Refutation of Spectral Graph Theory Conjectures with Monte Carlo Search

Milo Roucairol and Tristan Cazenave

LAMSADE, Université Paris Dauphine - PSL, CNRS

Abstract. We demonstrate how Monte Carlo Search (MCS) algorithms, namely Nested Monte Carlo Search (NMCS) and Nested Rollout Policy Adaptation (NRPA), can be used to build graphs and find counter-examples to spectral graph theory conjectures in minutes.

Keywords: Monte Carlo Search · Graph Theory · Conjecture · Refutation.

1 Introduction

Monte Carlo search algorithm have proven to be powerful as game playing agents, with recent successes like AlphaGo[19]. These algorithms have the advantage of only needing an evaluation function for the final state of the space they explore.

Graph conjectures are propositions on graph classes (any graph, trees, K-free...) that are suspected to be true and are awaiting a proof or a refutation. They lend themselves well to computer assisted proofs, as finding a counter example can be tedious to do manually. Spectral graph conjectures are appropriate to automated refutation because the property can often directly be turned into the evaluation function that can take many different values. Thanks to software like Auto-GraphiX [18] and Graffiti [10], there are plenty of such conjectures.

Adam Zsolt Wagner showed that one could find explicit counter-examples using deep reinforcement learning of a policy with the deep cross entropy method [26].

In this paper, the MCS algorithms will play the game of refuting conjectures by building counter-examples.

First we will present the refutation of graph theory conjectures, then the different algorithms we use to explore the problem space, after that the procedure we use to build graphs and the game rules, finally we expose our results on four different conjectures.

2 Refutation of Graph Theory Conjectures

2.1 Graph Theory Conjectures

Conjectures in graph theory can be difficult to refute manually, unless one has an intuition of a counter-example, building a large number of graphs and computing invariant values or NP-hard problems on them often results in a waste of time. Hopefully, computers can help us with these score computations, the goal of automated conjecture refutation is to automatize the exploration as well.

Multiple types of graph theory conjectures exist: existence, topological, flow based, connectivity, cycle, minors, spectral... The conjectures we are examining here are the spectral ones, they have the advantage of only requiring matrices calculations. The spectrum of a matrix is the set of its eigenvalues, spectral graph theory conjectures include conjectures on spectrum related invariants on different types of graph related matrices (adjacency, distance, laplacian...).

In this article we aim at evaluating the interest of Monte Carlo Search methods in order to refute some spectral graph theory conjectures. We are interested in conjectures that Adam Zsolt Wagner refuted in his article [26] as well as in other conjectures.

2.2 Algorithms Used to Refute Graph Conjectures

Wagner used deep neural networks with cross entropy. The network is used to learn a policy from a state, it is trained by constructing a batch of graphs according to the policy, evaluating and selecting the best graphs from the batch to adjust the neural network with their state-moves couples according to their scores (obtained during the evaluation), then it's repeated starting by the generation of a new batch until the conjecture is refuted.

Some of our conjectures come from Aouchiche and Hansen survey [1]. In order to explain how Graffiti selects conjectures they state:
"Graffiti generates many conjectures of a simple form (e.g. inequalities between two invariants or between an invariant and the sum of two others) then tests them on a database of graphs and discards those which are falsified. Should this test be passed, it is checked if the formulas are implied by known ones (in which case they are also discarded) and that they provide new information for at least one graph in the database, i.e., that they are stronger than the conjunction of all other formulas for that graph. If not, they are temporarily set aside. If yes, they are proposed to all graph theorists, in the electronic file "Written on the Wall", which reports on the status of almost 1000 conjectures. Many well-known graph theorists worked on these conjectures and this led to several dozen papers. Some of Graffiti's conjectures are about various topics in spectral graph theory, namely, the eigenvalues, as well as their multiplicity, of the adjacency, Laplacian and distance matrices of graphs."

3 Algorithms

In this section we present Monte Carlo Search and the different search algorithms we use: NMCS, NRPA and Greedy-BFS.

3.1 Monte Carlo Search

Monte Carlo Tree Search (MCTS) has been successfully applied to many games and problems [5].

Nested Monte Carlo Search (NMCS) [6] is an algorithm that works well for puzzles and optimization problems. It biases its playouts using lower level playouts. At level zero NMCS adopts a uniform random playout policy. Online learning of playout strategies combined with NMCS has given good results on optimization problems [24]. Other applications of NMCS include Single Player General Game Playing [20], Cooperative Pathfinding [3], Software testing [22], heuristic Model-Checking [23], the Pancake problem [4], Games [7] and the RNA inverse folding problem [21].

