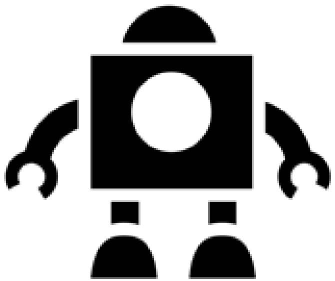
Exploratory Combinatorial Optimization with Reinforcement Learning

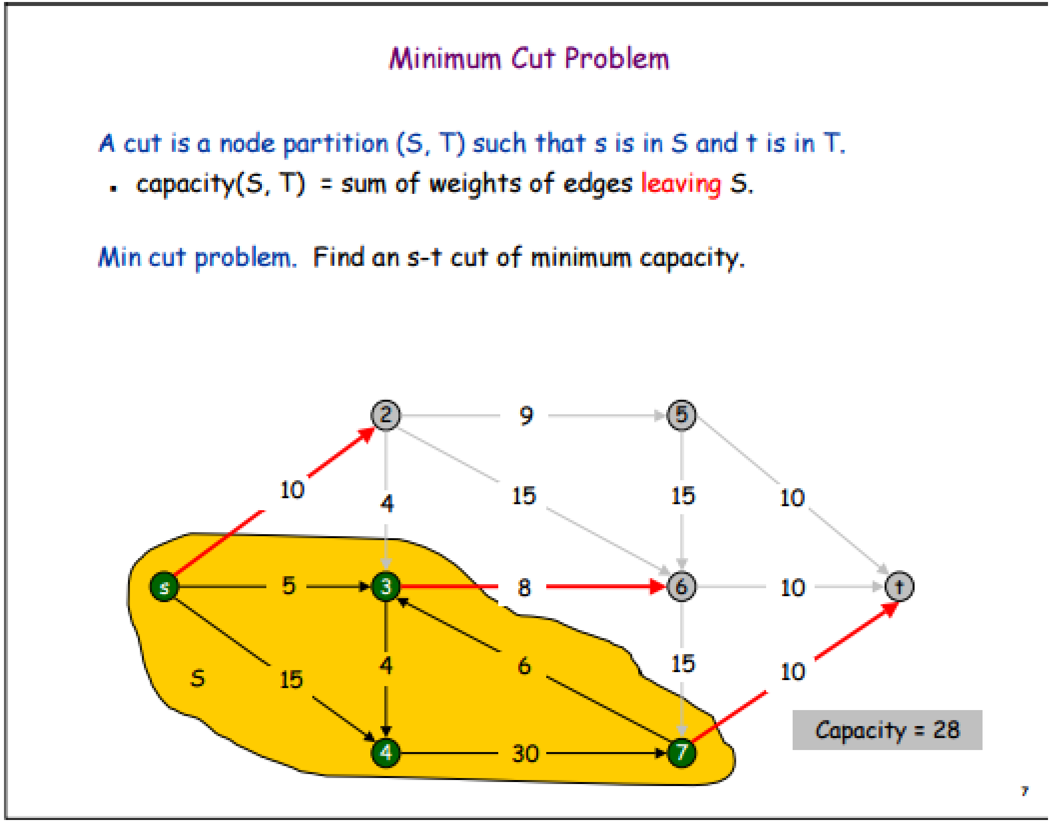
AAAI 2020

Thomas D. Barrett, William R. Clements, Jakob N. Foerster, A. I. Lvovsky





Is this problem easy or hard?



Is this problem easy or hard?

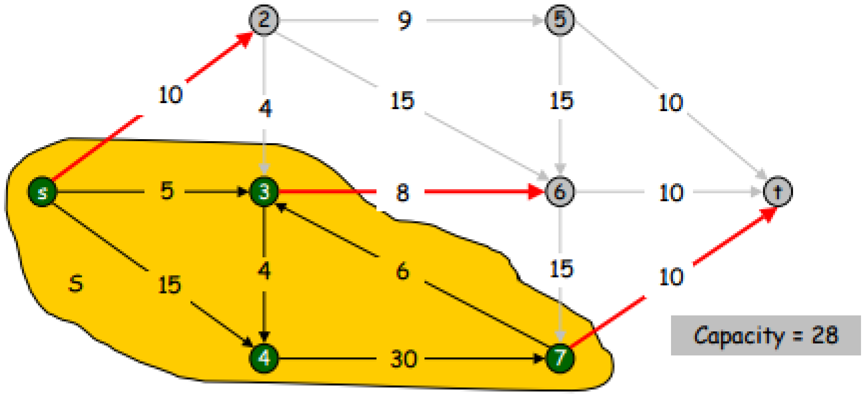
A: Easy. there is polynomial alg.

Minimum Cut Problem

A cut is a node partition (S, T) such that s is in S and t is in T.

- capacity(S, T) = sum of weights of edges leaving S.

Min cut problem. Find an s-t cut of minimum capacity.



7

What about this problem?

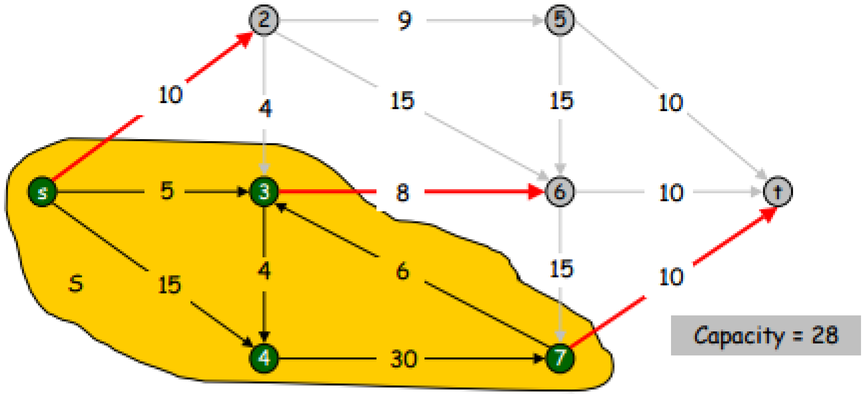
We only changed MIN to MAX…

MAX Cut Problem

A cut is a node partition (S, T) such that s is in S and t is in T.

- capacity(S, T) = sum of weights of edges leaving S.

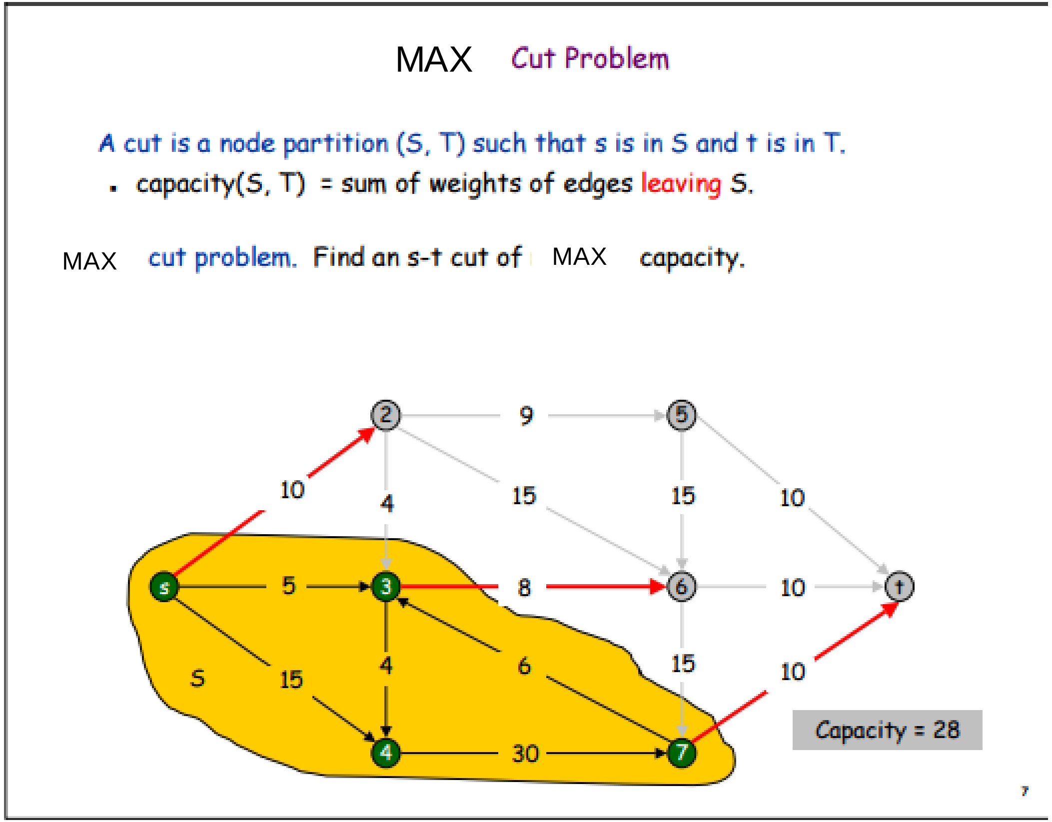
MAX cut problem. Find an s-t cut of MAX capacity.



