Coloring Big Graphs With AlphaGoZero

Huang, J., Patwary, M., & Diamos, G. (2019)[1].

Presenter: Newsha Seyedi

David R. Cheriton School of Computer Science

University of Waterloo

Introduction

Find chromatic number of a graph using RL.

Providing a scalable approach.

Using adapted AlphaGo Zero with graph embedding.

How to tackle the problem?

Framework for learning fast heuristics inspired by AlphaGo Zero:

- Design FastColorNet (Deep Neural Network)
- Using MCTS with UCB
- Using Graph Embeddings for FCN
- Run on HPC

Background

Graph Coloring

$\chi(G)$: Minimum Vertex Coloring

- Assign a color to each vertex.
- No two adjacent vertices use the same color.
- Minimize the number of colors.

Applications:

- Parallel computing.
- VLSI.
- Pattern matching.

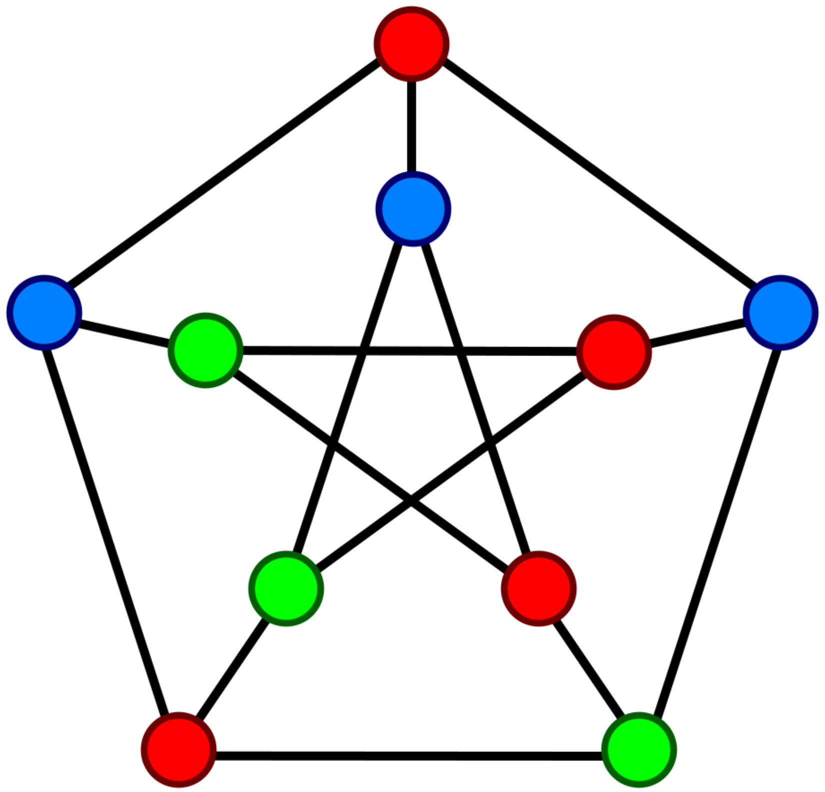


Figure 1: Vertex coloring for the Petersen graph

Algorithms

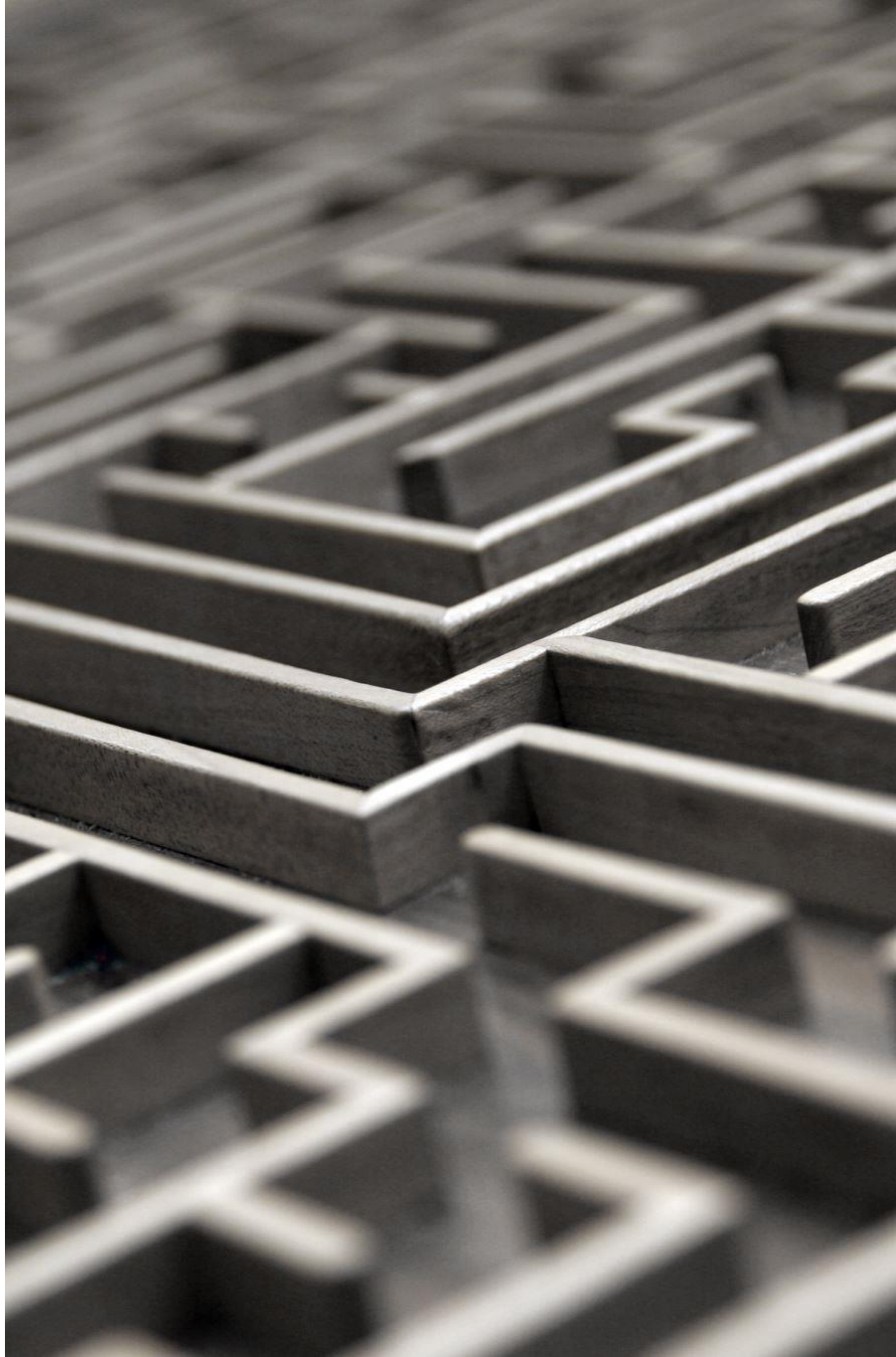
Complexity: NP-Hard

To make it fast: using heuristics

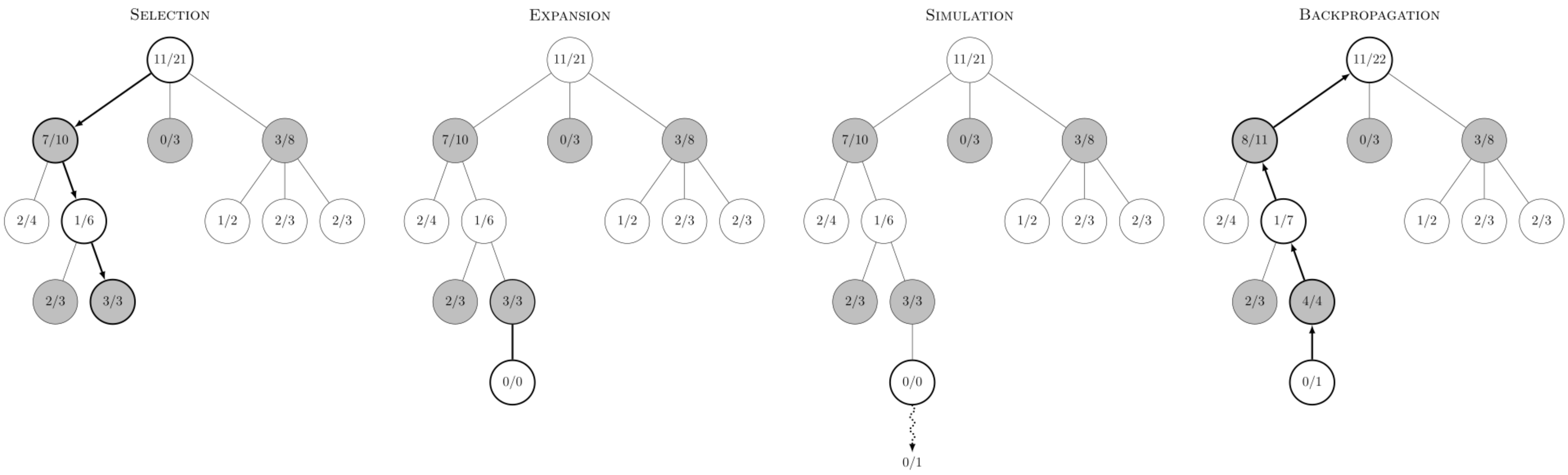
In practice: greedy heuristics

Challenges

- Knowledge about graph to design heuristics
- The complexity of getting training data



Monte Carlo Tree Search



Approach

Graph Coloring as an MDP

C : a matrix which represents the assignment of colors to graph.

The MDP state at step t: $C^{(t)}$.

The set of actions: A_i is the set of valid colors for vertex i at step t.

Reward function: the negative total number of colors used so far.