Online learning of a playout policy in the context of nested searches has been further developed for puzzles and optimization with Nested Rollout Policy Adaptation (NRPA) [25]. NRPA has found new world records in Morpion Solitaire and crosswords puzzles. NRPA has been applied to multiple problems: the Traveling Salesman with Time Windows (TSPTW) problem [8, 12], 3D Packing with Object Orientation [14], the physical traveling salesman problem [15], the Multiple Sequence Alignment problem [16] or Logistics [13]. The principle of NRPA is to adapt the playout policy so as to learn the best sequence of moves found so far at each level.

The use of Gibbs sampling in Monte Carlo Tree Search dates back to the general game player Cadia Player and its MAST playout policy [17].

3.2 Nested Monte Carlo Search

NMCS [6] is a Monte Carlo Search algorithm that recursively calls lower level NMCS on children states of the current state in order to decide which move to play next, the lowest level of NMCS being a random playout, selecting uniformly the move to execute among the possible moves. A heuristic can be added to the playout choices, but it is not the case with the NMCS used here.

Algorithm 1 gives the NMCS algorithm. The different notations used are:

$M_{current\text{-}state}$ denotes the legal moves available from the state *current-state*.

$randomChoice(L)$ is a function that returns an element selected uniformly from L.

3.3 Nested Rollout Policy Adaptation

Introduced by Christopher D. Rosin [25], NRPA is akin to mixing NMCS [6] and Q-learning [27], performances are generally better than NMCS but the exploration/exploitation can lock itself in a local minimum. The algorithm is also dependent on how moves are defined.

The algorithm uses a policy which consist in attributing a value to a move. The difference with Wagner's policy [26] is here the value of the move is not necessarily learned given the state of the graph, but given the definition of the move, the policy can thus be defined given the few previous moves or the number of moves preceding the current one. It is also a key difference between NRPA policy learning and Q-learning, Wagner's policy is closer to deep Q-learning.

Like the NMCS, the NRPA calls other lower level NRPA which start with the higher level current policy, these lower level NRPA return their best sequence which is used to update the higher level policy. At the lowest level, the NRPA launches a playout and randomly select moves according to the policy.

$softmaxChoice(L, policy)$ is a function that returns an element selected according to a softmax distribution on each element value mapped on the policy. Chances of selecting e from L are $\frac{exp(policy[e])}{sum([exp(policy[i]) for\ i\ in\ L])}$.

In all our experiments, NRPA's $Alpha$, the learning rate, is set to 1.

Algorithm 2 gives the NRPA algorithm.

3.4 Greedy Best First Search

Another algorithm we use is Greedy-BFS [11]. It consists in opening the best (or randomly one of the best) previously evaluated node, evaluating all its children and inserting these children in a list, sorted by their evaluation scores.

The evaluation can be done with a heuristic (in which case it is not Monte Carlo) or a Monte Carlo algorithm like a playout or a low level NMCS.

In this paper, the Greedy-BFS was used as a general search technique and not a Monte Carlo search algorithm. We use the score function (or the modified evaluation function in conjecture 2) to choose which leaves to expand.

4 Graph Generation

For each problem we want to solve, we define an interface. This interface provides, to the search algorithm, the classes for the *state* and for the *move* and their associated functions: *score*, *terminal*, M_{state} and *play*.

The goal is to find an instance of a class of graphs that does not respect a property speculated on that class of graphs, in order to do so we use reinforcement learning and tree search algorithms to explore the possible graphs from that class and hopefully converge to a counter-example.

Moves to generate graphs are the same in all the approaches presented here, the graph is built edge by edge. Moves are represented as couple of integers corresponding to the two vertices the edge will link. If one of the two integer corresponding vertex is not in the graph yet, it is added alongside the edge as a leaf (-1 as the second member of the couple means adding a leaf anyway).

What differs from one conjecture (interface) to another are the methods :
- *score* which derives from the conjecture we want to refute.
- *terminal* is necessary to MCS methods to know when stop a playout, it can be the size of the graph (what we used here) but also the number of edges or any property on the graph. Trying different constraints for that method is part of the experiments.
- M_{state} gives the legal moves from a state according to the model. For example a model of maximum degree 3 will exclude any moves that result in the edge addition on a vertex of degree 3.

5 Experiments on Conjectures

The experiments were made with Rust 1.59, on a Intel Core i5-6600K 3.50GHz using a single core (but parallel processing is very accessible).

