Nested Rollout Policy Adaptation for Monte Carlo Tree Search

Hengwei Guo

Outline

- **Background Build Up:**
  - Monte Carlo Search Algorithm
  - Monte Carlo Tree Search (MTCS)
    - Nested Monte Carlo Tree Search (NMCS)
- **Nested Rollout Policy Adaptation for MCTS (NRPA)**
  - experiments
  - performance
- **Summary**

Outline

- **Background Build Up:**
  - **Monte Carlo Search Algorithm**
  - Monte Carlo Tree Search (MTCS)
  - Nested Monte Carlo Tree Search (NMCS)
- **Nested Rollout Policy Adaptation for MCTS (NRPA)**
  - Test Domain:
    - Crossword Puzzle Construction
    - Morpion Solitaire: improves upon 35 year old record
  - performance
- **Summary**

Background Knowledge Recap

- **Simplest Monte Carlo Search Algorithm**
  - Iterative Sampling
  - it consists in playing random games until a solution is found or the search time is elapsed.
  - Estimate value function using sample episodes
  - Suitable for episodic tasks
    - No matter what actions we take, episode will terminate at some point (could take a long time though)
  - Will update on an episode-by-episode basis
    - Not action-by-action

First-Visit Monte Carlo Prediction

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

Generate an episode using π

For each state s appearing in the episode:

$G \leftarrow$ return following the first occurrence of s

Append G to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

**from lecture notes*

Monte Carlo Control with ES

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \leftarrow$ arbitrary
    $\pi(s) \leftarrow$ arbitrary
    $Returns(s,a) \leftarrow$ empty list

Repeat forever:
    Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
    Generate an episode starting from S_0, A_0, following π
    For each pair s, a appearing in the episode:
        $G \leftarrow$ return following the first occurrence of s, a
        Append G to $Returns(s,a)$
        $Q(s,a) \leftarrow$ average$(Returns(s,a))$
    For each s in the episode:
        $\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

**from lecture notes*

MC Control with ϵ-Soft Policies

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s,a) \leftarrow$ arbitrary

$Returns(s,a) \leftarrow$ empty list

$\pi(a|s) \leftarrow$ an arbitrary ε-soft policy

Repeat forever:

(a) Generate an episode using π

(b) For each pair s, a appearing in the episode:

$G \leftarrow$ return following the first occurrence of s, a

Append G to $Returns(s,a)$

$Q(s,a) \leftarrow \text{average}(Returns(s,a))$

(c) For each s in the episode:

$A^* \leftarrow \arg\max_a Q(s,a)$

For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

*from lecture notes

Outline

- **Background Build Up:**
  - Monte Carlo Search Algorithm
  - **Monte Carlo Tree Search (MTCS)**
    - Nested Monte Carlo Tree Search (NMCS)
- **Nested Rollout Policy Adaptation for MCTS (NRPA)**
  - experiments
  - performance
- **Summary**

- **Monte Carlo Tree Search**

- The MCTS algorithms produce very good solutions to problems where other traditional methods fail (**when you don’t have a good heuristic**).

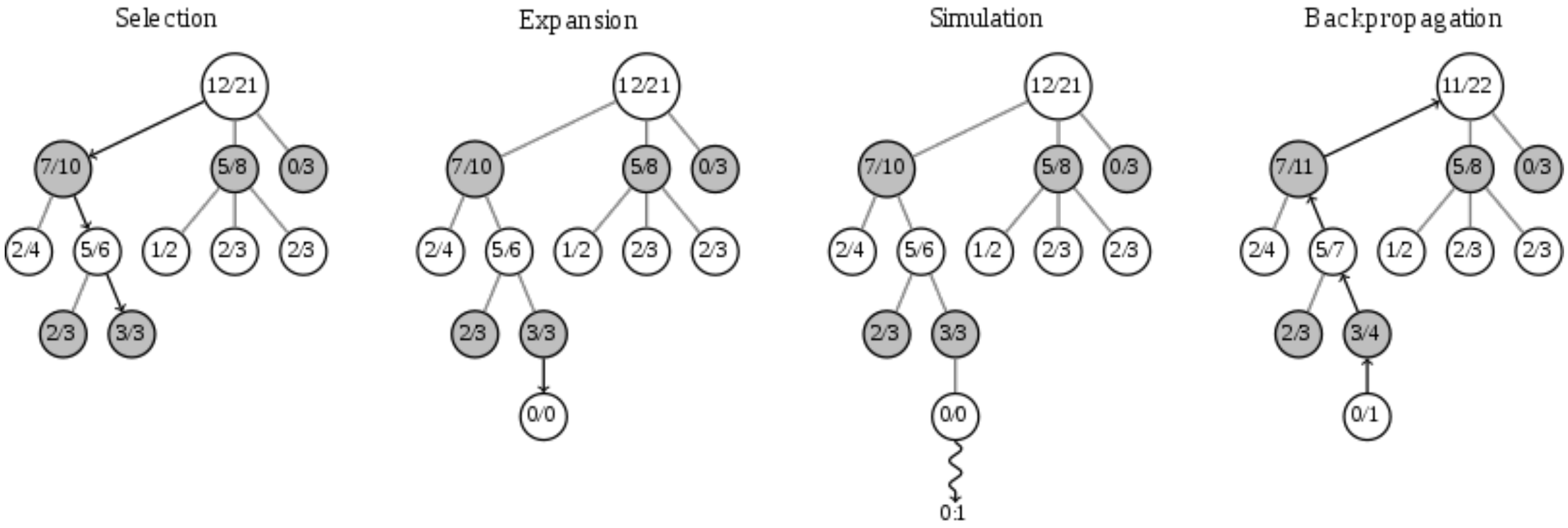
- These problems usually tend to have a very **large search space** when we cannot afford an exhaustive search, such as games like Go, Chess or SameGame.