★ Different graphs imply different MDP

Graph Coloring as a Zero-Sum game

- Self-play: play against the best previous coloring algorithm.
- New reward function: win (+1) – lose (-1) – tie (0)
  - Better for reward scaling and alpha-beta-pruning.

Is Graph Coloring Harder Than Go?

	Graph-32	**Chess**	Graph-128	**Go**	Graph-512	Graph-8192	Graph-10^7
Avg. MDP States	10^{21}	$\mathbf{10^{60}}$	10^{141}	$\mathbf{10^{460}}$	10^{790}	$10^{19,686}$	$10^{45,830,967}$
Avg. Moves Per Game	32	**40**	128	**200**	512	8,192	10^7

Table 1: Estimated MDP states for **single** random graphs of various sizes compared to Chess and Go.

General View

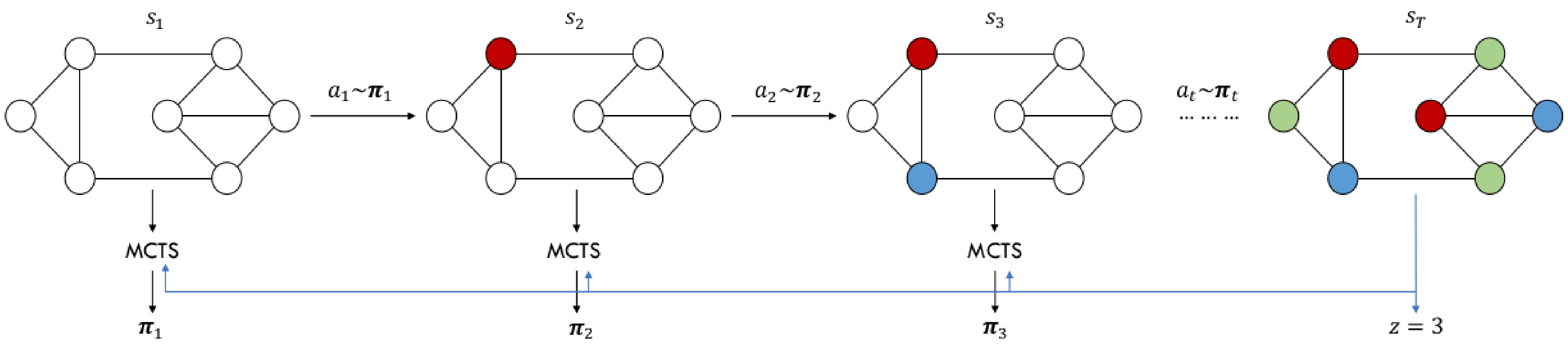


Figure 2: The reinforcement learning algorithm.