Every solution was verified using WolframAlpha symbolic computing of eigenvalues to ensure they were not due to floating point errors in Rust nalgebra 0.30.1 library.

We express every score function in order to maximise them.

5.1 Conjecture 1. AutoGraphiX

Conjecture 7 from [1], also present as conjecture 2.1 from [26], is used as a proof of concept in the aforementioned paper. We decided to refute all conjectures from [26] to show that MCS can be equally as good as the deep cross entropy.

Definition 1. *Let G be a connected graph, the matching number is the size of the matching that contains the largest number of edges of G.*

Definition 2. *Let G be a connected graph on n vertices, the distance spectrum $\lambda_1, ..., \lambda_n$ is the spectrum of the distance matrix, ranked in descending order (λ_1 being the largest, the index).*

This conjecture is the following :

Conjecture 1. *Let G be a connected graph on $n \geq 3$ vertices with index λ_1 and matching number μ. Then $\lambda_1 + \mu \geq \sqrt{n-1} + 1$.*

$$\lambda_1 + \mu \geq \sqrt{n-1} + 1 \Leftrightarrow 0 \geq \sqrt{n-1} + 1 - \lambda_1 - \mu$$

Which gives

Score function : $\sqrt{n-1} + 1 - \lambda_1 - \mu$

With a NRPA of level 3, and generating trees of size 19, we found a graph with a positive score of 0.016: $\lambda_1 + \mu \approx 5.227$ and $\sqrt{n-1} + 1 \approx 5.243$
It is the same counter-example (figure 1) as Adam Zsolt Wagner [26]. His results

helped us as we knew what to search for and that we could exploit NRPA policy learning to recreate the pattern found in his counter example. However it is interesting to try generating trees first anyway when refuting a spectral graph conjecture as the space to explore is way smaller than for any graph, and trees are extreme instances of the class, so these tend to be counter-examples.

The NRPA algorithm can lock itself in a policy leading to a local minimum while searching a counter-example to that conjecture. It is important to use restarts in this case: if a result is not found in less than a second the algorithm is launched again a few number of times (usually 10 to 20 times) until the refutation is found. While it takes a few hours to find the counter example with Wagner's deep cross entropy [26], our method can find it in seconds even with multiple relaunches. Using restarts this way can be parallelized easily.

We also were able to find a counterexample of size 18 (figure 2) with the same process, with a positive score of 0.012: $\lambda_1 + \mu \approx 5.101$ and $\sqrt{n-1} + 1 \approx 5.123$

5.2 Conjecture 2. Aouchiche-Hansen

Conjecture 2.15 from [2], also present as conjecture 2.3 from [26].

Definition 3. *Let G be a connected graph, the diameter is the maximum length of the shortest paths between all vertices of G.*

Definition 4. *Let G be a connected graph, the proximity is, over all the vertices of G, the minimum average distance to all the other vertices of G.*

Conjecture 2. *Let G be a connected graph on $n \geq 4$ vertices with diameter D, proximity π and distance spectrum $\lambda_1 \geq ... \geq \lambda_n$. Then $\pi + \lambda_{\lfloor \frac{2D}{3} \rfloor} \geq 0$.*

Results from Wagner paper helped us to know what kind of graph generate: a tree with more than 203 vertices. The intuition of that counter-example can also be seen in [2].

The NRPA algorithm, with the standard score function, finds trees of size 30 with similar scores (-0.4) as the deep cross entropy method [26] in 3s, when it takes days of training to the deep cross entropy to achieve these scores.

However none of the Monte Carlo method found a counter example with these information only, just like Wagner's deep cross entropy. We determined that the problem stemmed from the score function which lumps scores from different graphs close together, a graph can see its diameter increase but not see repercussions on its score due to the flooring (increasing the diameter of the graph is indeed the way to find a counter example). To solve this problem, we designed an evaluation function slightly different from the score function:

Score function: $-\pi - \lambda_{\lfloor \frac{2D}{3} \rfloor}$

Evaluation function: $-\pi - \lambda'_{2D}$

We define λ' by linearly interpolating λ to be 3 times as long.

With this new evaluation function, increasing the diameter of the graph always leads to a very different score. Instead of playouts, we then used that evaluation function in the greedy-BFS algorithm which found a counter example (figure 3) in about 5 minutes.

The graph shown in figure 3 had a score of approximately 0.000285, meaning $\pi + \lambda_{\lfloor \frac{2D}{3} \rfloor} < 0$, it's indeed a counter-example (it was verified to not be a floating point error with wolfram symbolic eigenvalues computing).

