ORIGINAL RESEARCH — WORD COUNT : 3506

Automated Refutation with Monte Carlo Search of Graph Theory Conjectures on the Maximum Laplacian Eigenvalue

Liora Taieb[a], Tristan Cazenave[a], Milo Roucairol[a] and Ararat Harutyunyan[a]

[a]LAMSADE, Université Paris Dauphine - PSL

ARTICLE HISTORY
Compiled October 21, 2024

ABSTRACT
We address the problem of automatic refutation of spectral graph theory conjectures with Monte Carlo methods. Usual ways are testing conjectures on an exhaustive database of graphs below a certain size, local search algorithms, or, more recently, deep reinforcement learning. We expand on previous works by finding smaller (and often sparser) counter-examples to spectral graph theory conjectures in seconds when it takes minutes or hours with other methods. We apply search algorithms (including state-of-the-art Monte Carlo Searches) to 68 automated conjectures already addressed by the deep cross-entropy method. In addition to the ones already disproved by deep cross-entropy, we refute 2 open conjectures until now. We highlight the efficiency of Monte Carlo Search algorithms compared to a state-of-the-art neural approach, and the advantages of the constructive method. Monte Carlo search can be used to automatically refute conjectures that are experimentally generated.

KEYWORDS
Monte Carlo Search; Spectral Graph Theory; Conjecture; Refutation

1. Introduction

Finding potential counterexamples to a graph theory conjecture can be a tiresome task. The search space of graphs of order n consists of at least of $\frac{2^{\binom{n}{2}}}{n!}$ non isomorphic graphs, with $n!$ being the maximal cardinality of a graph isomorphism class. Thus the search space's size is expanded in doubly exponential fashion when exploring multiple graph sizes when there is no prior intuition on the likely size of a counter-example.

This article expands the work of Roucairol and Cazenave (Roucairol & Cazenave, 2022, 2024), who were the first to use Monte Carlo Search methods to build counter-examples of spectral graph theory conjectures, in which they showed the effectiveness of the approach against neural state-of-the-art methods. Few works have been devoted to evaluating Monte Carlo search algorithms for combinatorial graph problems. Cazenave et al. applied it to graph coloring (Cazenave, Negrevergne, & Sikora, 2021), investigated before them only by Edelkamp et al. (Edelkamp, Externest, Kühl, & Kuske, 2017), both works being competitive with state-of-the-art SAT solvers.

Our focus in this paper is once again on spectral graph theory conjectures, as they are well-suited for automated refutation. Here, we focus on conjectures involving the maximum Laplacian eigenvalue of a graph. The property of a spectral theory conjecture

Corresponding Author. Email: tristan.cazenave@lamsade.dauphine.eu

can straightforwardly be translated into an evaluation function whose set of images is large and continuous, which usually provides a good assessment of the quality of the states evaluated. Many available softwares and libraries of various computer languages carry out eigenvalue calculations quickly.

The paper is organized as follows. In Sect. 2, we describe previous work in the refutation of graph theory conjectures and we focus on Monte Carlo search. In Sect. 3, we present the problem and the methodology used to explore the problem space. Finally, in Sect. 4 and 5, we present and discuss our results on multiple conjectures.

2. Refutation of Graph Theory Conjectures

2.1. State Of The Art

Graph conjectures are propositions on graph classes (any graph, trees, K-free...) that are thought to be true and are awaiting proof or a refutation. Many mathematicians have put forward spectral graph theory conjectures still open to this day (Liu & Ning, 2023). Automated softwares for creating conjectures emerged at the rise of powerful computers in the 80s, including Ingrid (Dutton, Brigham, & Gomez, 1989), GRAPH (Cvetković & Simić, 1994), Graffiti (DeLaVina, 2005) and AutoGraphiX (P. Hansen & Caporossi, 2000). Thanks to them, plenty of such conjectures are available and still open (Aouchiche & Hansen, 2010).