MCTS + UCB

- Selection: maximize the UCB
- Expansion
- Simulation: using FastColorNet
- Backpropagation

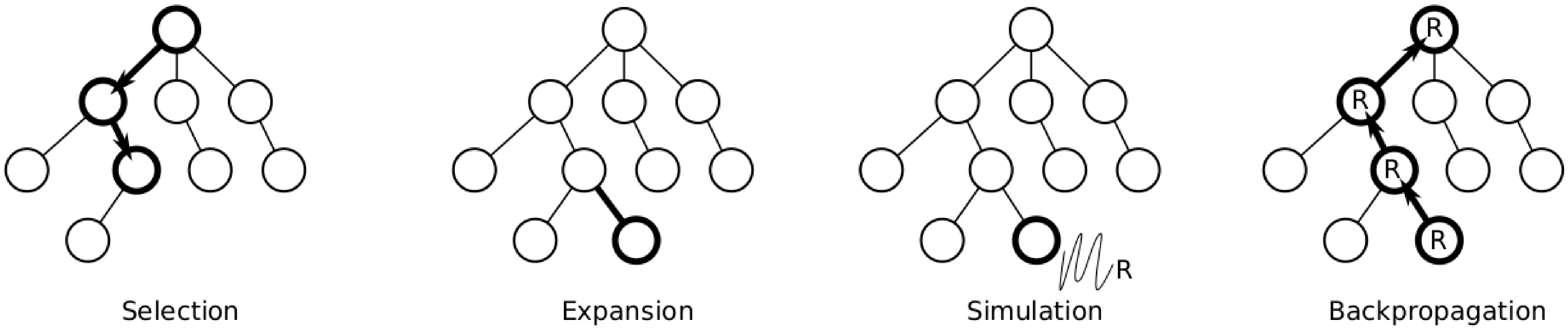


Figure 3: MCTS + UCB

Fast Color Net

$$(\boldsymbol{p}, v) \;=\; f_\theta(s)$$

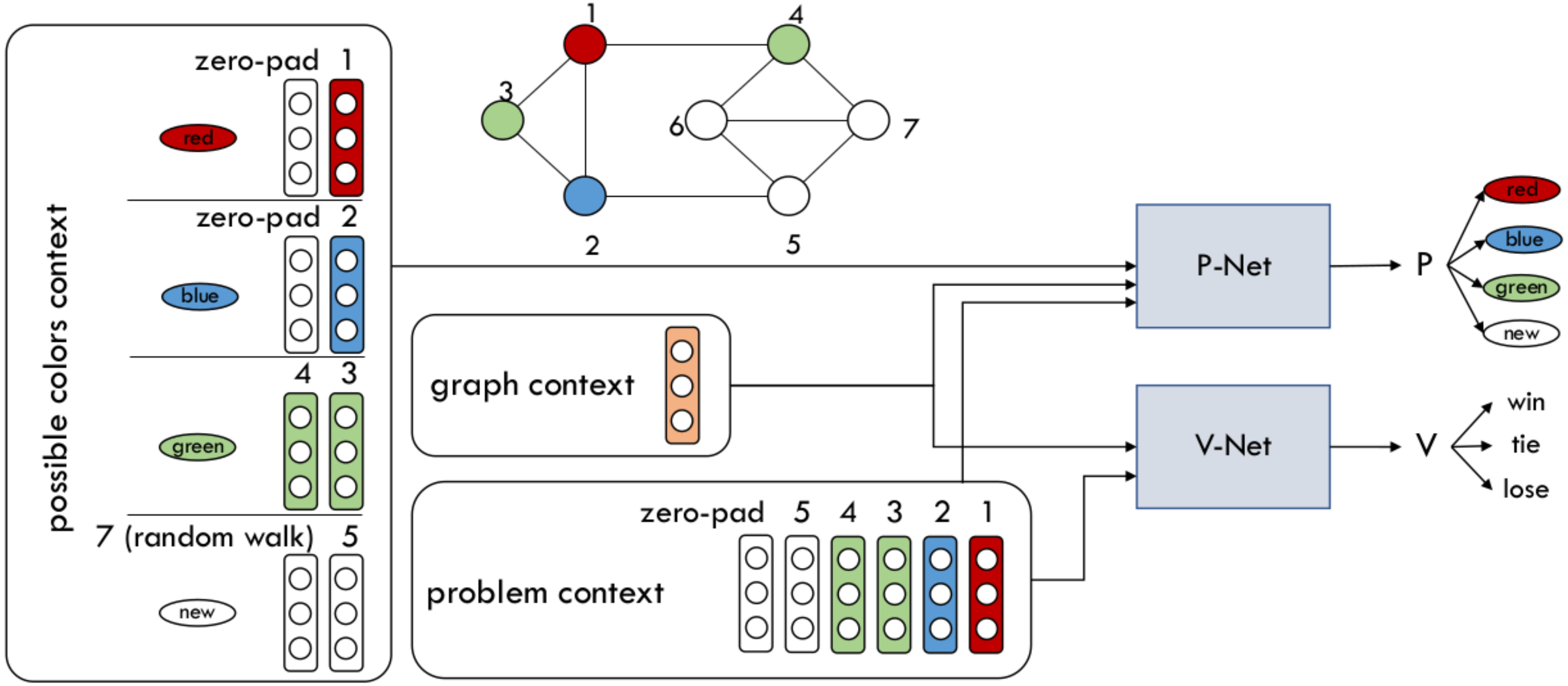


Figure 4: FastColorNet architecture computes $(\boldsymbol{p}, v)$ for graph G and colors C.