- simulation strategy is combined with a tree search to guide the search to more promising branches of the search tree. This often produces very good results within a reasonable amount of time.

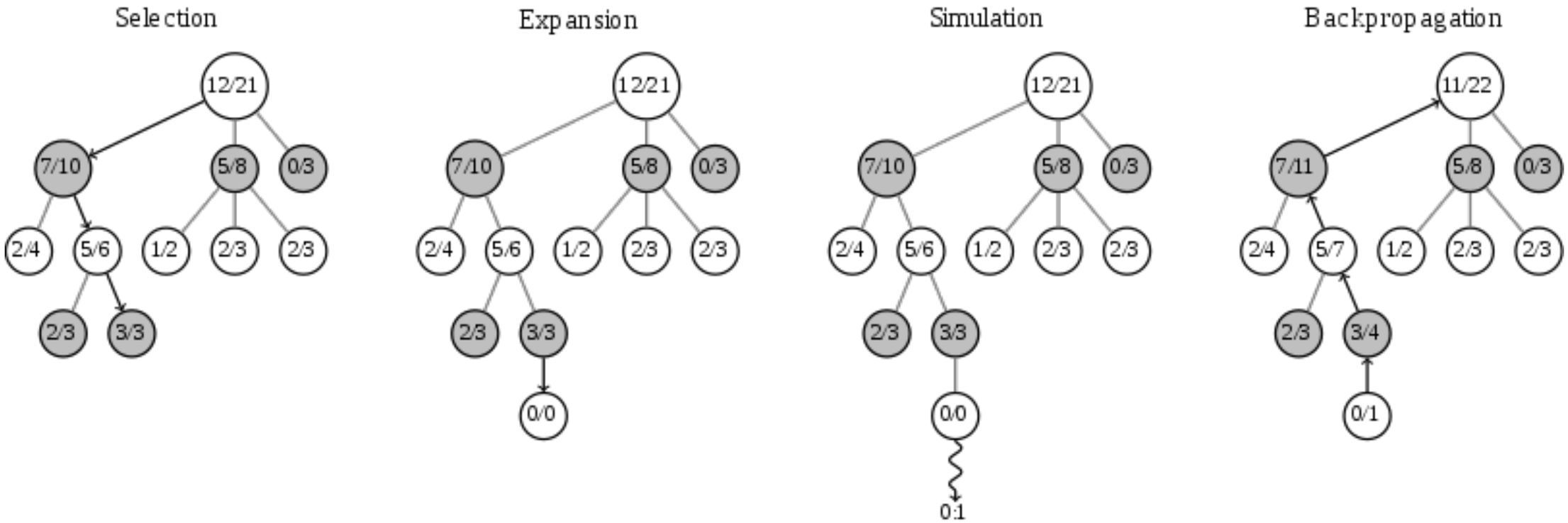
- **mostly static rollout policy**

Background Knowledge Recap

- **Monte Carlo Tree Search**



** picture from https://en.wikipedia.org/wiki/Monte_Carlo_tree_search*



- **Selection**: start from root R and select successive child nodes down to a leaf node L.

- **Expansion**: unless L ends the game with a win/loss for either player, either create one or more child nodes or choose from them node C.

- **Simulation**: play a random playout from node C.

- **Backpropagation**: use the result of the playout to update information in the nodes on the path from C to R.

** picture from https://en.wikipedia.org/wiki/Monte_Carlo_tree_search*

- **Monte Carlo Tree Search**

- Can MCTS be further improved?
    - We want to further utilize the results from previous searches
    - We want to update from a subtree rather than a single branch

Outline

- **Background Build Up:**
  - Monte Carlo Search Algorithm
  - Monte Carlo Tree Search (MTCS)
    - **Nested Monte Carlo Tree Search (NMCS)**
- **Nested Rollout Policy Adaptation for MCTS (NRPA)**
  - experiments
  - performance
- **Summary**

Background Knowledge Recap

- **Nested Monte Carlo Tree Search**
  - MCTS with nested expansion level, L
  - also memorizes the best sequence found by a previous search

Nested Monte Carlo Tree Search (NMCS)

- A NMCS repeats the following steps until time runs out or until the search terminates.

- **Selection**

  - During the first iteration the initial state (root) is selected.

  - Otherwise the action leading to the current best score is selected.