Graffiti uses known theorems to create and refine many conjectures in the form of inequalities between graph invariants. It then tests these conjectures on a database of graphs and discards the falsified ones. Its database is built by an exhaustive generation of graphs smaller than a threshold size. The system later checks if the inequalities are not implied by already known theorems and conjectures, including those it has created itself. If a conjecture passes these tests, it is proposed to graph theorists. "Written on the wall" (*Latest version of "Written on the wall"*, 2012) collects almost 1000 conjectures from Graffiti, along with the discussions of many renowned graph theorists. The efficiency of Graffiti's refutation process is strongly limited by the quality of the database, in other words its completeness and the maximal size of generated graphs.

AutoGraphiX uses local search to create - and also refute - some conjectures. The program uses a heuristic, the Variable Neighborhood Search (VNS) to identify extremal graphs and suggest conjectures based on their structure. Local search is naturally more intelligent than a naive exhaustive generation as it introduces constraints to reduce the search space, and has been used several times to solve combinatorial graph problems (Hertz & de Werra, 1987; Lidický, McKinley, & Pfender, 2024; Mehrabian et al., 2023).

Lately, machine and deep learning tackled combinatorial problems over graphs (Khalil, Dai, Zhang, Dilkina, & Song, 2017). Wagner's deep reinforcement learning technique (Wagner, 2021) is a pioneering refutation method of graph theory conjectures. It has been reworked in (Angileri et al., 2024) and used for research in Turán theory in (Mehrabian et al., 2023), Ramsey theory in (Ghebleh, Al-Yakoob, Kanso, & Stevanović, 2024) and spectral graph theory in (Al-Yakoob, Ghebleh, Kanso, & Stevanovic, 2024), disproving open conjectures in (Al-Yakoob et al., 2024; Ghebleh et al., 2024; Wagner, 2021). This program learns a policy of graph generation using a deep cross entropy. It trains a neural network to output a policy on the next edge to add to a graph. This way, it creates a batch of full graphs from which the network learns by minimizing the cross-entropy with the distribution of the best graphs of the batch. Wagner's method intrinsically understands the structure of the best graphs. But it

can take hours or days, and it is limited to the study of fixed sizes of graphs.

2.2. Monte Carlo Search

Monte Carlo Tree Search (MCTS) is a family of algorithms that combine tree search and reinforcement learning using playouts and heuristics (Browne et al., 2012). The search space (and not the search states) is represented by a tree, where the nodes are the partial constructions, the edges are the next possible moves and the leaves are terminal states. It performs numerous simulations and stores the statistics of actions and their resulting evaluations to make more educated choices in each iteration. Constraints on legal moves delimit the possible extent of the search space.

Monte Carlo search algorithms were proven to be powerful in puzzles and optimization problems, with recent successes like AlphaGo (Silver et al., 2016). They often excel in games (Cazenave, 2009; Méhat & Cazenave, 2010) and non-games applications including biology (Edelkamp & Tang, 2015; Portela, 2018), logistics (Edelkamp et al., 2016; Edelkamp & Greulich, 2014) and many more (Browne et al., 2012). These algorithms have the advantage of only needing an evaluation function for the final state of the space they explore.

The efficiency of MCTS comes from its incremental construction of graphs, which can produce large solutions. It is similar to local search in that it improves the construction by moving towards the best neighboring solutions. As a local search and reinforcement learning method, MCTS can fall into a local optima if the evaluation score is too noisy, but decisions based on reinforcement learning help balance this limitation. Some of them such as Nested Rollout Policy Adaptation learn a general policy that serves as a playout guiding heuristic, in line with the principle of the deep cross entropy, without being limited by a fixed graph size.

MCTS is efficient in solving optimization problems where the score function and the legal moves are relatively inexpensive to compute, as these calculations will be made many times during the simulations. The difficulty with these methods lies in the choice of an efficient exploration of the search space, and the formalization of an objective yet feasible evaluation function. For NP-hard problems, it is possible to bias the search intelligently (Cazenave, 2017; Cazenave et al., 2021).

