

UNIVERSITY OF  
**WATERLOO**



## Searching for Hypergraphs Using Reinforcement Learning

Parsa Salimi, Tamon Stephen

June 10, 2022

We investigate the potential of deep reinforcement learning methods for identifying interesting or extremal combinatorial structures. The concrete aim is to find hypergraphs that are challenging for the Fredman and Khachiyan duality checking algorithm. We present preliminary results that suggest that the approach is promising.

\end{abstract}

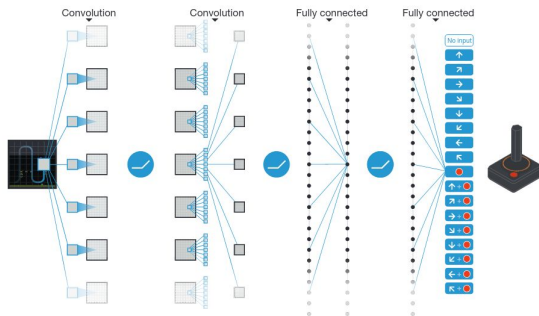
## \section{Introduction}

The study of combinatorial problems is motivated by their appearances in many practical contexts, e.g., graph coloring, traveling salesman problem, clique detection, and hypergraph coloring. Formalized as decision problems, the goal is to decide if an input has a certain property, e.g.,  $k$ -colorability or  $k$ -freeness. A problem is called hard if there is no polynomial-time algorithm to decide it, and complete if it is hard for a particular complexity class. A typical example of a hard problem is the graph coloring problem, which is NP-complete.

When designing efficient algorithms, it is important to study hard inputs. For example, a graph coloring algorithm is interesting if it is efficient on inputs that are hard for other algorithms. Such inputs are called extremal inputs. In the case of graph coloring, the most famous extremal inputs are Moore graphs

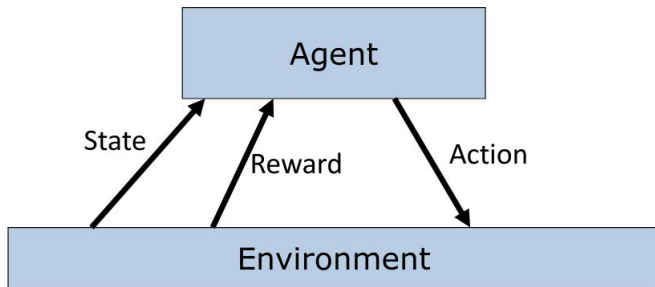
# Reinforcement learning for game playing

- (Relatively) recent breakthrough in AI: Superhuman performance in chess, go, Atari, and more!
- Driving force behind this is Deep Reinforcement learning (DRL).
- DRL = reinforcement learning (old) + deep learning (new).



Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533. DOI: [10.1038/nature14236](https://doi.org/10.1038/nature14236)

- Interested in learning about machine learning, in particular reinforcement learning.
- Wagner proposed to apply machine learning to pure mathematics by looking for counter-examples conjectures in extremal graph theory with some success.
- We are interested in some extremal hypergraph problems, and would like to see if this machinery can help.
- Common feature includes a large search space where it is unclear what kind of structure is useful.

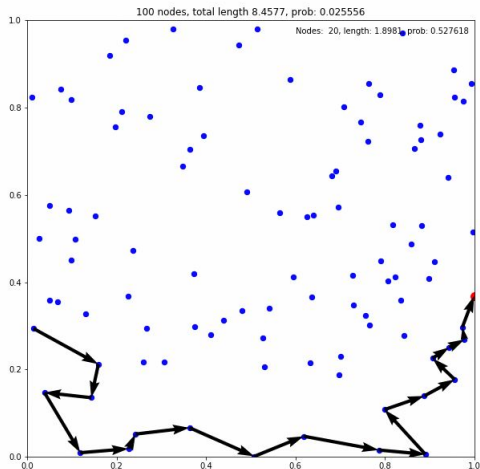


# Example: Travelling Salesperson Problem (TSP)

**States** Partial path of visited nodes in  $\mathcal{G}$ .

**Action** A node of  $\mathcal{G}$  that is not part of the state.

**Rewards** Negative length of new edge, or perhaps wait until circuit is closed.



Wouter Kool, Herke van Hoof, and Max Welling. "Attention, Learn to Solve Routing Problems!"