- **Simulation**

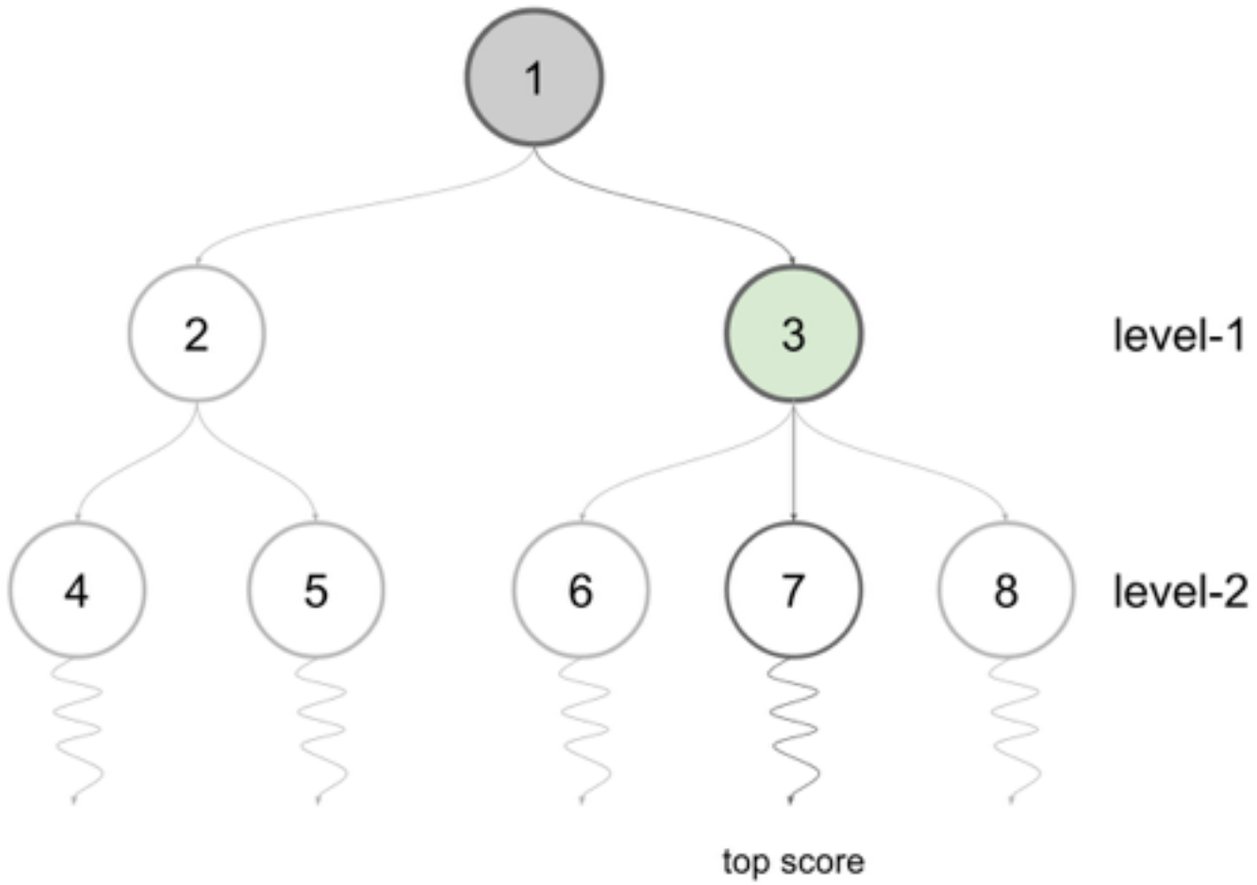
  - For the previously selected state all legal actions are determined.

  - For each of these next actions a simulation is played out.

- **Backpropagation**

  - The best result found during the previous simulation step is stored in memory.

Example of a level-2 search step



Explanation:

1. Node 1 is selected.
2. Next, a complete 2 level deep subtree of the search space with 1 as its root is traversed. This subtree has 5 leafs, 4 – 8.
3. For each leaf a normal simulation is played out.
4. The best result is found by the simulation from node 7. The result is propagated back and the next node from the best result (node 3) is selected for the next iteration.

* picture from http://blog.leifbattermann.de/2016/06/05/nested-monte-carlo-search/

Outline

- **Background Build Up:**
  - Monte Carlo Search Algorithm
  - Monte Carlo Tree Search (MTCS)
    - **Nested Monte Carlo Tree Search (NMCS)**
- **Nested Rollout Policy Adaptation for MCTS (NRPA)**
  - experiments
  - performance
- **Summary**

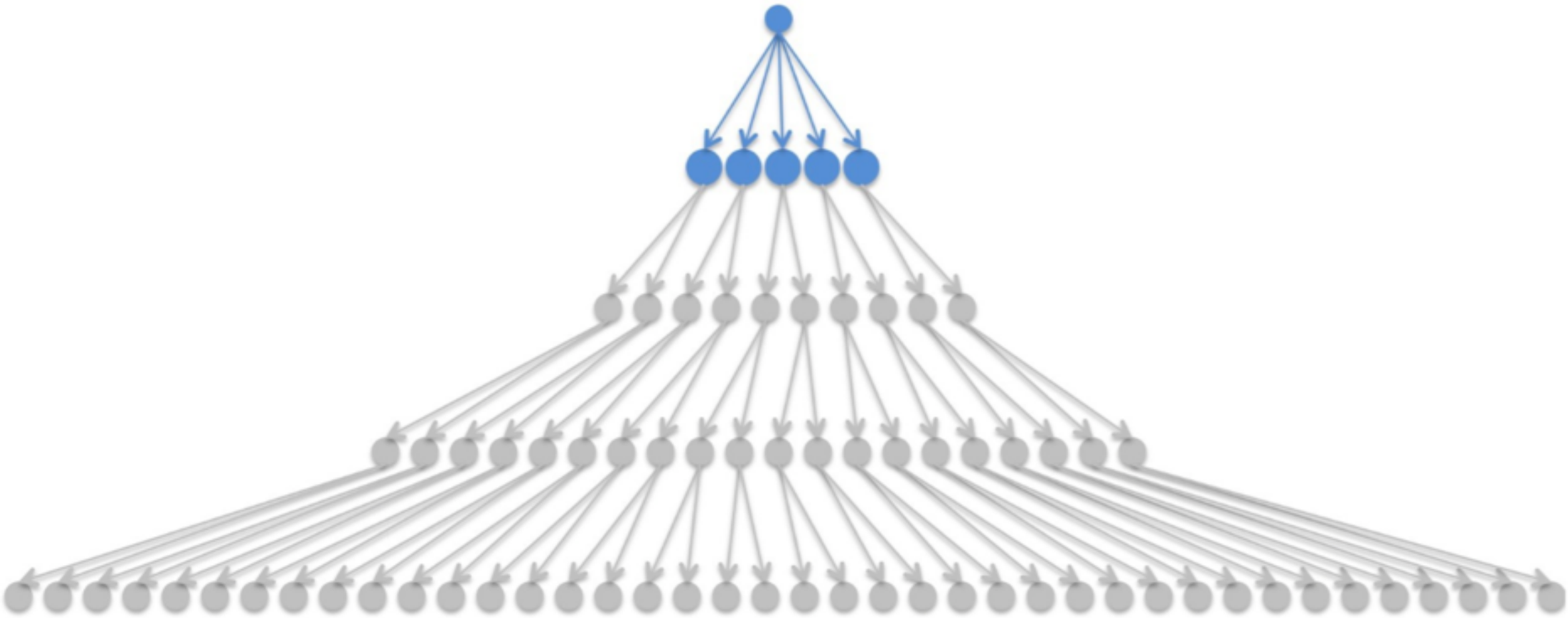
Following slides are adapted from

Christopher Rosin
Parity Computing, Inc.
San Diego, California, U.S.A.