3. Problem and Methodology

The conjectures we study come from (Brankov, Hansen, & Stevanović, 2006) and involve the largest eigenvalue μ of the Laplacian matrix of graphs (degree matrix minus adjacency matrix). They were created automatically similarly to Graffiti with a database of 273 214 connected graphs with up to 9 vertices, enhanced by a few special graphs. We focus on the 68 conjectures attempted by Al-Yakoob et al. with the deep cross entropy method (Al-Yakoob et al., 2024). All conjectures are presented in the appendix, their order comes from (Al-Yakoob et al., 2024), different from the order (Brankov et al., 2006). They are of the form

$$\mu \leqslant \max_{v_i} f\left(d_i, m_i\right) \quad (1) \qquad \text{or} \qquad \mu \leqslant \max_{v_i \sim v_j} f\left(d_i, m_i, d_j, m_j\right) \quad (2)$$

where v_i is any vertex of G, d_i is the degree of this vertex and m_i is the average of the degrees of v_i's neighbors, and f is some function involving these parameters.

The evaluation function naturally is μ - the value on the right of the inequality.

To avoid floating point errors when calculating eigenvalues, we force the result to be greater than 0.0001 for the graph to be considered a counter-example (it is more than enough for graphs of size 20). The conjectures apply to any graph, therefore we begin with the naive approach of not restraining the legal moves. Following the work of Roucairol and Cazenave (Roucairol & Cazenave, 2022), we then narrow the legal moves to create only trees, as the search space is way smaller and trees are more likely to converge toward extreme graphs with extreme properties.

We apply 3 Monte Carlo search algorithms to the problem as well as the Greedy Best-First Search algorithm, the evolutionary algorithm Covariance Matrix Adaptation - Evolutionary Strategy and an Iterated Local Search. Their pseudo-codes are presented in the appendix.

- Nested Monte Carlo Search (NMCS) (Cazenave, 2009) uses nested levels of playouts with random playouts at the base level. At each recursion level, each legal moves is assigned a score from the results of the lower level NMCS starting from the move, and the best move is selected this way.
- Nested Rollout Policy Adaptation (NRPA) (Rosin, 2011) is similar to a NMCS, but learns a policy with nested levels of best sequences. At the lowest level, it becomes a playout with the learned policy.
- Generalized Rapid Action Value Estimation (GRAVE) (Cazenave, 2015) uses the All Moves As First (AMAF) heuristic to update move statistics, taking into account all the moves that were played in the playout and not only the first one. It incorporates statistics from a higher reference state in the tree, which is the closest ancestor state that has more playouts than a given constant.
- Greedy Best-First Search (GBFS) is a simple and deterministic greedy algorithm opening the best state from a list, evaluating the children of this state, and inserting these children back in the list according to their evaluation.
- Covariance Matrix Adaptation - Evolutionary Strategy (CMA-ES) (N. Hansen, 2016) is an evolutionnary method that samples children from the multivariate Gaussian distribution of the best parents, with the aim of producing even better children. By definition, it understands the structures of best graphs to which it applies mutations.
- Iterated Local Search (ILS) (Lourenço, Martin, & Stützle, 2019) repeatedly applies local search on perturbed solutions.

4. Results

The data and code that support the findings of this study are openly available at:
https://github.com/liorataieb/RefutationSpectralConjectures.

The experiments were made with Rust 2024.1, on an Intel Core i7-1365U 5.2GHz using a single core. Each algorithm has been allocated a maximum of 1 minute per conjecture and each algorithm has been ran several times. The then unrefuted conjectures have been alloted additional time, between 15 minutes and 1 hour. The terminal state of the algorithms and playouts occurs when the graph reaches 20 vertices, as it is the fixed size studied with the deep cross entropy. We also tried various terminal sizes between 15 and 50. We present in Table 1 the results of the algorithms, only for the refuted conjectures for the sake of clarity.

- NCMS, NRPA and GRAVE were ran with and without heuristic on the choice of the next move in playouts.

- Only NMCS, NRPA, GRAVE and GBFS were restricted to trees.
- NMCS and NRPA used at most a level of 3.
- GRAVE used a reference of 50.
- CMA-ES used a λ of 5, 10 or 15.
- Several variations of ILS were tried, inspired from the work in (Lourenço et al., 2019).
- CMA-ES and ILS were tried on solutions of size 20.