In: *International Conference on Learning Representations*. 2019

◀ ◻ ▶ ◀ ◻ ▶ ◀ ◻ ▶ ◀ ◻ ▶ ◀ ◻ ▶ ◀ ◻ ▶ ◀ ◻ ▶ ◀ ◻ ▶

↶ ↷ ↺ ↻

- Idea: (Wagner<sup>1</sup>): train reinforcement learning to be good at the “game of finding  $x$ ” where  $x$  is a graph with specific properties.
- We have a function on graphs  $f : G \rightarrow \mathbb{R}$  that represents a quantity of combinatorial interest.
- Problems of extremal combinatorics involve minimizing  $f$  over (classes of) graphs  $G$ .
- In many cases  $f$  is non-linear and hard to understand. It may be sensitive to structure, but we don't know what structure.
- An opportunity for reinforcement learning?

## Example (Wagner)

Find graph with highest  $\lambda_1 + \mu$ , where  $\lambda_1$  is the largest eigenvalue and  $\mu$  is the matching number.

---

<sup>1</sup>Adam Zsolt Wagner. *Constructions in combinatorics via neural networks*. 2021. arXiv: 2104.14516 [math.CO].

## Example (Aouchiche and Hansen [AH10])

For all connected graphs  $G$ ,  $\lambda_1 + \mu \geq \sqrt{n-1} + 1$ .

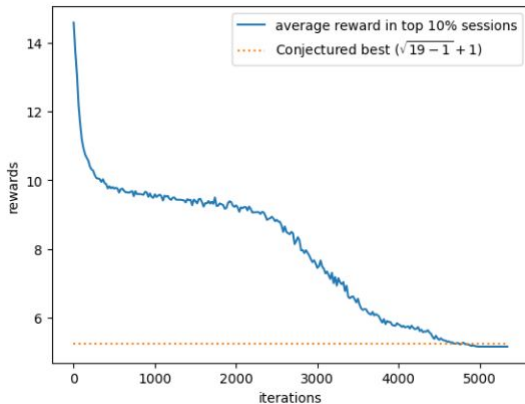


Image from Wagner [1]



# Illustration - learning structure for the conjecture

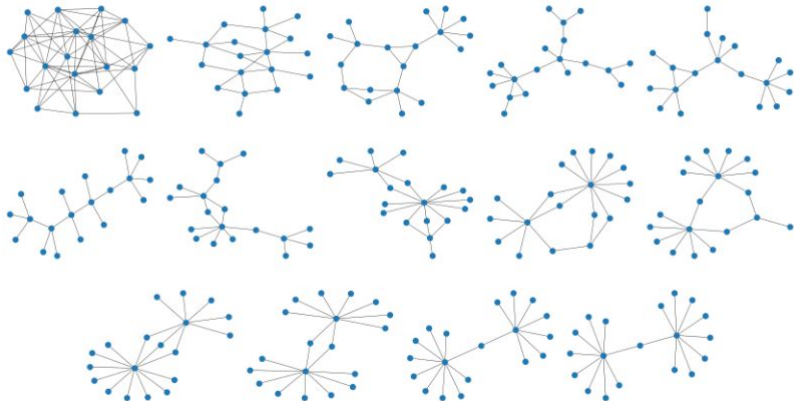
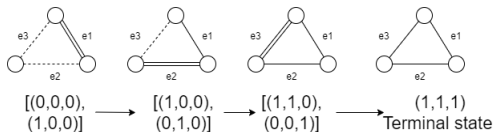


Image from Wagner [1]

# Reinforcement learning for graphs (summary)

- The approach used in the above example is to formulate the problem of finding an extremal graph as a game.
- Each stage of the game is a decision on whether to include a given edge.
- At the end, give a reward based on the final construction.
- Adjust network weights and repeat.
- There are many details involved in implementing this, including:
  - The order the edges are presented.
  - The algorithm for updating the reinforcement learning network.
  - The number of iterations to run.

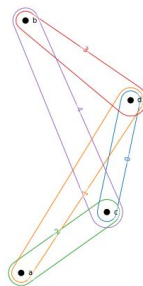
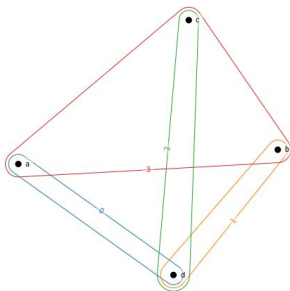
These have a large impact on the success of the algorithm.