What about this problem?

We only changed MIN to MAX…

A: Hard. (actually NP-Hard)



Important problems are often hard

Can we still solve them?

Bonus:

MAX-CUT cannot even be approximated to a ratio better than 0.873 unless P=NP

Solving hard problems

One might try to…

Solving hard problems

One might try to…

- Focus on easy special cases

Solving hard problems

One might try to…

- Focus on easy special cases
- Approximations

Solving hard problems

One might try to…

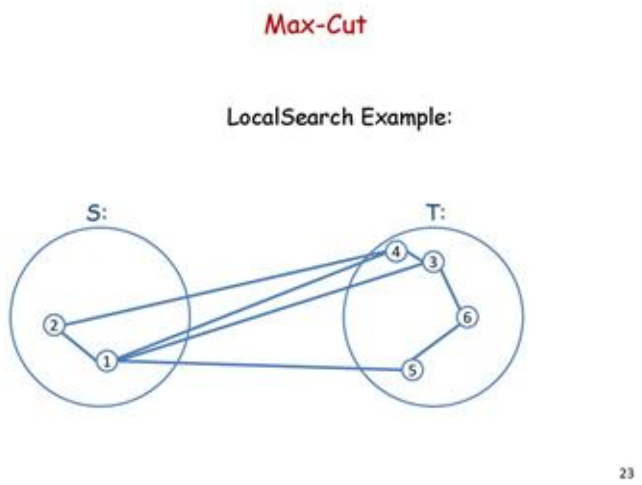
- Focus on easy special cases
- Approximations
- Greedy local search

Greedy local search

- Take best local step
- Repeat until local optimum

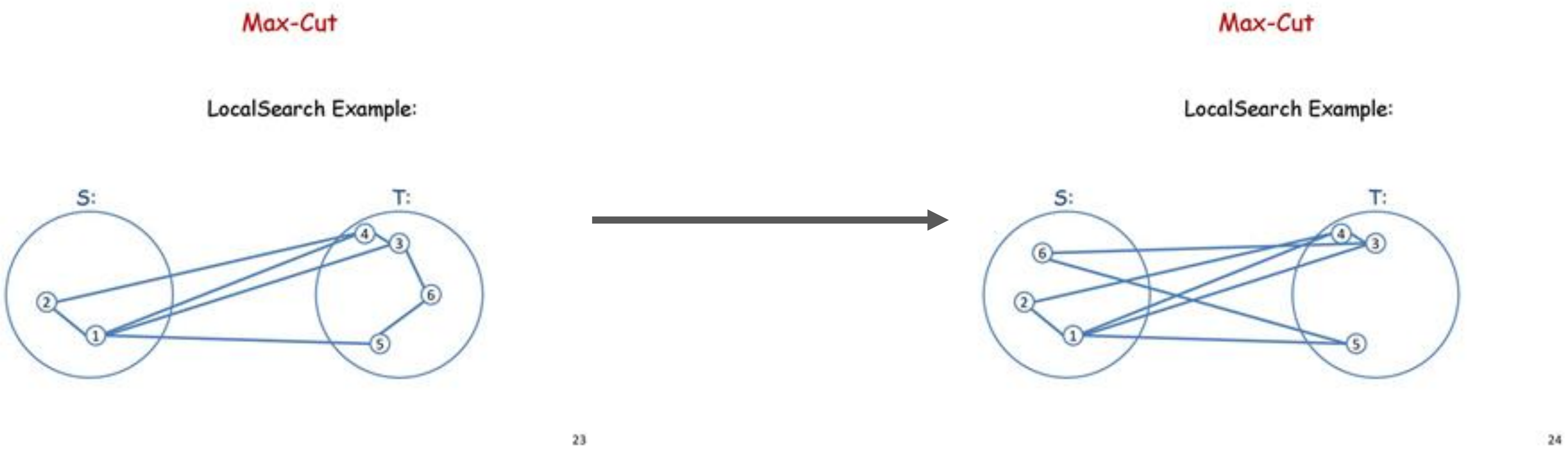
Greedy local search

- Take best local step
- Repeat until local optimum



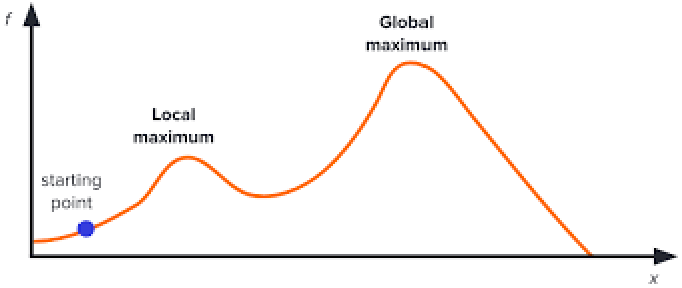
Greedy local search

- Take best local step
- Repeat until local optimum



Greedy local search

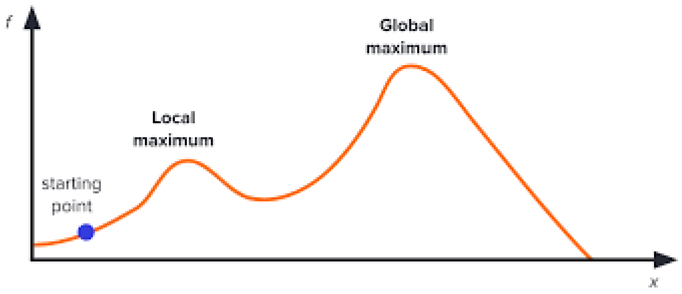
local optimum << global optimum



Greedy local search

local optimum << global optimum

- Best local step might be bad in long term!



Idea

Make greedy local search algorithms smarter

Idea

Make greedy local search algorithms smarter

- Steps are still local and greedy, but…

Idea

Make greedy local search algorithms smarter

- Steps are still local and greedy, but…
- With respect to (learned) **long term** value

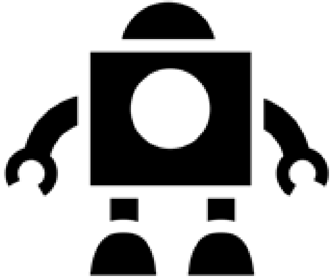
Idea

Make greedy local search algorithms smarter

- Steps are still local and greedy, but…
- With respect to (learned) **long term** value
- This is what RL algorithms do!