Graph Embeddings

Algorithm 1 Graph Embedding

Input: parameters $\theta \in \widetilde{\mathcal{T}}$
Initialize $\widetilde{\mu}_i^{(0)} = \mathbf{0}$, for all $i \in \mathcal{V}$
for $t = 1$ **to** T **do**
  for $i \in \mathcal{V}$ **do**
    $\nu_i = [d_i, \widetilde{\mu}_i]$, d_i is i's degree (one-hot)
    $l_i = [\nu_i, d_j, \widetilde{\mu}_j^{(t-1)}]$, where j = random($\mathcal{N}(i)$)
    $c_i = 0$
    for $k = 1$ **to** L **do**
      $l_i, c_i = LSTM_\theta(c_i, l_i)$
    end for
    $\widetilde{\mu}_i^{(t)} = LSTM_\theta(c_i, v_i)$
  end for
end for{fixed point equation update}
return $\{\widetilde{\mu}_i^T\}_{i \in \mathcal{V}}$

Fast Color Net Layers

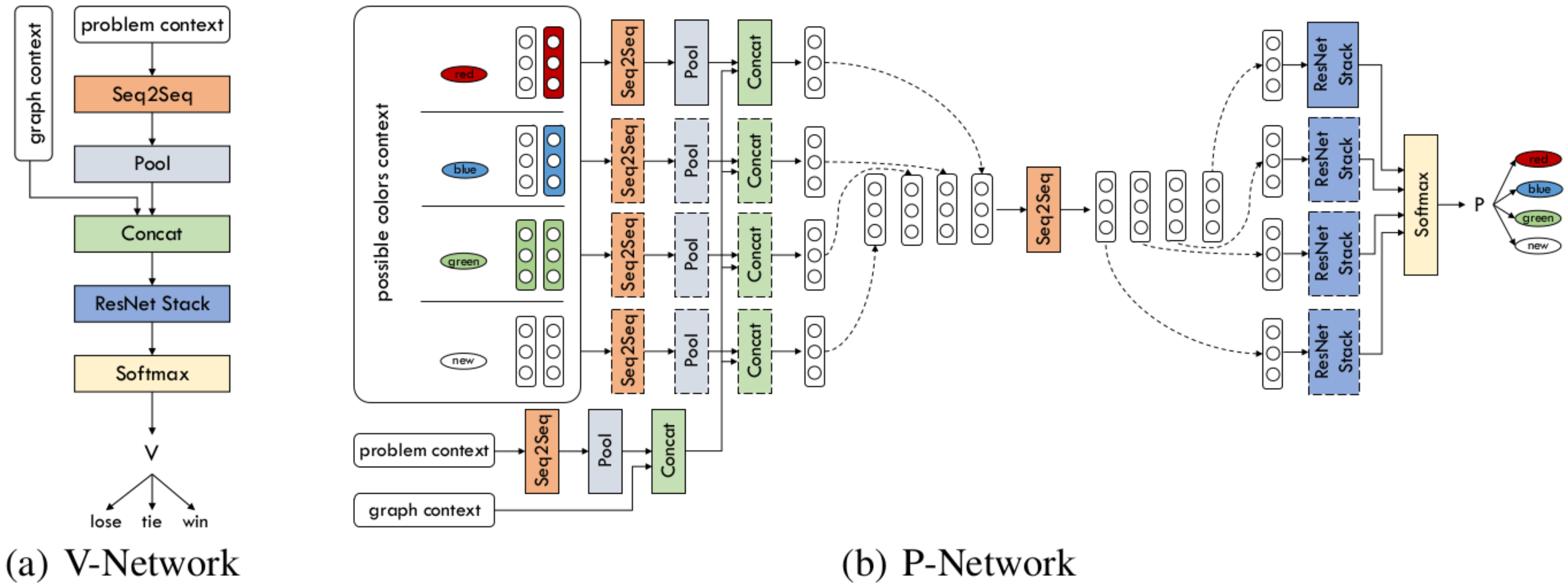


Figure 5: Fast Color Net Layers

High Performance Training System

- Use data-parallelism for training.
- stochastic gradient descent with MPI for communication.
- The MCTS is completely data-parallel.