## Definition (Hypergraph)

Let  $V$  be a finite set. A *hypergraph* on  $V$  is a tuple  $(V, \mathcal{E})$  where  $\mathcal{E}$  is a set of subsets of  $V$ . an element of  $\mathcal{E}$  is called an *edge* and an element of  $V$  is called a *vertex*.

**Example:** Let  $V = \{1, 2, 3, 4, 5\}$  and  $\mathcal{E} = \{\{1, 3, 5\}, \{2, 3\}, \{1, 4\}, \{2, 4, 5\}\}$ . Then  $(V, \mathcal{E})$  is a hypergraph.



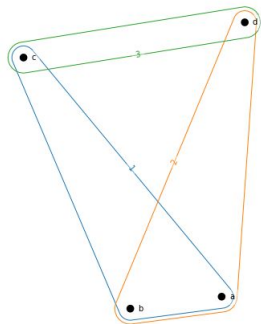
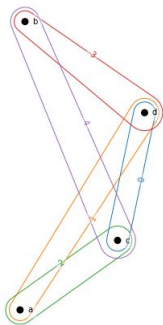
A hypergraph is *simple* if no edge contains another edge.

## Definition

Given a hypergraph  $H$ , a minimal *hitting-set* is a set  $C \subseteq V(H)$  that intersects all the edges in  $\mathcal{E}$ .





## Definition

The *Transversal Hypergraph* of  $H$  is the hypergraph obtained by keeping the vertices of  $H$  and taking all the minimal hitting sets as edges.



- The transversal hypergraph problem is a well known combinatorial problem that arises in many applications, for instance in computing minimal cut sets in metabolic networks in computational biology.
- Fredman and Khachiyan<sup>2</sup> (1996) proposed a novel recursive algorithm for computing transversal hypergraphs.
- The engine of this algorithm is a routine that checks whether a pair of hypergraphs is already a transversal pair.
- The algorithm works in worst-case time  $m^{O(\log^2 m)}$ , where  $m$  is the combined size of the two hypergraphs.
- The algorithm of Fredman and Khachiyan is poorly understood in practice.

---

<sup>2</sup>Michael L. Fredman and Leonid Khachiyan. "On the Complexity of Dualization of Monotone Disjunctive Normal Forms". In: *Journal of Algorithms* 21.3 (1996), pp. 618–628. DOI: 10.1006/jagm.1996.0062.    

# Frequencies of variables in hypergraph-transversal pairs

The recursion uses the variable that occurs with maximum frequency in both the hypergraph and its prospective dual, this frequency is critical in the complexity analysis.

## Definition

Given a hypergraph  $\mathcal{H} = (V, \mathcal{E})$ , we say that a variable  $x \in V$  *occurs with frequency*  $\varepsilon$  if

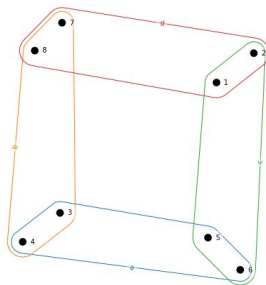
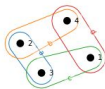
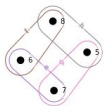
$$\varepsilon_x(\mathcal{H}) := \frac{\#\{E : x \in E, E \in \mathcal{E}\}}{|\mathcal{H}|} = \varepsilon.$$

Furthermore, we say that a vertex occurs with frequency at least  $\varepsilon$  in a pair  $(\mathcal{F}, \mathcal{G})$  if it occurs with frequency at least  $\varepsilon$  in either  $\mathcal{F}$  or  $\mathcal{G}$ . The frequency of  $x$  in  $(\mathcal{F}, \mathcal{G})$  is the maximum of  $\varepsilon_x(\mathcal{F})$  and  $\varepsilon_x(\mathcal{G})$  and is denoted by  $\varepsilon_x(\mathcal{F}, \mathcal{G})$ .

The *maximum frequency* in  $(\mathcal{F}, \mathcal{G})$  is the maximum, over all vertices  $x$ , of the frequency of  $x$  in  $(\mathcal{F}, \mathcal{G})$ , and is denoted by  $\varepsilon(\mathcal{F}, \mathcal{G})$ .

Challenging examples for the Fredman-Khachiyan algorithm have *low* maximum frequency.