Reinforcement Learning 101

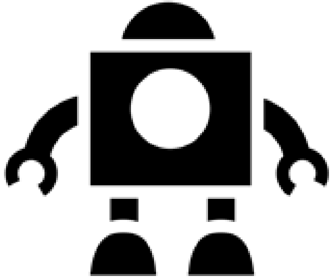
- Q-function of policy $\boldsymbol{\pi}$ $$q^{\pi}(x,a) = r(x,a) + \gamma \mathbb{E}_{x'|x,a} \mathbb{E}_{a' \sim \pi(x')} [q^{\pi}(x',a')]$$



Reinforcement Learning 101

- Q-function of policy $\boldsymbol{\pi}$ $$q^{\pi}(x,a) = r(x,a) + \gamma \mathbb{E}_{x'|x,a} \mathbb{E}_{a' \sim \pi(x')} \left[q^{\pi}(x',a') \right]$$

- Optimal Q-function $$q^{*}(x,a) = \max_{\pi} q^{\pi}(x,a) = r(x,a) + \gamma \mathbb{E}_{x'|x,a} \left[\max_{a' \in A} q^{*}(x',a') \right]$$



Reinforcement Learning 101

- Q-function of policy $\boldsymbol{\pi}$

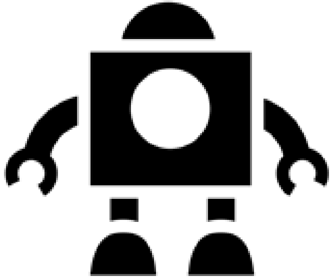
$$q^{\pi}(x,a) = r(x,a) + \gamma \mathbb{E}_{x'|x,a} \mathbb{E}_{a' \sim \pi(x')} \left[q^{\pi}(x',a') \right]$$

- Optimal Q-function

$$q^{*}(x,a) = \max_{\pi} q^{\pi}(x,a) = r(x,a) + \gamma \mathbb{E}_{x'|x,a} \left[\max_{a' \in A} q^{*}(x',a') \right]$$

- Deep Q-Learning

$$\ell_{\mathrm{DQN}}(\boldsymbol{\theta}; \mathcal{D}) \doteq \frac{1}{2} \sum_{(x,a,r,x') \in \mathcal{D}} \left(r + \gamma \max_{a' \in \mathcal{A}} Q^{*}(x', a'; \boldsymbol{\theta}^{\mathrm{old}}) - Q^{*}(x, a; \boldsymbol{\theta}) \right)^{2}.$$



Reinforcement Learning 101

- Q-function of policy $\boldsymbol{\pi}$

$$q^{\pi}(x,a) = r(x,a) + \gamma \mathbb{E}_{x'|x,a} \mathbb{E}_{a' \sim \pi(x')} [q^{\pi}(x',a')]$$

- Optimal Q-function

$$q^{*}(x,a) = \max_{\pi} q^{\pi}(x,a) = r(x,a) + \gamma \mathbb{E}_{x'|x,a} \left[\max_{a' \in A} q^{*}(x',a') \right]$$

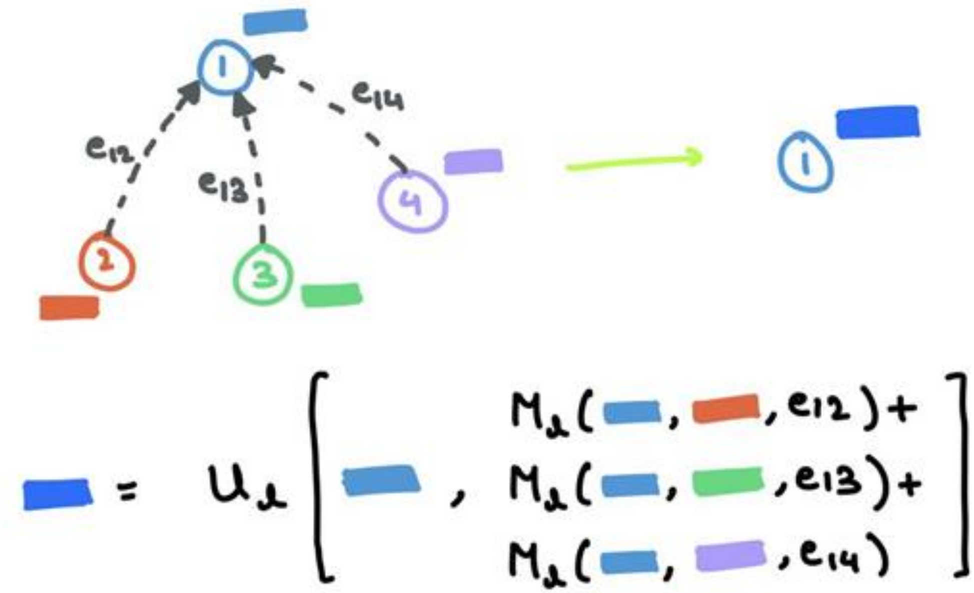
- Deep Q-Learning

$$\ell_{\mathrm{DQN}}(\boldsymbol{\theta}; \mathcal{D}) \doteq \frac{1}{2} \sum_{(x,a,r,x') \in \mathcal{D}} \left(r + \gamma \max_{a' \in \mathcal{A}} Q^{*}(x', a'; \boldsymbol{\theta}^{\mathrm{old}}) - Q^{*}(x, a; \boldsymbol{\theta}) \right)^{2}.$$

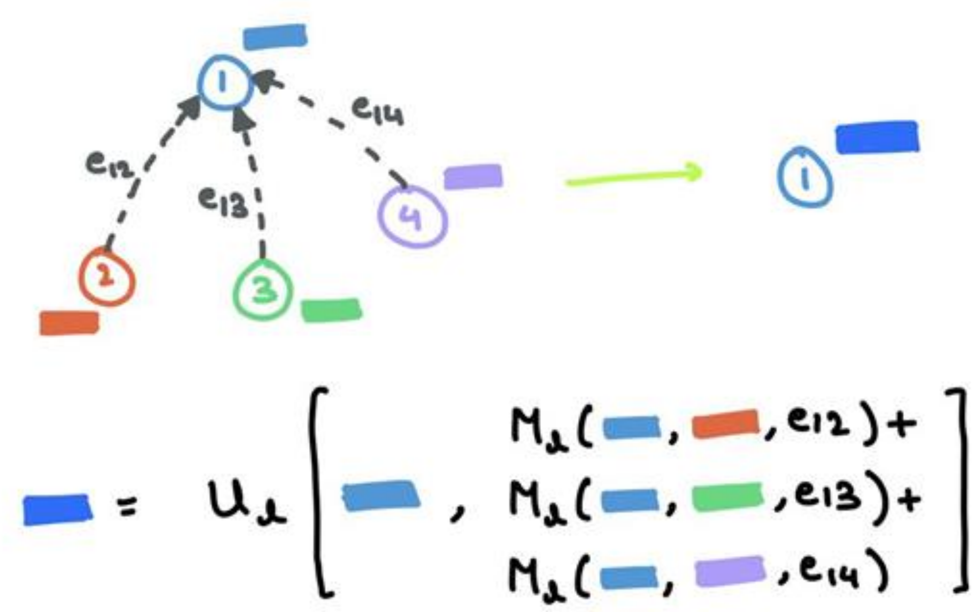
"Bootstrapping"

"Off-policy"

Message Passing Neural Network



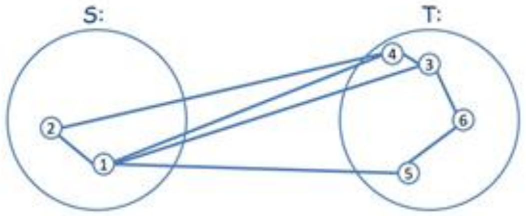
Message Passing Neural Network

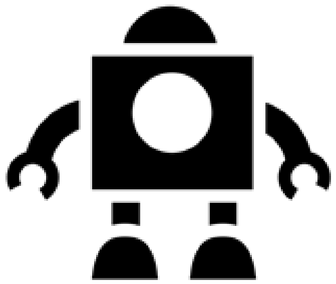


Observation: More layers -> more propagation

RL for MAX-CUT

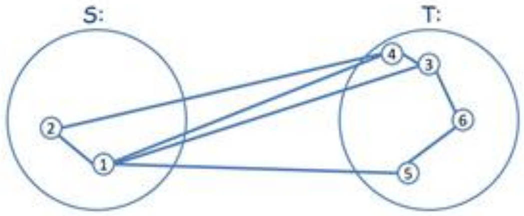
- What are the states?