5.3 Conjecture 3. Collins

Conjecture 10 from [9], also present as conjecture 2.4 from [26].

Given a tree T on n vertices, its adjacency matrix $A(T)$ and its distance matrix $D(T)$.

$CPD(T)$ is the characteristic polynomial of $D(T)$:

$$CPD(T) = det(D(T) - Ix) = \sum_{k=0}^{n} \delta_k x^k$$

Let the coefficients $d_k = \frac{2^k}{2^{n-2}} |\delta_k|$ for k in 0, ..., n-2 be the normalized coefficient of the characteristic polynomial.

Let $p_D(T)$ be the peak of the normalized coefficient of the characteristic polynomial.

$CPA(T)$ is the characteristic polynomial of $A(T)$:

$$CPA(T) = det(A(T) - Ix) = \sum_{k=0}^{n} a_k x^k$$

Let $p_A(T)$ be the peak of the non-zero coefficients (a_k) of $CPA(T)$.

Conjecture 3. *Given a tree T, $CPA(T)$ form an unimodal sequence and its peak $p_A(T)$ is at the same place as $p_D(T)$.*

Like Adam Zsolt Wagner [26], we only refuted the second part of the conjecture with a level 2 NMCS, using the same score function.

Score function : $|\frac{p_A(T)}{\#\lambda_{\neq 0}} - (1 - \frac{p_D(T)}{n-2})|$

The counter-example in figure 4 was found in less than a second, it features a $p_A(T)$ at 19/32 and a $p_D(T)$ at 16/30 which gives no doubt that the peaks are at different positions.

5.4 Conjecture 4. Graffiti 137

That conjecture was made automatically by a program named Graffiti [10], it is present in the survey [1] as an already refuted conjecture.

Let $Hc = \sum_{uv \in E} \frac{1}{d(u)+d(v)}$ the harmonic of graph G with vertices E, $d(u)$ is the degree of the vertex u.

Conjecture 4. *For any graph G, the second biggest adjacency matrix eigenvalue λ_2 is inferior to the harmonic of the graph : $\lambda_2 < Hc(G)$.*

Score function : $\lambda_2 - Hc(G)$

A counter-example of size 7 (figure 5) is quickly found by NMCS of level 2, NRPA of level 2 or even Greedy-BFS with $\lambda_2 \approx 1.786$ and $Hc(G) \approx 1.576$.

5.5 Comparison of Search Algorithms

Table 1 gives the times required to refute each conjecture for each algorithm. The machine used has a i5-6600K 3.5 GHz CPU. We see that NMCS can refute conjectures 3 and 4 almost instantly but does not find refutations to conjectures 1 and 2. NRPA also refutes conjectures 3 and 5 almost instantly and refutes conjecture 1 in 0.1 seconds when Wagner solves it in a few hours. Greedy-BFS refutes conjectures 2 and 4. It is the only algorithm (among those we tried) able to refute conjecture 2, using a modified score function to guide a non MC search. It solves it in 5 minutes when Wagner algorithm solves it in a few days.

Conjecture 2 could not be refuted with the natural score function derived from the conjecture but could be solved with Greedy-BFS using a more informative score function. NRPA of level 2 with the natural score function could attain a score of -0.4 in 3s instead of a few days in Wagner's work [26].

6 Conclusion

Monte Carlo search methods proved to be powerful ways of refuting conjecture from spectral graph theory much faster than Wagner's deep cross entropy method [26]. Trying to build trees even when the conjecture is applied on any graph can also be helpful as it reduces the amount of possible builds greatly, it is inexpensive and should be tried first.

Table 1. Summary of our results

Method	Conj 1	Conj 2*	Conj 3	Conj 4
NMCS	-	-	1s (lv 2)	0s (lv 2)
NRPA	1s (lv 3)	-	1s (lv 2)	0s (lv 2)
Greedy-BFS	-	291s	-	0s

(-) denotes a failure, an inability to refute the conjecture.
* Refuted using the evaluation function, different from the score function, that evaluates non terminal states.

However, these methods present limits. Computing score functions that require eigenvalues on big trees (over size 500) can be very costly. They are also dependent on the shape of the score function: a noisy score function with many local minimum can be challenging, as well as a score function with more discrete results can lead to an absence of differentiation in the paths to explore (see conjecture 2). Conjectures requiring to compute a NP hard problem can also severely increase the computing time even for small graphs (30 vertices).