christopher.rosin@gmail.com

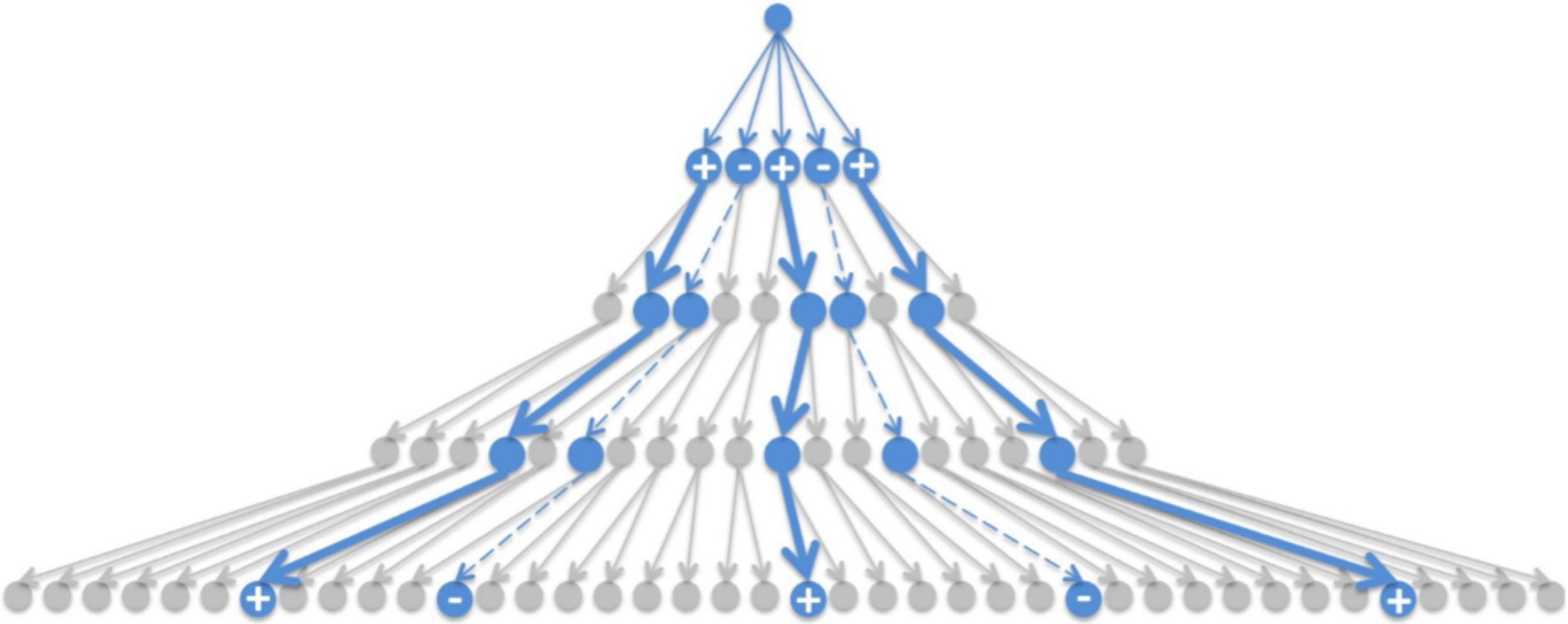
- **Can NMCS be further improved?**

How to select nodes when far away from the leaves?