Empirical Analysis of Results

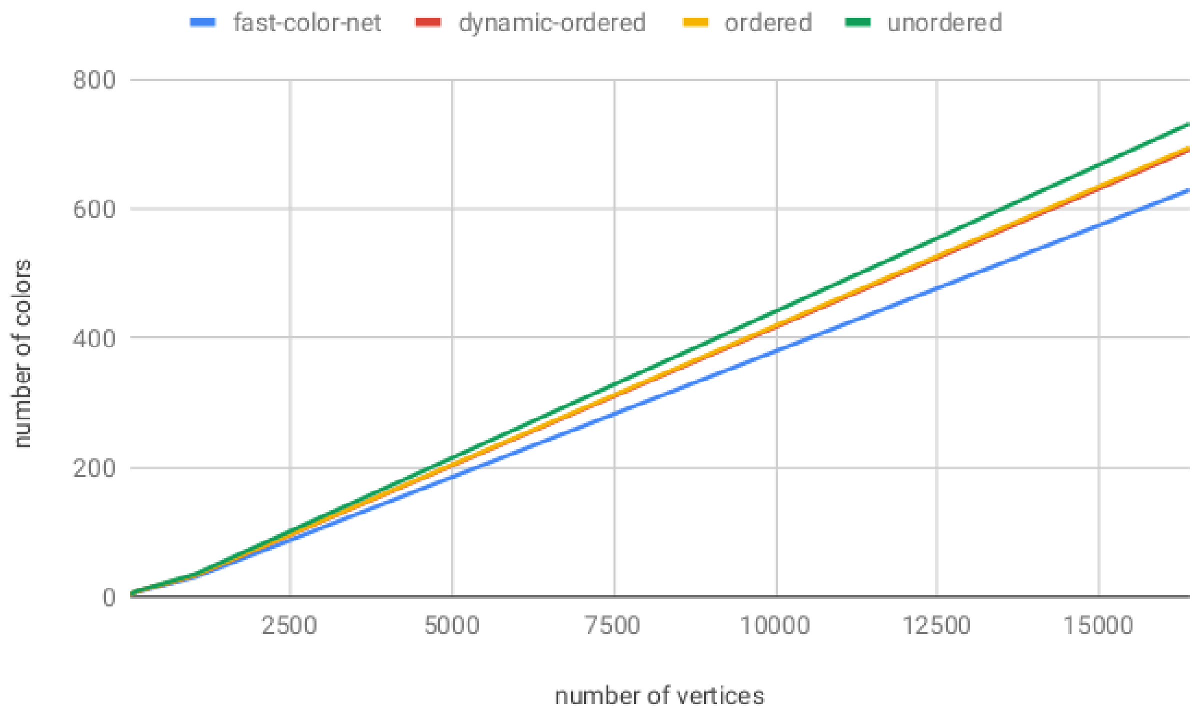


Figure 6: The number of colors used on the WS graph test sets as a function of vertex count.