In the future we aim to refute more conjectures and to improve the Greedy-BFS with MCS method, potentially with pruning strategies too.

You can access the Rust code needed to refute these four conjectures here : https://github.com/RoucairolMilo/refutation-COCOON2022

Acknowledgment

Thanks to Édouard Bonnet for fruitful discussions. This work was supported in part by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute).

References

1. Aouchiche, M., Hansen, P.: A survey of automated conjectures in spectral graph theory. Linear Algebra and its Applications **432**(9), 2293–2322 (Apr 2010). https://doi.org/10.1016/j.laa.2009.06.015, https://www.sciencedirect.com/science/article/pii/S0024379509003061
2. Aouchiche, M., Hansen, P.: Proximity, remoteness and distance eigenvalues of a graph. Discrete Applied Mathematics **213**, 17–25 (Nov 2016). https://doi.org/10.1016/j.dam.2016.04.031, https://www.sciencedirect.com/science/article/pii/S0166218X16302037

3. Bouzy, B.: Monte-carlo fork search for cooperative path-finding. In: Computer Games Workshop at IJCAI. pp. 1–15 (2013)
4. Bouzy, B.: Burnt pancake problem: New lower bounds on the diameter and new experimental optimality ratios. In: SOCS. pp. 119–120 (2016)
5. Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of Monte Carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games **4**(1), 1–43 (Mar 2012). https://doi.org/10.1109/TCIAIG.2012.2186810
6. Cazenave, T.: Nested Monte-Carlo Search. In: Boutilier, C. (ed.) IJCAI. pp. 456–461 (2009)
7. Cazenave, T., Saffidine, A., Schofield, M.J., Thielscher, M.: Nested monte carlo search for two-player games. In: AAAI. pp. 687–693 (2016)
8. Cazenave, T., Teytaud, F.: Application of the nested rollout policy adaptation algorithm to the traveling salesman problem with time windows. In: Learning and Intelligent Optimization - 6th International Conference, LION 6. pp. 42–54 (2012)
9. Collins, K.L.: On a conjecture of Graham and Lovász about distance matrices. Discrete Applied Mathematics **25**(1), 27–35 (Oct 1989). https://doi.org/10.1016/0166-218X(89)90044-9, https://www.sciencedirect.com/science/article/pii/0166218X89900449
10. Delavina, E.: Some History of the Development of Graffiti. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science 69: Graphs and Discovery. pp. 81–118
11. Doran, J.E., Michie, D.: Experiments with the graph traverser program. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences **294**(1437), 235–259 (1966)
12. Edelkamp, S., Gath, M., Cazenave, T., Teytaud, F.: Algorithm and knowledge engineering for the tsptw problem. In: Computational Intelligence in Scheduling (SCIS), 2013 IEEE Symposium on. pp. 44–51. IEEE (2013)
13. Edelkamp, S., Gath, M., Greulich, C., Humann, M., Herzog, O., Lawo, M.: Monte-Carlo tree search for logistics. In: Commercial Transport, pp. 427–440. Springer International Publishing (2016)
14. Edelkamp, S., Gath, M., Rohde, M.: Monte-Carlo tree search for 3d packing with object orientation. In: KI 2014: Advances in Artificial Intelligence, pp. 285–296. Springer International Publishing (2014)
15. Edelkamp, S., Greulich, C.: Solving physical traveling salesman problems with policy adaptation. In: Computational Intelligence and Games (CIG), 2014 IEEE Conference on. pp. 1–8. IEEE (2014)
16. Edelkamp, S., Tang, Z.: Monte-Carlo tree search for the multiple sequence alignment problem. In: Proceedings of the Eighth Annual Symposium on Combinatorial Search, SOCS 2015. pp. 9–17. AAAI Press (2015)
17. Finnsson, H., Björnsson, Y.: Simulation-based approach to general game playing. In: Aaai. vol. 8, pp. 259–264 (2008)
18. Hansen, P., Caporossi, G.: AutoGraphiX: An Automated System for Finding Conjectures in Graph Theory. Electronic Notes in Discrete Mathematics **5**, 158–161 (Jul 2000). https://doi.org/10.1016/S1571-0653(05)80151-9, https://www.sciencedirect.com/science/article/pii/S1571065305801519
19. Holcomb, S.D., Porter, W.K., Ault, S.V., Mao, G., Wang, J.: Overview on DeepMind and Its AlphaGo Zero AI. In: Proceedings of the 2018 International Conference on Big Data and Education. pp. 67–71. ACM, Honolulu HI USA (Mar 2018). https://doi.org/10.1145/3206157.3206174, https://dl.acm.org/doi/10.1145/3206157.3206174