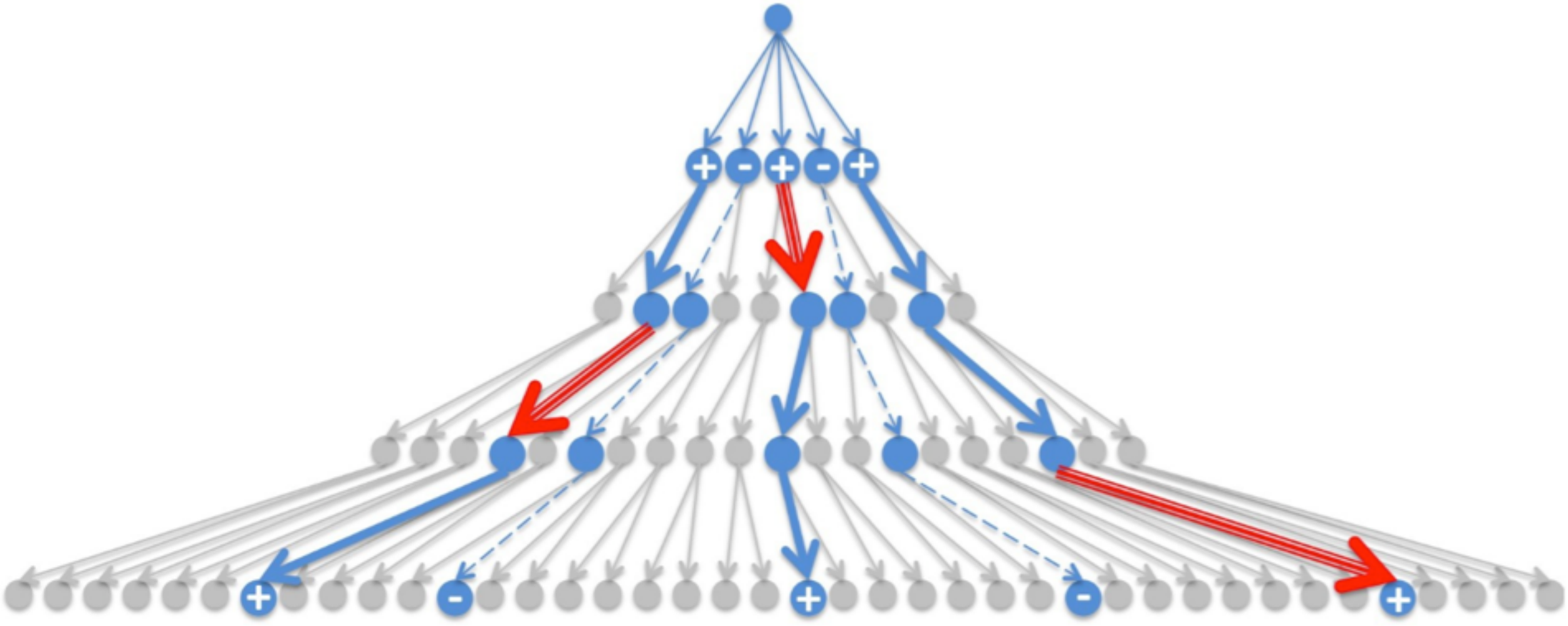
Monte Carlo Tree Search

Favor nodes which reach successful leaves (+) via **rollouts**



Monte Carlo Tree Search: Rollout Policy Adaptation

Identify & favor **successful actions** across rollout trajectories



NRPA: Rollouts

policy[x]: adaptable parameter on actions with code x

From root to leaf:

Probability of action i ~ $e^{\text{policy[code(action i)]}}$

NRPA: Level 1 Search

For N iterations:

- seq = **rollout(policy)**
- If score(seq)>=score(best) then best=seq
  - **Adapt policy**: gradient ascent on log(Probability of best)

Return best

Parameters:

- N: # iterations (N=100)
- Gradient ascent step size (1.0)

NRPA: Nested Search

NRPA(level,policy)

For N iterations:

- If level=1 then seq = rollout(policy)

  else seq = **NRPA(level-1,policy)**

- If score(seq)>=score(best) then best=seq
- **Adapt policy**: gradient ascent on log(Probability of best)

Return best

Outline

- **Background Build Up:**
  - Monte Carlo Search Algorithm
  - Monte Carlo Tree Search (MTCS)
    - **Nested Monte Carlo Tree Search (NMCS)**
- **Nested Rollout Policy Adaptation for MCTS (NRPA)**
  - experiments
  - performance
- **Summary**

Test Problem: Crossword Puzzle Construction

- Given dictionary & empty grid
- Make a legal crossword with subset of dictionary
- Score is #words; break ties with #letters

```
SALEM..
T.A.A.A
P.N.D.L
A.S.I.B
U.I.S.A
LINCOLN
..G.N.Y
```

NRPA code identifies word & position & orientation

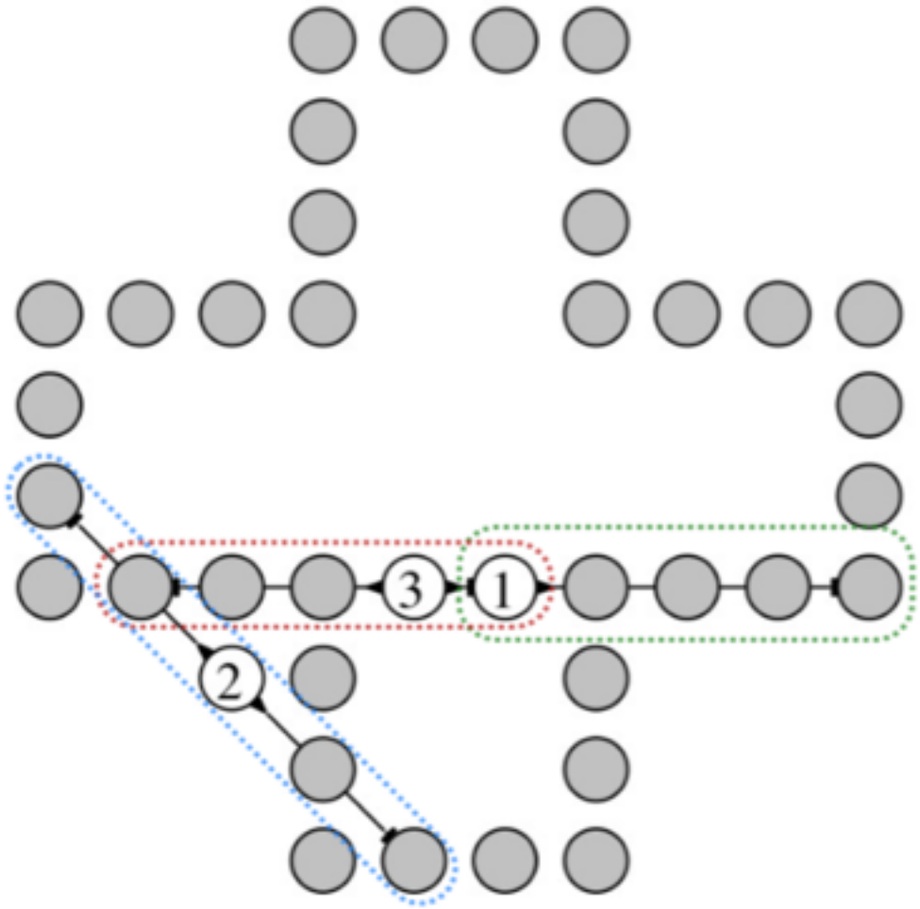
GAMES Magazine contest instances:

- **CrossC** (2006): 63-word dictionary, 19x19 grid
- **CrossP** (1994): 52-word dictionary, 19x19 grid

Contests established high quality winning solutions.