Empirical Analysis of Results

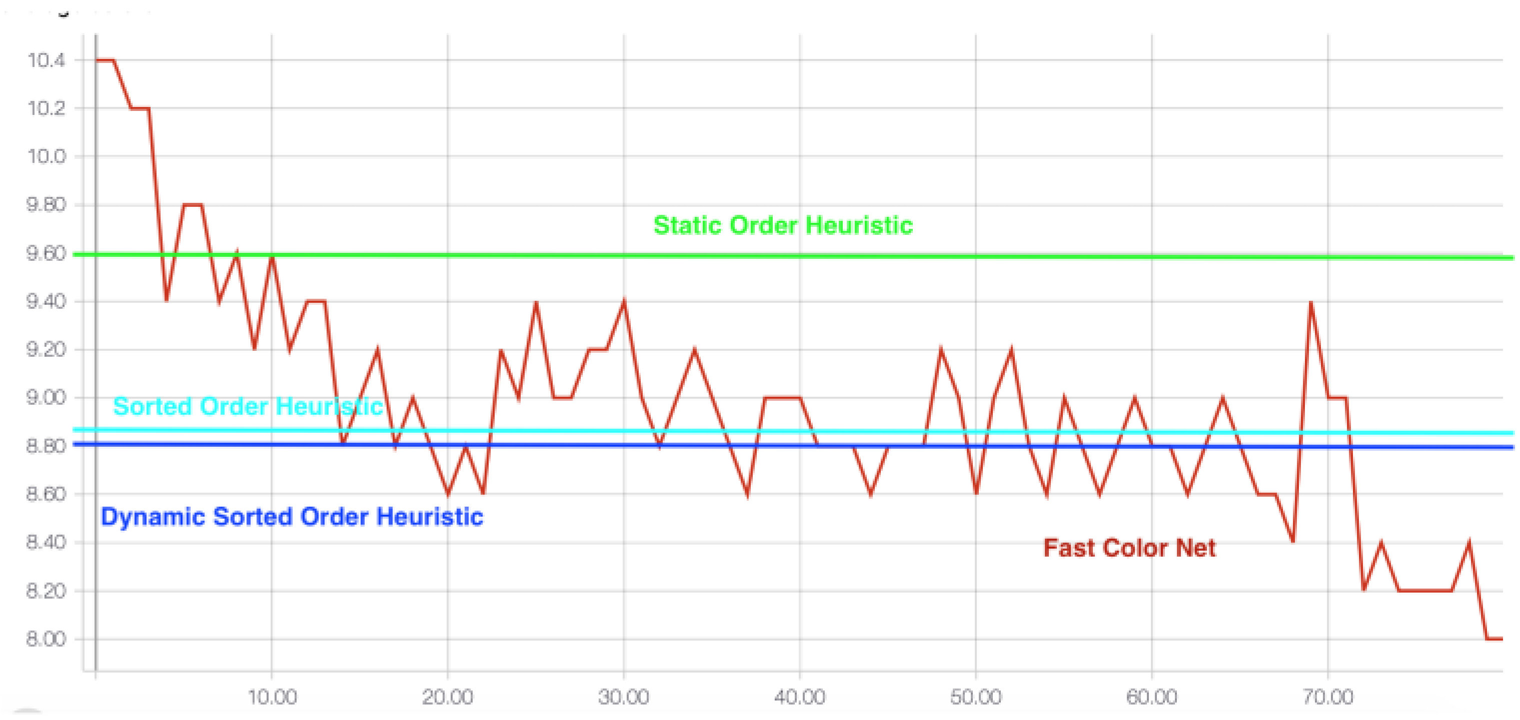


Figure 7: Improvement in learned heuristic with more training.

Empirical Analysis of Results

Dataset	ER-1K	WS-1K	ER-16K	WS-16K	ER-10M	WS-10M	SS-CIR	SS-LP	SS-Web	SS-FE
Unordered	34.3	59.2	732.8	265.35	42923	16415	4.2	4.25	3.75	4.85
Ordered	32.45	57.35	715.2	261.8	40347	15922	3.15	**2.95**	2.6	4.05
Dynamic	32.2	57.15	708.5	261.2	37524	15843	3.55	3.15	2.7	4.25
FCN-train	**29.58**	**52.5**	**660.19**	**237.03**	**35362**	**14924**	**3.0**	**2.95**	**2.4**	**3.75**
FCN-test	31.7	56.59	702.57	258.3	37849	15023	3.1	**2.95**	2.55	4.1
FCN-gen	33.9	57.66	708.13	267.53	43415	17262	4.15	4.3	3.7	4.95

Table 2: The average number of colors across our test sets. FCN is our FastColorNet architecture. SS means SuiteSparse (CIR: circuits, LP: linear programming, FE: finite-element). FCN-train represents performance when a graph present in the training set is evaluated on, FCN-test uses a model trained on the same type of graph, and FCN-gen tests generalization performance of a model trained on random graphs of many sizes.

Criticism

- In most cases, Dynamic performs better than FCN-test and FCN-Gen.
- They do not mention the time of each algorithm in the results.

Conclusion

- Convert graph coloring to an MDP.
- Convert the problem to a zero-sum game.
- Solve the zero-sum game using FastColorNet + MCTS.
- Evaluate for different types of graphs.

Reference

- [1] Huang, J., Patwary, M., & Diamos, G. (2019). Coloring big graphs with alphagozero. *arXiv preprint arXiv:1902.10162*.

Thank you!