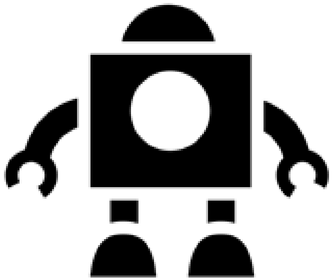




RL for MAX-CUT

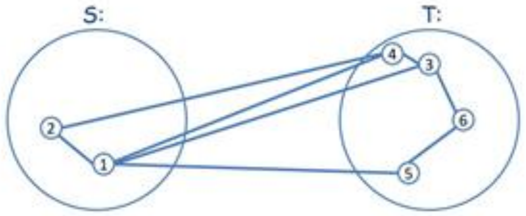
- What are the states?
- What are the actions?

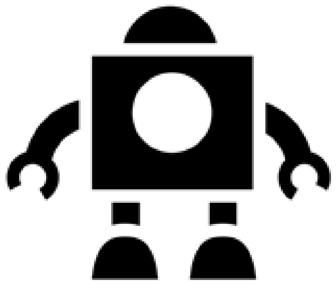




RL for MAX-CUT

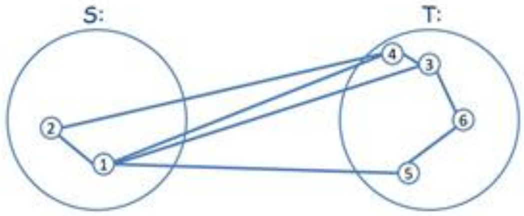
- What are the states?
- What are the actions?
- What are the immediate rewards?

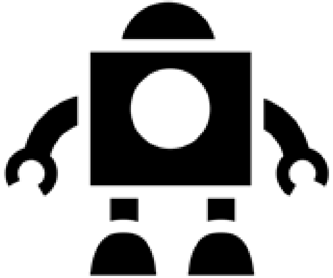




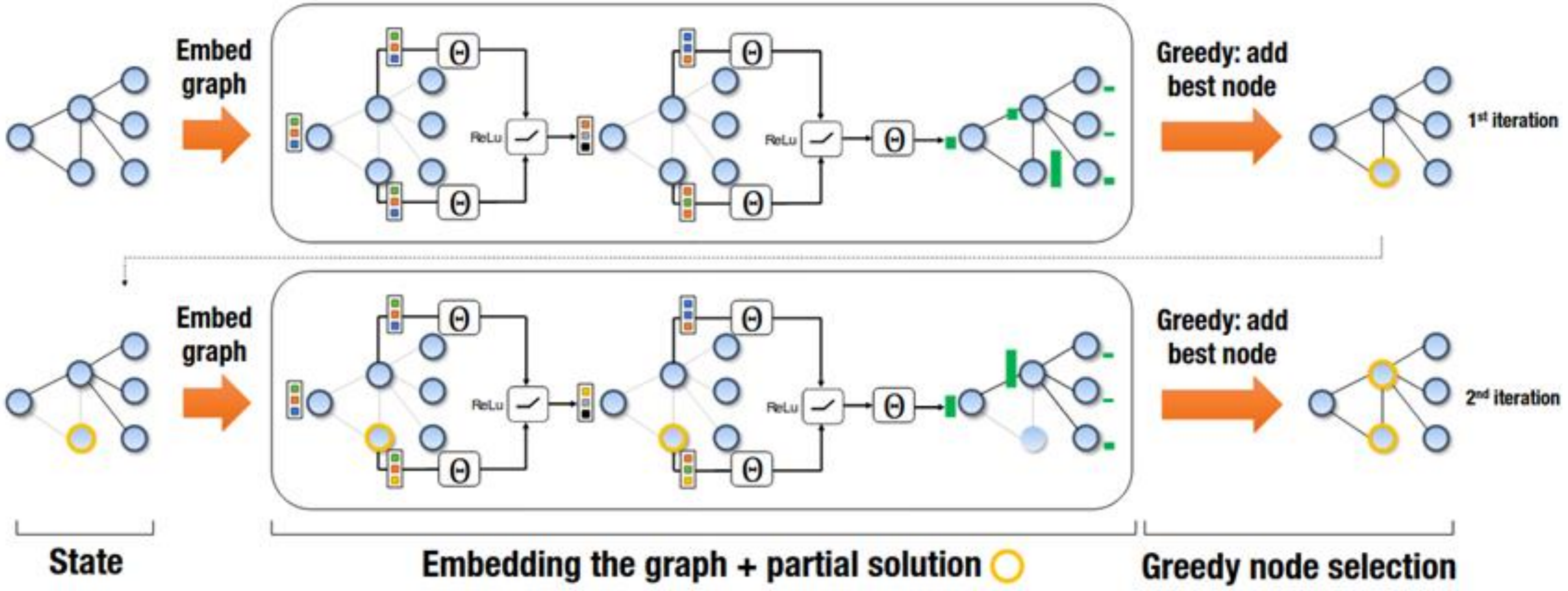
RL for MAX-CUT

- What are the states?
- What are the actions?
- What are the immediate rewards? $r(S, v) = c(h(S'), G) - c(h(S), G)$



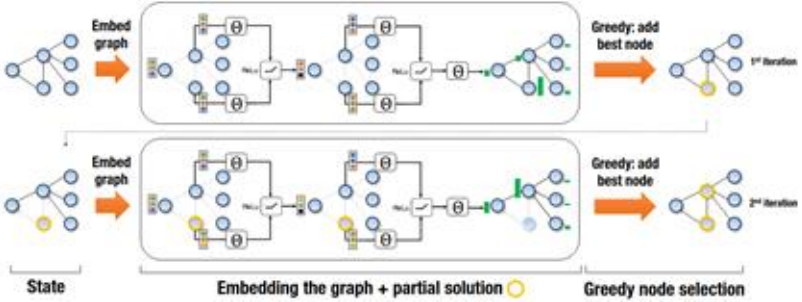


Baseline: S2V-DQN



S2V-DQN - Training

For some number of episodes…

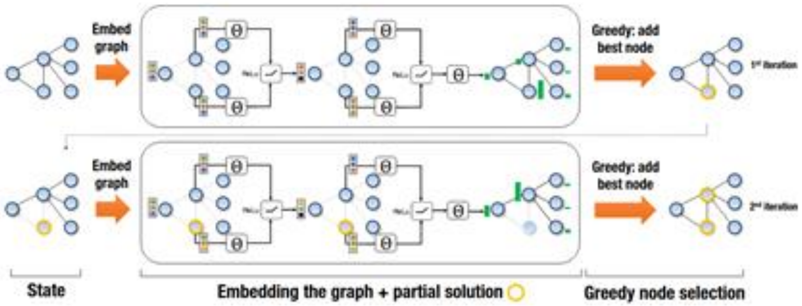


S2V-DQN - Training

For some number of episodes…

Draw some graph G from train set

Init: S <- empty, Q* = MPNN(S)