Test Problem: Morpion Solitaire

- Add dots to form lines of 5 dots
- Score is #lines



- **MorpD:** parallel lines must have disjoint endpoints
- **MorpT:** endpoints may touch

NRPA code uniquely identifies line

MorpT records:

- 146: Prior computer record (Akiyama 2010: NMCS-based)
- 170: Human-generated record (Bruneau, 1976)
- 177: New record obtained with NRPA

(see morpionsolitaire.com)

Outline

- **Background Build Up:**
  - Monte Carlo Search Algorithm
  - Monte Carlo Tree Search (MTCS)
    - **Nested Monte Carlo Tree Search (NMCS)**
- **Nested Rollout Policy Adaptation for MCTS (NRPA)**
  - experiments
  - performance
- **Summary**

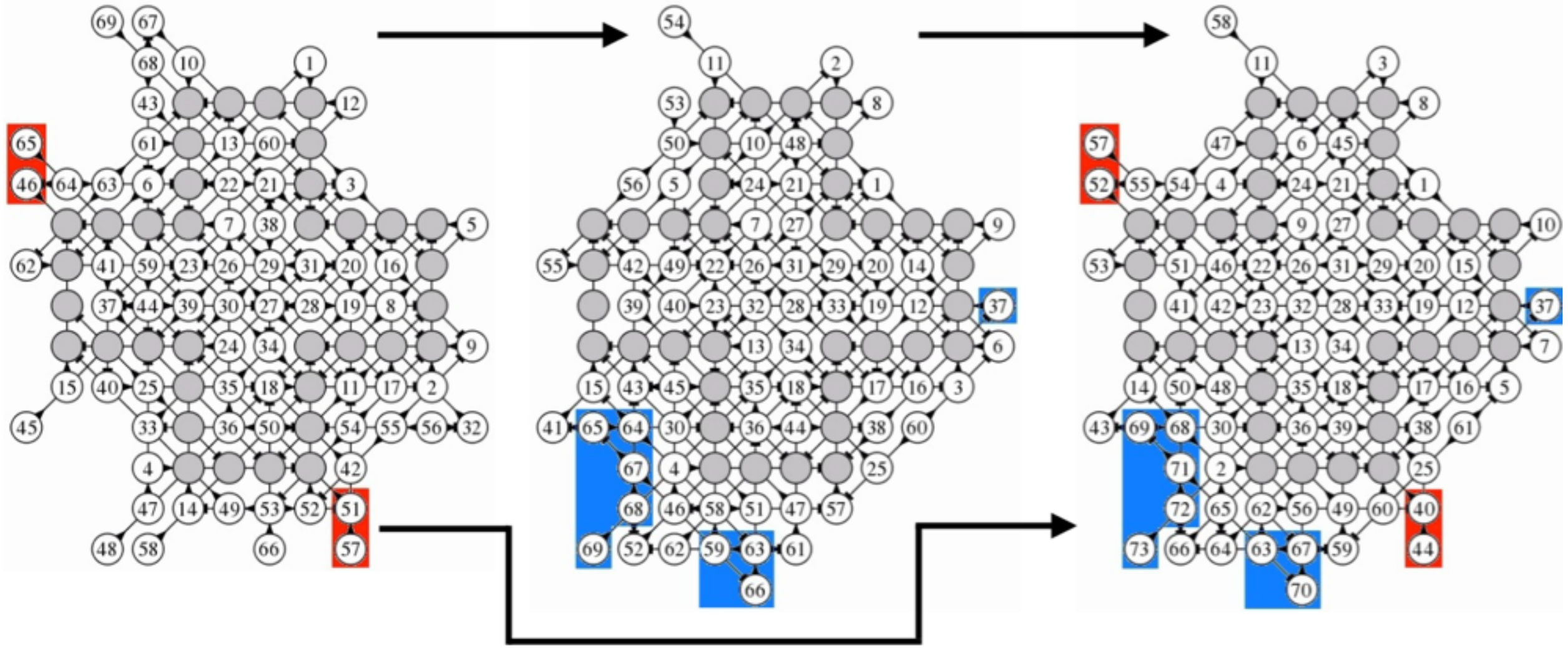
Test Problem Summary

	Max Depth	Max Branching Factor	Old Record	New NRPA Record	NRPA Runs
MorpT	177	46	170 (Bruneau, 1976)	177	28 Runs @ Level 5
MorpD	82	46	80 (Cazenave, IJCAI'09)	82	40 Runs @ Level 4
CrossC	37	1197	36 words, 256 letters	37 words, 259 letters	40 Runs @ Level 4
CrossP	28	988	28 words, 236 letters	28 words, 238 letters	40 Runs @ Level 4