20. Méhat, J., Cazenave, T.: Combining UCT and Nested Monte Carlo Search for single-player general game playing. IEEE Transactions on Computational Intelligence and AI in Games **2**(4), 271–277 (2010)
21. Portela, F.: An unexpectedly effective Monte Carlo technique for the RNA inverse folding problem. BioRxiv p. 345587 (2018)
22. Poulding, S.M., Feldt, R.: Generating structured test data with specific properties using nested Monte-Carlo search. In: GECCO. pp. 1279–1286 (2014)
23. Poulding, S.M., Feldt, R.: Heuristic model checking using a Monte-Carlo tree search algorithm. In: GECCO. pp. 1359–1366 (2015)
24. Rimmel, A., Teytaud, F., Cazenave, T.: Optimization of the Nested Monte-Carlo algorithm on the traveling salesman problem with time windows. In: EvoApplications. LNCS, vol. 6625, pp. 501–510. Springer (2011)
25. Rosin, C.D.: Nested rollout policy adaptation for Monte Carlo Tree Search. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 649–654 (2011)
26. Wagner, A.Z.: Constructions in combinatorics via neural networks. arXiv:2104.14516 [cs, math] (Apr 2021), http://arxiv.org/abs/2104.14516, arXiv: 2104.14516
27. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3), 279–292 (May 1992). https://doi.org/10.1007/BF00992698, https://doi.org/10.1007/BF00992698

A Appendix - Algorithms

Algorithm 1 The NMCS algorithm.

```
NMCS ($current\text{-}state$, $level$)
if $level = 0$ then
    $ply \leftarrow 0$
    $seq \leftarrow \{\}$
    while $current\text{-}state$ is not terminal do
        $move \leftarrow randomChoice(M_{current\text{-}state})$
        $current\text{-}state \leftarrow play(current\text{-}state, move)$
        $seq[ply] \leftarrow move$
        $ply+ = 1$
    end while
    return score($current\text{-}state$), $seq$
else
    $best\text{-}score \leftarrow -\infty$
    while $current\text{-}state$ is not terminal do
        for each $move$ in $M_{current\text{-}state}$ do
            $next\text{-}state \leftarrow play(current\text{-}state, move)$
            $(score, seq) \leftarrow$NMCS $(next\text{-}state, level - 1)$
            if $score \geq best\text{-}score$ then
                $next\text{-}best\text{-}state \leftarrow next\text{-}state$
                $best\text{-}score \leftarrow score$
                $best\text{-}sequence \leftarrow seq$
            end if
        end for
        $current\text{-}state \leftarrow next\text{-}best\text{-}state$
    end while
    return $(best\text{-}score, best\text{-}sequence)$
end if
```

Algorithm 2 The NRPA algorithm.

```
NRPA ($policy$, $level$)
if $level = 0$ then
    $current\text{-}state \leftarrow root()$
    $ply \leftarrow 0$
    $seq \leftarrow \{\}$
    while $current\text{-}state$ is not terminal do
        $move \leftarrow softmaxChoice(M_{current\text{-}state}, policy)$
        $current\text{-}state \leftarrow play(current\text{-}state, move)$
        $seq[ply] \leftarrow move$
        $ply+ = 1$
    end while
    return score ($current\text{-}state, seq$)
else
    $best\text{-}score \leftarrow -\infty$
    for $N$ $iterations$ do
        $(result, new) \leftarrow$ NRPA($policy$, $level - 1$)
        if $result \geq best\text{-}score$ then
            $best\text{-}score \leftarrow result$
            $seq \leftarrow new$
        end if
        $pol \leftarrow adapt(pol, seq)$
    end for
    return ($best\text{-}score, seq$)
end if

Adapt ($policy$, $seq$)
$node \leftarrow root()$
$pol' \leftarrow pol$
for $ply = 0$ TO $length(seq) - 1$ do
    $pol'[(node, seq[ply])]$ += $Alpha$
    $z \leftarrow Sum([exp(pol[(node, m)])$ for $m$ in $M_{node}])$
    for each $move$ in $M_{node}$ do
        $pol'[(node, move)]$ -= $\frac{Alpha \cdot exp(pol[(node,move)])}{z}$
    end for
    $node \leftarrow play(node, seq[ply])$
end for
return $pol'$
```