S2V-DQN - Training

For some number of episodes…

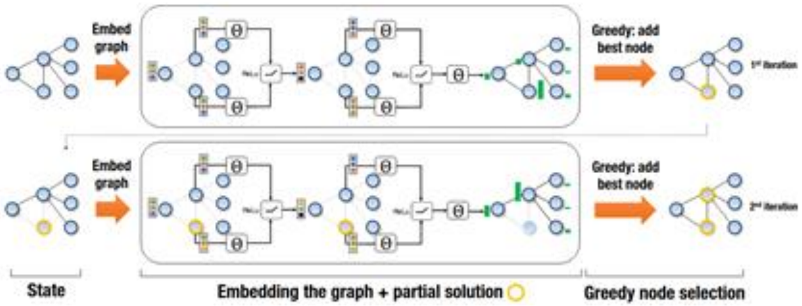
Draw some graph G from train set

Init: S <- empty, Q* = MPNN(S)

Repeat until Q*(v) < 0 for all v:

Add vertex with largest Q* to S

Q* = MPNN(S)



S2V-DQN - Training

For some number of episodes…

Draw some graph G from train set

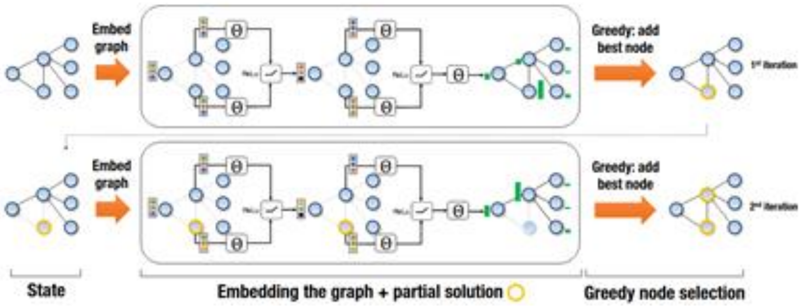
Init: S <- empty, Q* = MPNN(S)

Repeat until Q*(v) < 0 for all v:

Add vertex with largest Q* to S

Q* = MPNN(S)

if iter % K update MPNN



S2V-DQN - Training

For some number of episodes…

Draw some graph G from train set

Init: S <- empty, Q* = MPNN(S)

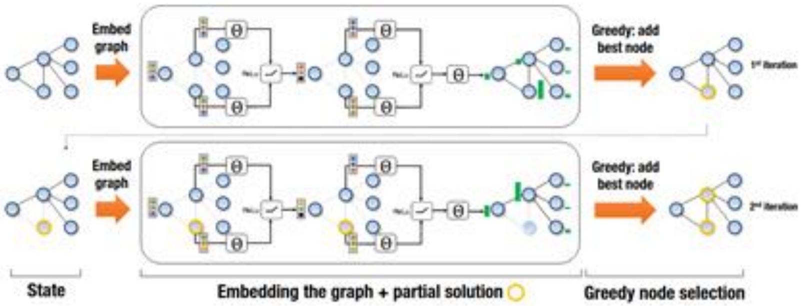
Repeat until Q*(v) < 0 for all v:

Add vertex with largest Q* to S

Q* = MPNN(S)

if iter % K update MPNN

\+ decaying epsilon greedy!



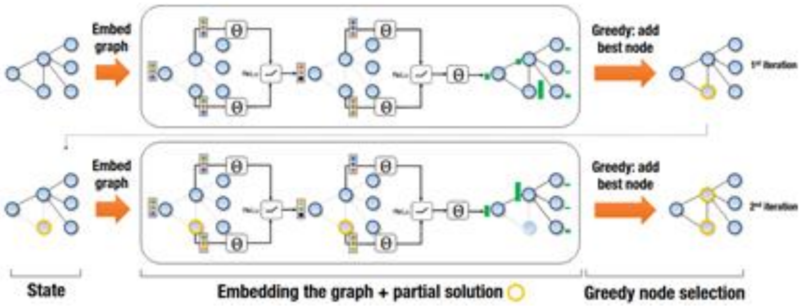
S2V-DQN - Testing on a new graph

Init: S <- empty, Q* = MPNN(S)

Repeat until Q*(v) < 0 for all v:

Add vertex with largest Q* to S

Q* = MPNN(S)



S2V-DQN - Testing on a new graph

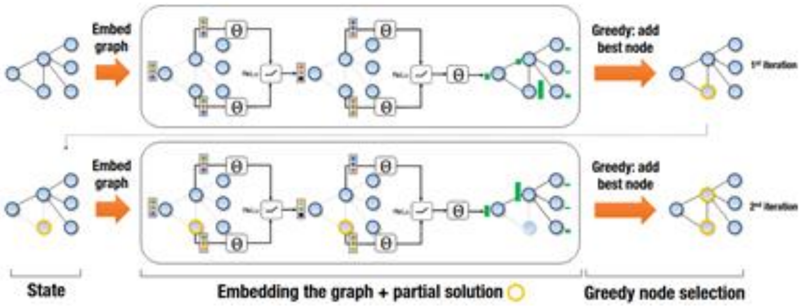
Init: S <- empty, Q* = MPNN(S)

Repeat until Q*(v) < 0 for all v:

Add vertex with largest Q* to S

Q* = MPNN(S)

No exploration during testing!



- S2V-DQN >>> greedy local searches!

Instance	OPT	S2V-DQN	MaxcutApprox	SDP
G54100	110	**108**	80	54
G54200	112	**108**	90	58
G54300	106	**104**	86	60
G54400	114	**108**	96	56
G54500	112	**112**	94	56
G54600	110	**110**	88	66
G54700	112	**108**	88	60
G54800	108	**108**	76	54
G54900	110	**108**	88	68
G5410000	112	**108**	80	54
Approx. ratio	1	**1.02**	1.28	1.90

- S2V-DQN >>> greedy local searches!
- Not only for MAX-CUT!

Table 3: Realistic data experiments, results summary. Values are average approximation ratios.

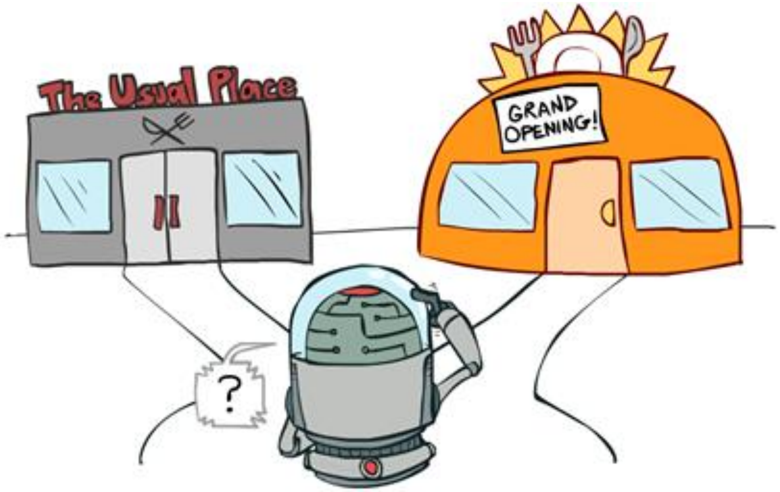
Problem	**Dataset**	**S2V-DQN**	**Best Competitor**	**2^{nd} Best Competitor**
MVC	MemeTracker	**1.0021**	1.2220 (MVCApprox-Greedy)	1.4080 (MVCApprox)
MAXCUT	Physics	**1.0223**	1.2825 (MaxcutApprox)	1.8996 (SDP)
TSP	TSPLIB	**1.0475**	1.0800 (Farthest)	1.0947 (2-opt)

But S2V-DQN is still limited:

- Does not explore during testing!
- Cannot revert decisions!