NRPA Behavior

Permutation enables alternative **extension**

Policy retains history, enabling **hybrid**

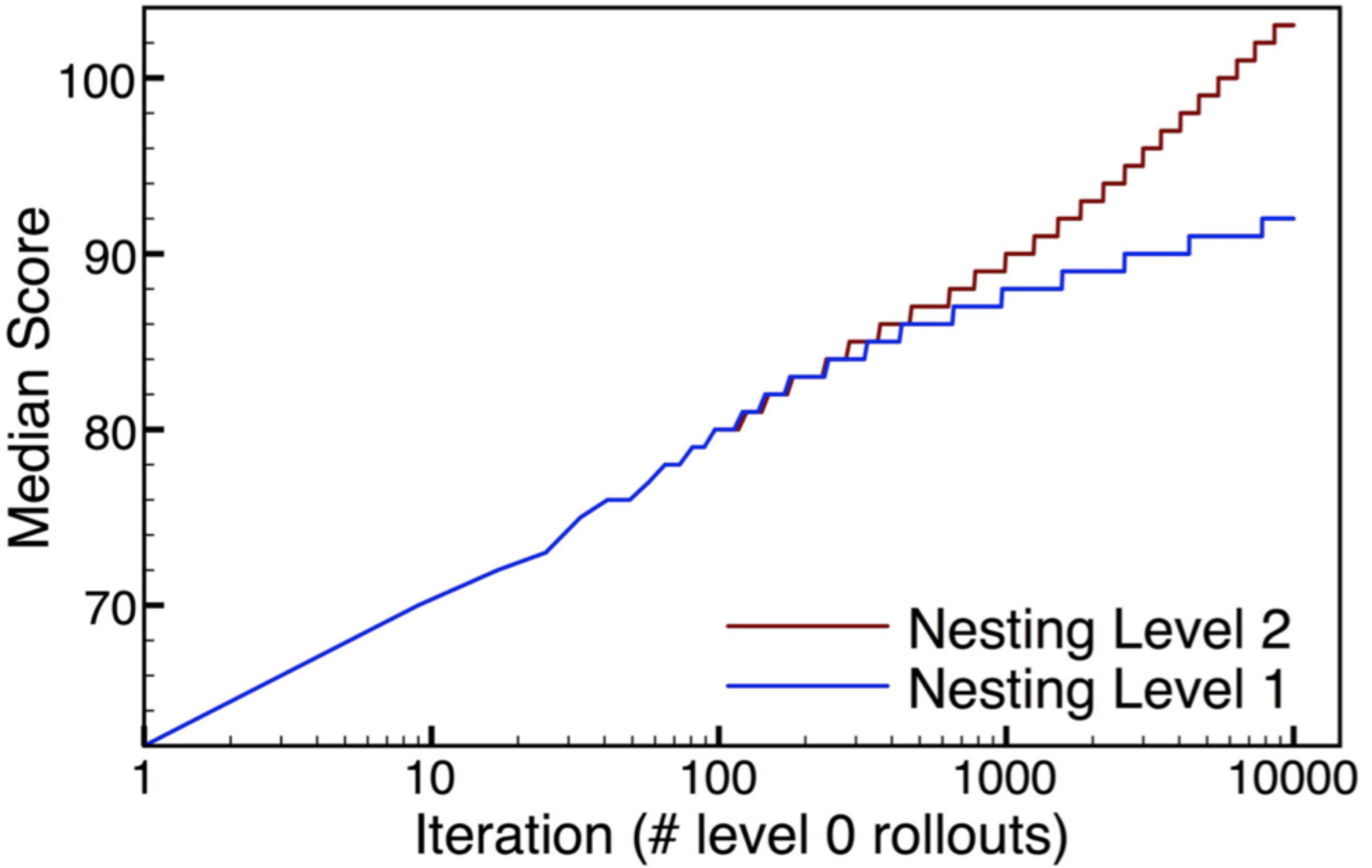


Score 69

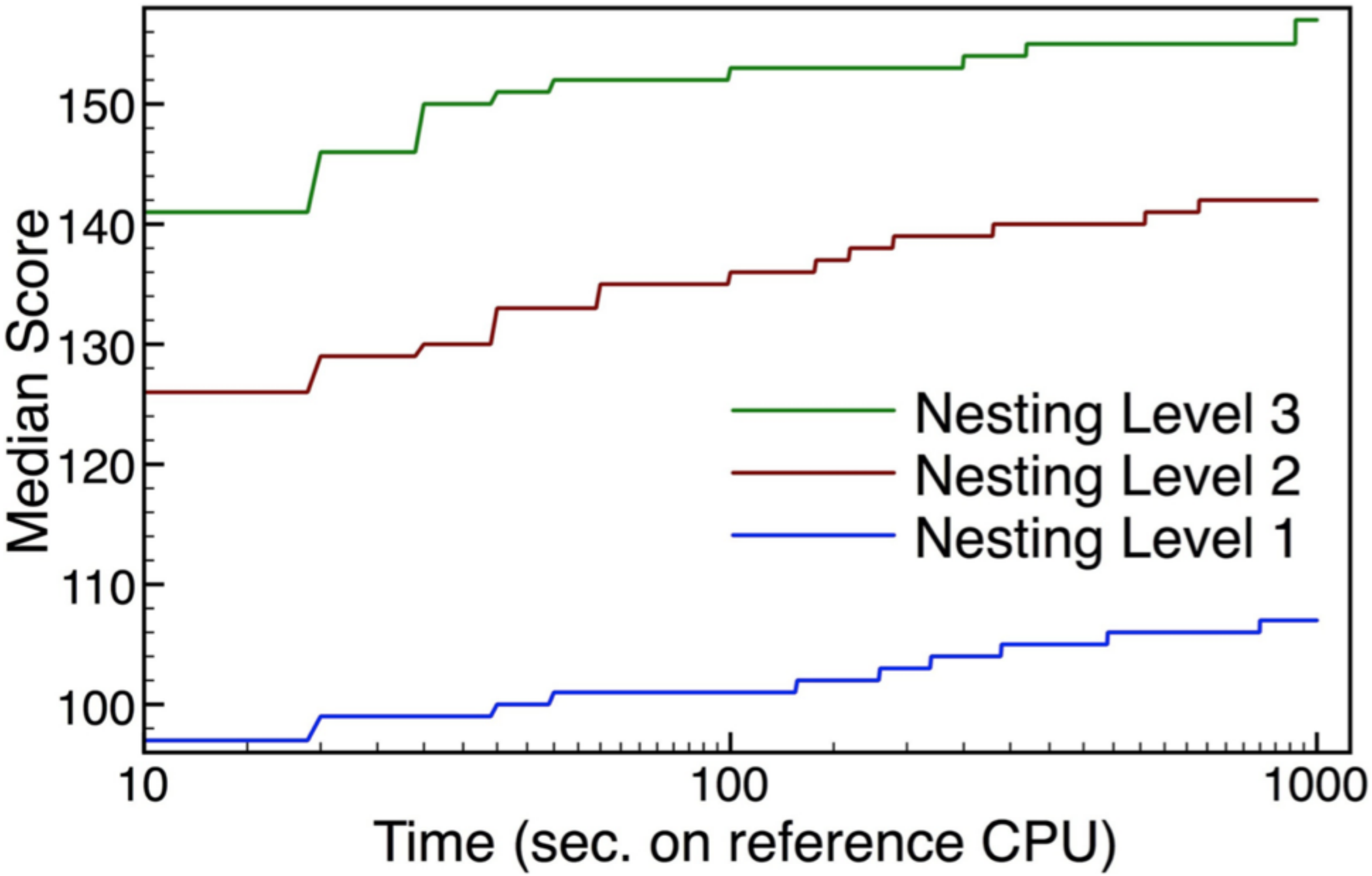
Score 69

Score 73

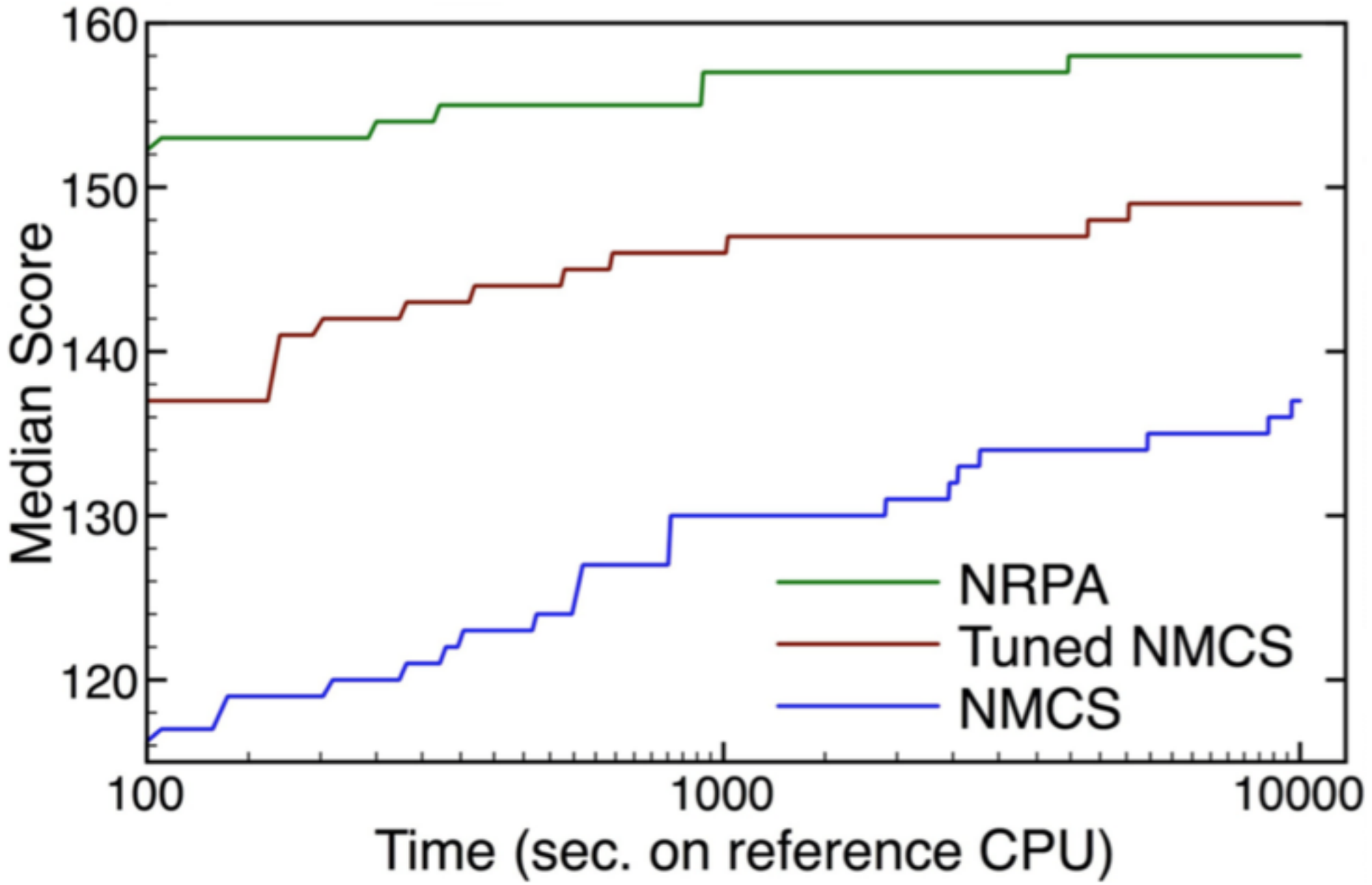
NRPA on MorpT



NRPA on MorpT



NMCS Performance Comparison



- NRPA best even if NMCS runs 100x longer
  - Even when NMCS uses rollout policy tuned for MorpT
- Other 3 problems: NRPA best even if NMCS runs 10x longer

Conclusion

- NRPS successfully searches via rollout policy adaptation