B Appendix - Graphs

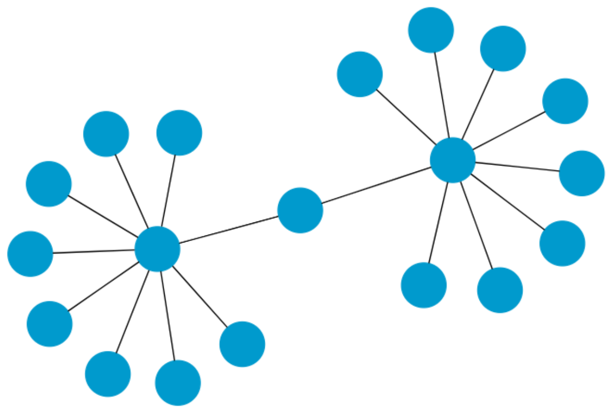

Fig. 1. Counter example of conjecture 1 of size 19

Edges: 0-1, 0-2, 0-3, 0-4, 0-5, 0-6, 0-9, 0-11, 0-18, 4-7, 7-8, 7-10, 7-12, 7-13, 7-14, 7-15, 7-16, 7-17

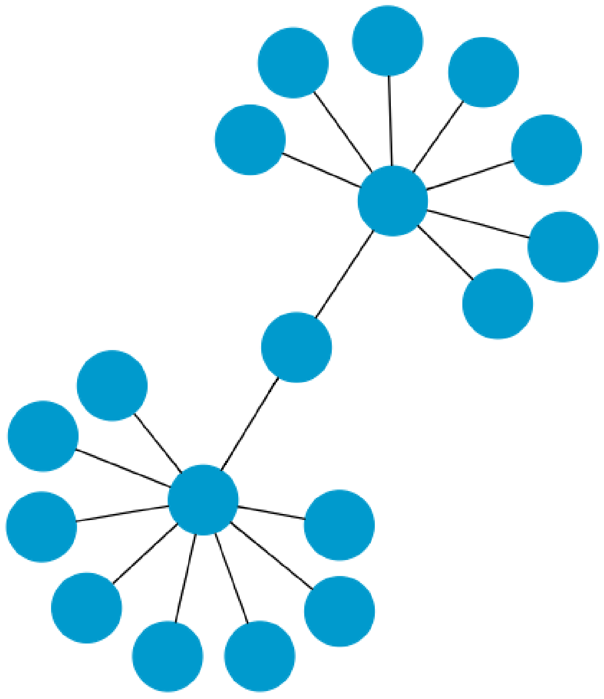

Fig. 2. Counter example of conjecture 1 of size 18

Edges: 0-1, 0-2, 0-3, 0-4, 0-5, 0-6, 0-9, 0-11, 4-7, 7-8, 7-10, 7-12, 7-13, 7-14, 7-15, 7-16, 7-17