Proposed method: ECO-DQN

Quote from paper: “...***instead of learning to construct a single good solution, learn to explore for improving solutions***”



ECO-DQN Improvement #1: Flipping actions

method	S2V-DQN	ECO-DQN
action	S' = S + v	S' = S + v or S' = S - v
initialization	S <- empty	S <- random
testing	Deterministic, greedy w.r.t Q*	Tries 50 inits, picks best cut!

ECO-DQN Improvement #1: Flipping actions

method	S2V-DQN	ECO-DQN
action	S’ = S + v	S’ = S + v or S’ = S - v
initialization	S <- empty	S <- random
testing	Deterministic, greedy w.r.t Q*	Tries 50 inits, picks best cut!

- However, flipping actions do not automatically improve!

ECO-DQN Improvement #2: Explorative rewards

S2V-DQN rewards: $r(S, v) = c(h(S'), G) - c(h(S), G)$

ECO-DQN Improvement #2: Explorative rewards

S2V-DQN rewards: $r(S, v) = c(h(S'), G) - c(h(S), G)$

- Late iterations: almost always negative! Less exploration.

ECO-DQN Improvement #2: Explorative rewards

S2V-DQN rewards: $r(S, v) = c(h(S'), G) - c(h(S), G)$

- Late iterations: almost always negative! Less exploration.

ECO-DQN rewards: $\mathcal{R}(s_t) = \max(C(s_t) - C(s^*),\ 0)/|V|$

- No punishment for reducing cut value -> more exploration!

ECO-DQN Improvement #2: Explorative rewards

S2V-DQN rewards: $r(S, v) = c(h(S'), G) - c(h(S), G)$

- Late iterations: almost always negative! Less exploration.

ECO-DQN rewards: $\mathcal{R}(s_t) = \max(C(s_t) - C(s^*),\ 0)/|V|$

- No punishment for reducing cut value -> more exploration!
- Add 1/|V| to **unseen** local OPTs (small intrinsic reward)

ECO-DQN Improvement #3: Rich observations

S2V-DQN state: binary encoding of set S

- Input to MPNN is not rich, not contextual

ECO-DQN Improvement #3: Rich observations

S2V-DQN state: binary encoding of set S

- Input to MPNN is not rich, not contextual

ECO-DQN states:

- Context from episode!

1. Vertex state, i.e. if v is currently in the solution set, S.
2. Immediate cut change if vertex state is changed.
3. Steps since the vertex state was last changed.
4. Difference of current cut-value from the best observed.
5. Distance of current solution set from the best observed.
6. Number of available actions that immediately increase the cut-value.
7. Steps remaining in the episode.

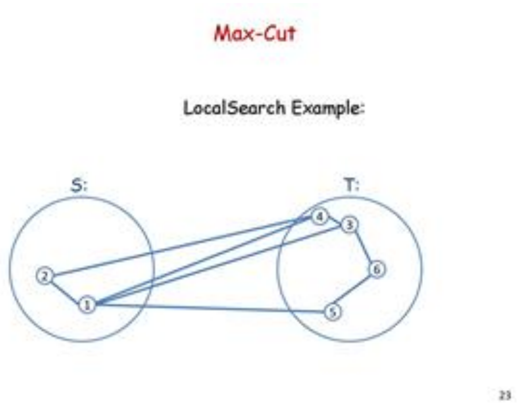
ECO-DQN - Experiments

Terminology:

ECO-DQN - Experiments

Terminology:

- MaxCutApprox (MCA) - greedy local search, no RL



ECO-DQN - Experiments

Terminology:

- MaxCutApprox (MCA) - greedy local search, no RL
- “Reversible” agent - can flip vertices (ECO-DQN, MCA-rev)

ECO-DQN - Experiments

Terminology:

- MaxCutApprox (MCA) - greedy local search, no RL
- “Reversible” agent - can flip vertices (ECO-DQN, MCA-rev)
- “Irreversible” agent - only adds vertices (S2V-DQN, MCA-irrev)

ECO-DQN - Experiments

Terminology:

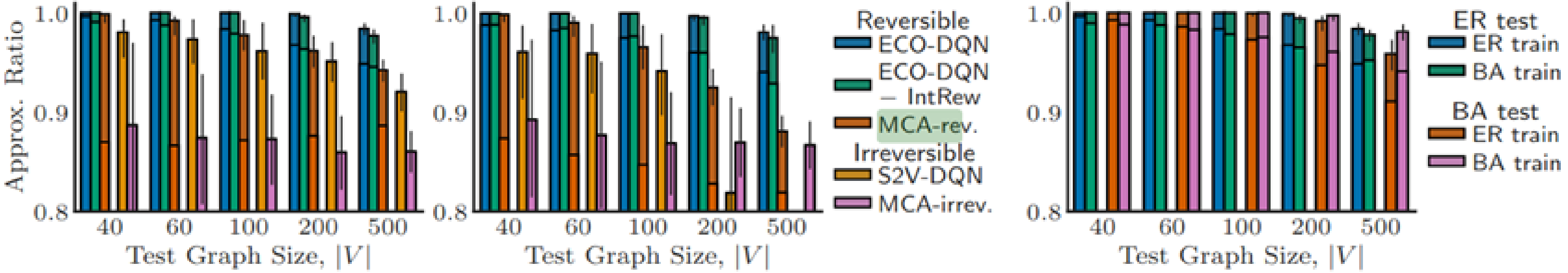
- MaxCutApprox (MCA) - greedy local search, no RL
- “Reversible” agent - can flip vertices (ECO-DQN, MCA-rev)
- “Irreversible” agent - only adds vertices (S2V-DQN, MCA-irrev)
- ER - Erdos-Renyi. BA - Barabasi-Albert. (families of graphs)

ECO-DQN - Experiments

Test → Train ↓	$\|V\|$=20	$\|V\|$=40	$\|V\|$=60	$\|V\|$=100	$\|V\|$=200	$\|V\|$=500	$\|V\|$=20	$\|V\|$=40	$\|V\|$=60	$\|V\|$=100	$\|V\|$=200	$\|V\|$=500
	ER graphs						BA graphs					
$\|V\|$=20	$0.99^{+0.01}_{-0.01}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$0.98^{+0.01}_{-0.01}$	$0.95^{+0.01}_{-0.01}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$0.99^{+0.01}_{-0.01}$	$0.98^{+0.01}_{-0.01}$
$\|V\|$=40	—	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$0.98^{+0.01}_{-0.01}$	—	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$0.98^{+0.01}_{-0.01}$
$\|V\|$=60	—	—	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$0.99^{+0.01}_{-0.01}$	—	—	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$0.99^{+0.01}_{-0.01}$
$\|V\|$=100	—	—	—	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	—	—	—	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	$0.98^{+0.01}_{-0.01}$
$\|V\|$=200	—	—	—	—	$1.00^{+0.00}_{-0.00}$	$1.00^{+0.00}_{-0.00}$	—	—	—	—	$0.99^{+0.01}_{-0.01}$	$0.98^{+0.01}_{-0.01}$

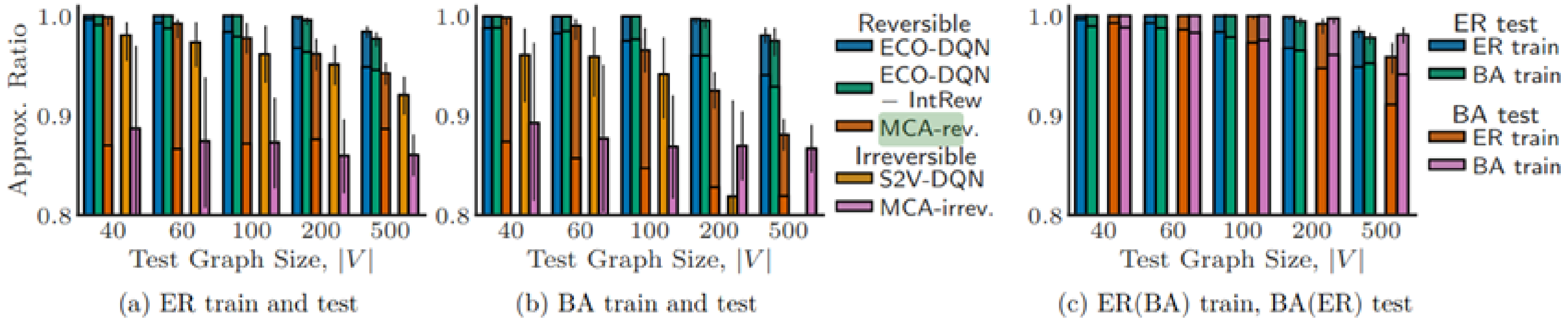
Table 2: Generalisation performance of ECO-DQN, using 50 randomly initialised episodes per graph.