- Gurvich and Khachiyan (1998)<sup>3</sup> found a recursively defined family of pairs of hypergraphs  $(F_i, G_i)$  with candidate smallest frequency.
- $F_1 = \{\{1\}, \{2\}\}$ , so  $G_1 = \{\{1, 2\}\}$ , and  $\varepsilon = 1$ .
- $F_i$  consists of two copies of the products of two disjoint copies of  $F_{i-1}$ .
- With  $i = 2$ , we have the following pair of hypergraphs which attain  $\varepsilon = 0.5$ :

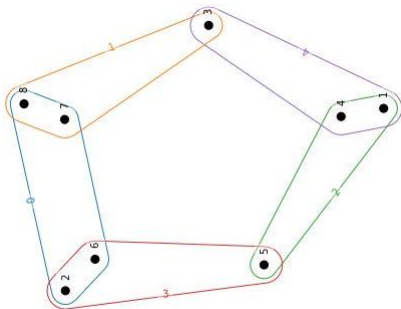


The pair  $(F_i, G_i)$  attain frequency  $1/2^i$ .

<sup>3</sup>Vladimir Gurvich and Leonid Khachiyan. "On the frequency of the most frequently occurring variable in dual monotone DNFs". In: *Discrete Mathematics* 169.1-3 (1997), pp. 245–248. DOI: 10.1016/s0012-365x(96)00090-8.

- With reinforcement learning, we were able to beat  $F_2$  on 8 variables.
- Our hypergraph has a frequency of  $2/5$ , compared to  $(F_2, G_2)$ 's frequency of  $\frac{1}{2}$ .
- And here it is:  $\mathcal{J} = \{\{2, 6, 7, 8\}, \{3, 7, 8\}, \{1, 4, 5\}, \{2, 5, 6\}, \{1, 3, 4\}\}$ .
- With transversal:

$$\text{Tr}(\mathcal{J}) = \{\{3, 4, 6\}, \{2, 3, 5\}, \{8, 1, 2\}, \{8, 3, 5\}, \{4, 5, 7\}, \{2, 4, 7\}, \{1, 5, 7\}, \\ \{1, 3, 6\}, \{8, 4, 5\}, \{1, 2, 7\}, \{3, 5, 7\}, \{1, 2, 3\}, \{1, 6, 7\}, \{8, 1, 6\}, \{8, 4, 6\}, \{3, 5, 6\}, \\ \{8, 2, 4\}, \{8, 1, 5\}, \{2, 3, 4\}, \{4, 6, 7\}\}$$





- Fix  $V$  and an ordering of the subsets of  $V$  (the possible edges).
- State space: a pair  $(a, b)$  with  $a, b \in \{0, 1\}^{2^n}$ .
- First element : Edges added so far.
- Second element : Edge currently being considered (an indicator string).
- Actions: either 0 or 1.
- Rewards:  $f(H)$  once all edges have been considered, 0 otherwise.
- Transition: Add the decision to  $a$ . advance one edge in  $b$ .

## Ordering of the edges

We order the edges by size. The ordering within the edges of the same size is unspecified.

## Invalid structures (non-simple hypergraphs)

We remove non-minimal edges before comparing  $f$ .

## Reinforcement learning algorithm

We use the cross-entropy method with neural networks.

## Search space

Rises exponentially as we increase  $n$ . Inherent problem.

- Neural network: State  $\rightarrow$  Probability distribution over actions.
- Let the neural network play: Give initial state, sample action, go to corresponding state, until the episode is finished.
- Repeat this  $N$  times.
- Take the top  $p$  percent of the  $N$  episodes, and treat them as ideal examples.
- Train the Neural Network on these toy examples.
- Now the Neural Network is slightly better at finding good examples.
- Repeat until we get interesting results.

## Summary:

- Deep reinforcement learning can be used to perform smart search in huge search spaces.
- This has been applied in context of graphs.
- We've shown that this approach can work on hypergraphs as well.
- We found a pair of hypergraphs on 8 vertices with lower maximum frequency than  $(F_2, G_2)$ .
- This took some energy and was not necessarily expected. It is not so clear how else we might have found this.
- It is also not so clear that this is optimal.

Code available on GitHub<sup>4</sup> (paper soon!)

---

<sup>4</sup>Parsa Salimi. *DualNeuralNet*. <https://github.com/parsa-salimi/DualNeuralNet> © 2021. 