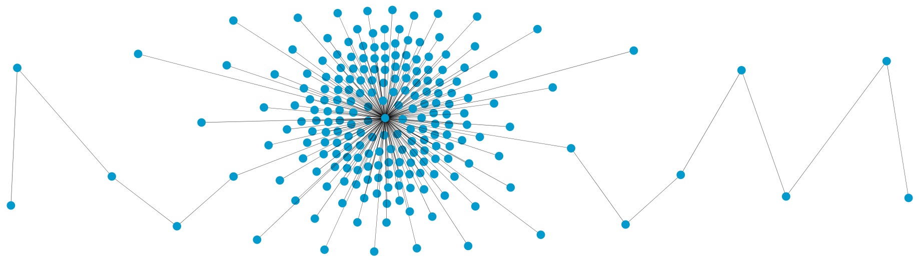

Fig. 3. Counter example of conjecture 2

Edges: 0-1, 0-8, 0-13, 0-14, 0-15, 0-16, 0-17, 0-18, 0-19, 0-20, 0-21, 0-22, 0-23, 0-24, 0-25, 0-26, 0-27, 0-28, 0-29, 0-30, 0-31, 0-32, 0-33, 0-34, 0-35, 0-36, 0-37, 0-38, 0-39, 0-40, 0-41, 0-42, 0-43, 0-44, 0-45, 0-46, 0-47, 0-48, 0-49, 0-50, 0-51, 0-52, 0-53, 0-54, 0-55, 0-56, 0-57, 0-58, 0-59, 0-60, 0-61, 0-62, 0-63, 0-64, 0-65, 0-66, 0-67, 0-68, 0-69, 0-70, 0-71, 0-72, 0-73, 0-74, 0-75, 0-76, 0-77, 0-78, 0-79, 0-80, 0-81, 0-82, 0-83, 0-84, 0-85, 0-86, 0-87, 0-88, 0-89, 0-90, 0-91, 0-92, 0-93, 0-94, 0-95, 0-96, 0-97, 0-98, 0-99, 0-100, 0-101, 0-102, 0-103, 0-104, 0-105, 0-106, 0-107, 0-108, 0-109, 0-110, 0-111, 0-112, 0-113, 0-114, 0-115, 0-116, 0-117, 0-118, 0-119, 0-120, 0-121, 0-122, 0-123, 0-124, 0-125, 0-126, 0-127, 0-128, 0-129, 0-130, 0-131, 0-132, 0-133, 0-134, 0-135, 0-136, 0-137, 0-138, 0-139, 0-140, 0-141, 0-142, 0-143, 0-144, 0-145, 0-146, 0-147, 0-148, 0-149, 0-150, 0-151, 0-152, 0-153, 0-154, 0-155, 0-156, 0-157, 0-158, 0-159, 0-160, 0-161, 0-162, 0-163, 0-164, 0-165, 0-166, 0-167, 0-168, 0-169, 0-170, 0-171, 0-172, 0-173, 0-174, 0-175, 0-176, 0-177, 0-178, 0-179, 0-180, 0-181, 0-182, 0-183, 0-184, 0-185, 0-186, 0-187, 0-188, 0-189, 0-190, 0-191, 0-192, 0-193, 0-194, 0-195, 0-196, 0-197, 0-198, 0-199, 0-200, 0-201, 0-202, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 8-9, 9-10, 10-11, 11-12

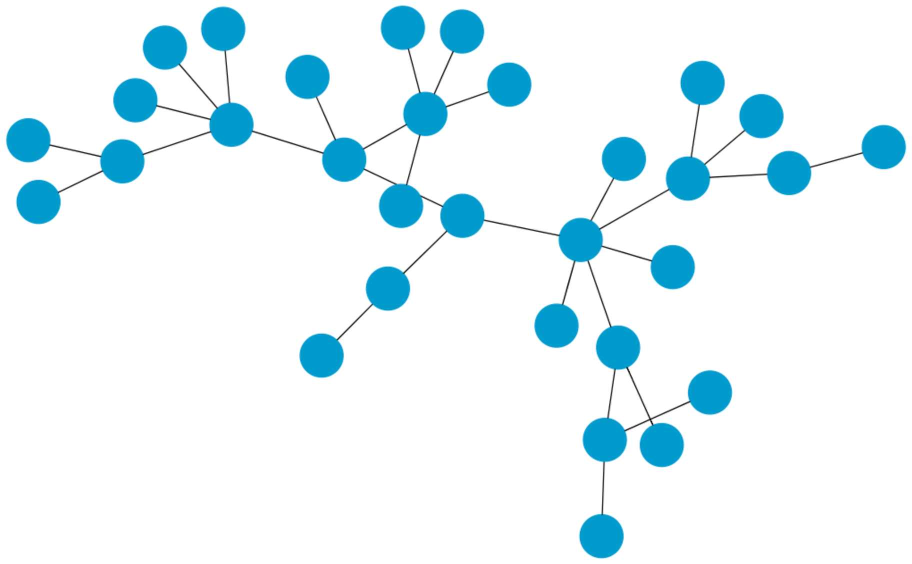

Fig. 4. Counter example of conjecture 3

Edges: 0-1, 0-2, 0-13, 0-27, 1-5, 1-7, 1-9, 1-22, 2-3, 3-4, 3-8, 3-12, 4-10, 4-25, 5-6, 5-16, 5-21, 6-14, 6-28, 9-11, 10-20, 11-24, 13-19, 14-15, 15-17, 15-18, 15-23, 19-26, 19-30, 28-29

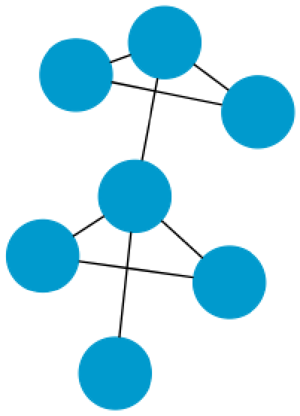

Fig. 5. Counter example of conjecture 4

Edges: 0-1, 0-2, 1-2, 0-3, 3-4, 4-5, 3-5, 3-7