ECO-DQN - Experiments



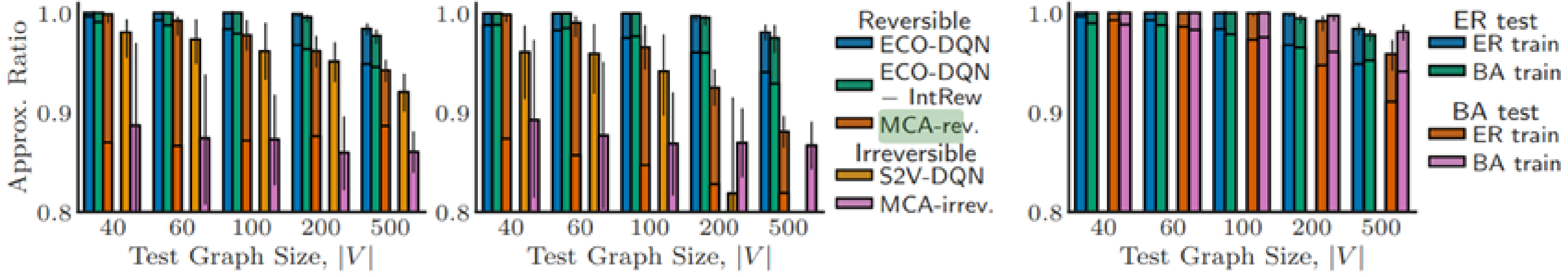
(a) ER train and test (b) BA train and test (c) ER(BA) train, BA(ER) test

ECO-DQN - Experiments



- ECO-DQN dominates on larger test graphs. (Figures a,b)

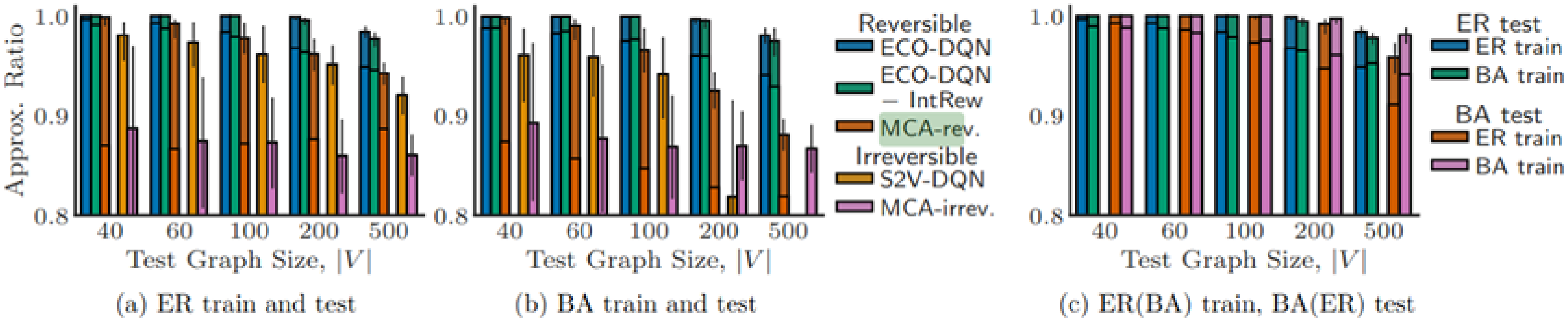
ECO-DQN - Experiments



(a) ER train and test (b) BA train and test (c) ER(BA) train, BA(ER) test

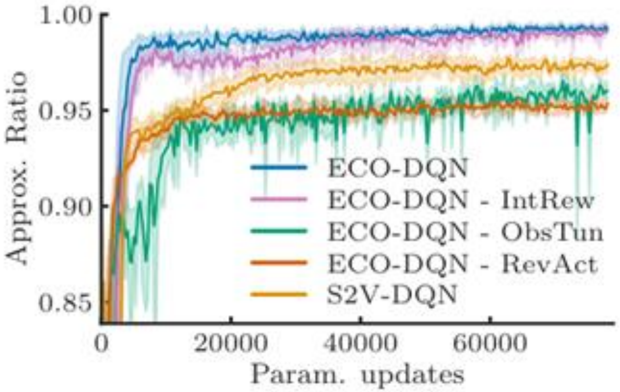
- generalizes to unseen graph types. (Figure c)

ECO-DQN - Experiments



- Random initializations help a lot! (small horizontal bars)

ECO-DQN - Experiments



(a) Learning curves: ER graphs with $|V| = 40$

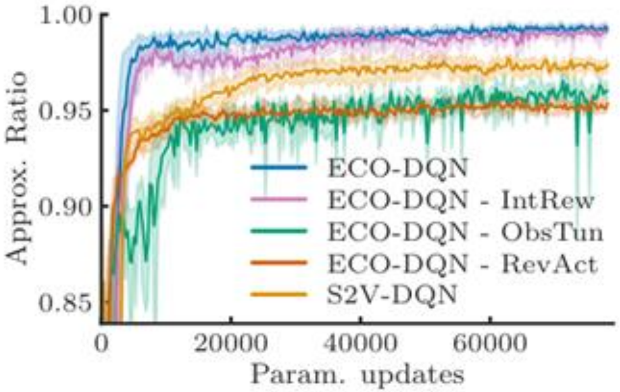
Agent	$\|V\|$=20	$\|V\|$=40	$\|V\|$=60	$\|V\|$=100	$\|V\|$=200
ECO-DQN	$0.97^{+0.03}_{-0.03}$	$1.00^{+0.00}_{-0.00}$	$0.99^{+0.01}_{-0.01}$	$0.99^{+0.01}_{-0.01}$	$0.98^{+0.01}_{-0.01}$
S2V-DQN	$0.97^{+0.03}_{-0.03}$	$0.98^{+0.01}_{-0.02}$	$0.98^{+0.01}_{-0.02}$	$0.92^{+0.02}_{-0.02}$	$0.95^{+0.02}_{-0.02}$
MCA-irrev	$0.89^{+0.06}_{-0.11}$	$0.89^{+0.04}_{-0.05}$	$0.87^{+0.05}_{-0.05}$	$0.87^{+0.03}_{-0.04}$	$0.86^{+0.03}_{-0.03}$

(b) Single episode performance: ER graphs

Agent	$\|V\|$=20	$\|V\|$=40	$\|V\|$=60	$\|V\|$=100	$\|V\|$=200
ECO-DQN	$0.99^{+0.01}_{-0.01}$	$0.99^{+0.01}_{-0.01}$	$0.98^{+0.00}_{-0.02}$	$0.97^{+0.02}_{-0.03}$	$0.93^{+0.02}_{-0.03}$
S2V-DQN	$0.97^{+0.01}_{-0.03}$	$0.96^{+0.03}_{-0.04}$	$0.94^{+0.02}_{-0.04}$	$0.95^{+0.02}_{-0.03}$	$0.94^{+0.02}_{-0.02}$
MCA-irrev	$0.92^{+0.05}_{-0.08}$	$0.89^{+0.05}_{-0.06}$	$0.88^{+0.04}_{-0.05}$	$0.87^{+0.03}_{-0.04}$	$0.87^{+0.03}_{-0.03}$

(c) Single episode performance: BA graphs

ECO-DQN - Experiments



(a) Learning curves: ER graphs with $|V| = 40$

Agent	$\|V\|$=20	$\|V\|$=40	$\|V\|$=60	$\|V\|$=100	$\|V\|$=200
ECO-DQN	$0.97^{+0.03}_{-0.03}$	$1.00^{+0.00}_{-0.00}$	$0.99^{+0.01}_{-0.01}$	$0.99^{+0.01}_{-0.01}$	$0.98^{+0.01}_{-0.01}$
S2V-DQN	$0.97^{+0.03}_{-0.03}$	$0.98^{+0.01}_{-0.02}$	$0.98^{+0.01}_{-0.02}$	$0.92^{+0.02}_{-0.02}$	$0.95^{+0.02}_{-0.02}$
MCA-irrev	$0.89^{+0.06}_{-0.11}$	$0.89^{+0.04}_{-0.05}$	$0.87^{+0.05}_{-0.05}$	$0.87^{+0.03}_{-0.04}$	$0.86^{+0.03}_{-0.03}$

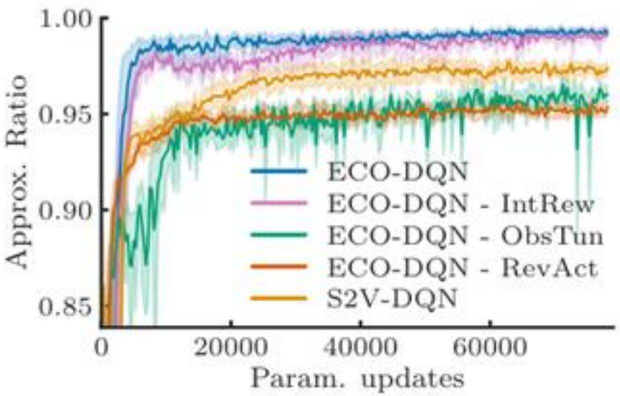
(b) Single episode performance: ER graphs

Agent	$\|V\|$=20	$\|V\|$=40	$\|V\|$=60	$\|V\|$=100	$\|V\|$=200
ECO-DQN	$0.99^{+0.01}_{-0.01}$	$0.99^{+0.01}_{-0.01}$	$0.98^{+0.00}_{-0.02}$	$0.97^{+0.02}_{-0.03}$	$0.93^{+0.02}_{-0.03}$
S2V-DQN	$0.97^{+0.01}_{-0.03}$	$0.96^{+0.03}_{-0.04}$	$0.94^{+0.02}_{-0.04}$	$0.95^{+0.02}_{-0.03}$	$0.94^{+0.02}_{-0.02}$
MCA-irrev	$0.92^{+0.05}_{-0.08}$	$0.89^{+0.05}_{-0.06}$	$0.88^{+0.04}_{-0.05}$	$0.87^{+0.03}_{-0.04}$	$0.87^{+0.03}_{-0.03}$

(c) Single episode performance: BA graphs

- Without rich observations or flipping actions, ECO-DQN < S2V-DQN!

ECO-DQN - Experiments



(a) Learning curves: ER graphs with $|V| = 40$

Agent	$\lvert V\rvert$=20	$\lvert V\rvert$=40	$\lvert V\rvert$=60	$\lvert V\rvert$=100	$\lvert V\rvert$=200
ECO-DQN	$0.97^{+0.03}_{-0.03}$	$1.00^{+0.00}_{-0.00}$	$0.99^{+0.01}_{-0.01}$	$0.99^{+0.01}_{-0.01}$	$0.98^{+0.01}_{-0.01}$
S2V-DQN	$0.97^{+0.03}_{-0.03}$	$0.98^{+0.01}_{-0.02}$	$0.98^{+0.01}_{-0.02}$	$0.92^{+0.02}_{-0.02}$	$0.95^{+0.02}_{-0.02}$
MCA-irrev	$0.89^{+0.06}_{-0.11}$	$0.89^{+0.04}_{-0.05}$	$0.87^{+0.05}_{-0.05}$	$0.87^{+0.03}_{-0.04}$	$0.86^{+0.03}_{-0.03}$

(b) Single episode performance: ER graphs

Agent	$\lvert V\rvert$=20	$\lvert V\rvert$=40	$\lvert V\rvert$=60	$\lvert V\rvert$=100	$\lvert V\rvert$=200
ECO-DQN	$0.99^{+0.01}_{-0.01}$	$0.99^{+0.01}_{-0.01}$	$0.98^{+0.00}_{-0.02}$	$0.97^{+0.02}_{-0.03}$	$0.93^{+0.02}_{-0.03}$
S2V-DQN	$0.97^{+0.01}_{-0.03}$	$0.96^{+0.03}_{-0.04}$	$0.94^{+0.02}_{-0.04}$	$0.95^{+0.02}_{-0.03}$	$0.94^{+0.02}_{-0.02}$
MCA-irrev	$0.92^{+0.05}_{-0.08}$	$0.89^{+0.05}_{-0.06}$	$0.88^{+0.04}_{-0.05}$	$0.87^{+0.03}_{-0.04}$	$0.87^{+0.03}_{-0.03}$

(c) Single episode performance: BA graphs

- Without rich observations or flipping actions, ECO-DQN < S2V-DQN!
- Intrinsic rewards speed up convergence

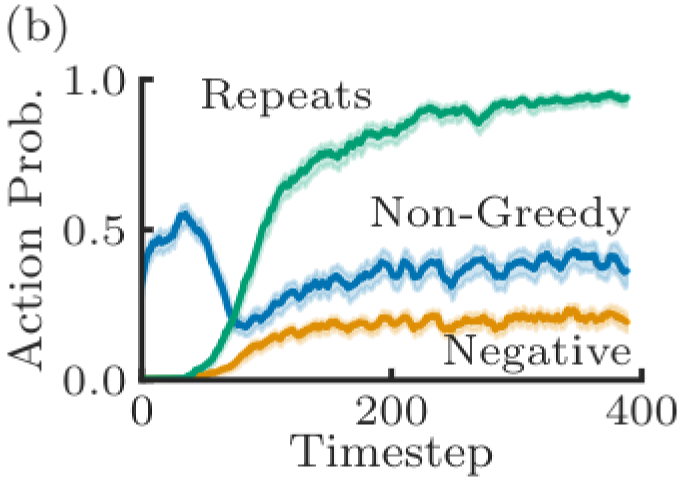
ECO-DQN - Experiments

Dataset	ECO-DQN	S2V-DQN	MCA-(rev, irrev)
Physics	1.000	0.928	0.879, 0.855
G1-10	0.996	0.950	0.947, 0.913
G22-32	0.971	0.919	0.883, 0.893

Table 1: Average performance on known benchmarks.

ECO-DQN - Experiments

- Explorative! Takes “bad” actions



Outlook

Not the first CO+RL combination, but great improvements!

Outlook

Not the first CO+RL combination, but great improvements!

+ Novelty in “learning to explore”
+ Great ablations!

Outlook

Not the first CO+RL combination, but great improvements!

+ Novelty in “learning to explore”
+ Great ablations!
- Compare DQN with DDQN? Actor-Critic methods?
- Only MAX-CUT