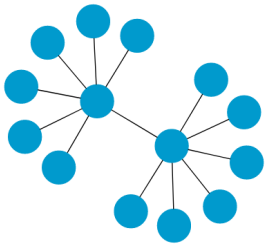
The deep cross entropy in (Al-Yakoob et al., 2024) disproved 25 conjectures mostly with graphs of size 20, up to size 24. 5 more conjectures were refuted in the article using an exhaustive research on subquartic graphs (graph with degree at most 4) of size 14 at most. As shown in Table 1, our algorithms combined were able to refute 29 conjectures : the 25 refuted by the deep cross entropy, 2 that were still open (Conjectures 45 and 48, see Figure 1a and Figure 1b) and 2 (out of 5) that were refuted by a subquartic graph.

By aggregating the results of 1-minute runs, NMCS outperforms the deep cross entropy and refutes 28 conjectures, NRPA refutes 26, GRAVE refutes 26 and GBFS refutes 16. ILS and CMA-ES refute no conjectures. The best number of disproved conjectures obtained without restart, leaving only 1 minute by conjecture, is 23 conjectures for NMCS, NRPA and GRAVE, achieved when we restrict the search space to trees and use a level of 3 for NMCS and NRPA. with the same constraints applied to exploration of the entire search space, NMCS and GRAVE produced a maximum of 17 refuted conjectures, NRPA produced at most 4.

Parallelized restarts of MCTS methods are strongly advised given the difference between the total number of conjectures refuted and the one of the best run of each algorithm. Our experiments show that applying a heuristic to moves during the playouts systematically increases the number of refuted conjectures compared with non-heuristic applications of the algorithms, although the refuted conjectures are not always the same. The same goes when searching for trees rather than any graphs. It is truly a cost-effective variation that should be tried first.

Increasing the level of NRPA and NMCS yields more refutation, but not every conjectures can be refuted with a higher level. Conjecture 28 was refuted by NRPA only with a level of 2 or lower. Likewise, Conjecture 48 can be disproved by NMCS only with a level of 2 or lower, because NMCS with a level of 3 starts evaluating graphs at size 4. All other conjectures have been refuted with a level of 3. We do not advise to go higher than a level of 3, as the trade-of between detailed exploration and computational power requirements is profitable. As for the terminal size of graphs, trying different ones is good practice; most of the conjectures were refuted with a size of 20, but not all, as Conjecture 62 with GRAVE was refuted only with a size of 50 when restricting the search space to trees.

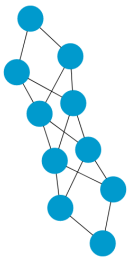
It is interesting to note that GRAVE is the only algorithm that found a counter-example for Conjecture 51. NRPA favors patterns and NMCS succeeds more where the solutions are more chaotic. GRAVE can be described as a mix of the two (not in terms of algorithm, but policy learning behavior), and it is illustrated by the structure of the counter-example found (see Figure 2b). Furthermore, NMCS yields smaller counter-examples than NRPA, itself producing smaller results than GRAVE. NRPA and GRAVE execute a repetitive strategy usually leading to larger graphs. Given that the counter-examples that are not trees found by NMCS and GRAVE have very little structure, 1 minute may not be enough for NRPA to produce solutions, explaining the discrepancy in results between the algorithms when they are not limited to the search



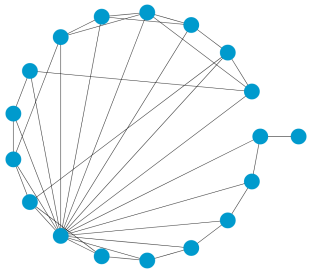
(a) Edges : 0-1, 0-4, 0-5, 0-8, 0-10, 0-11, 0-12, 1-2, 1-3, 1-6, 1-7, 1-9, 1-13

(b) Edges : 0-1, 0-2

Figure 1. A counter-example of Conjecture 45 (left) and Conjectures 48, 57 and 61 (right)



(a) Edges : 0-1, 0-9, 1-2, 1-8, 2-3, 2-7, 2-9, 3-4, 3-6, 3-8, 4-5, 4-7, 5-6, 6-7, 7-8, 8-9

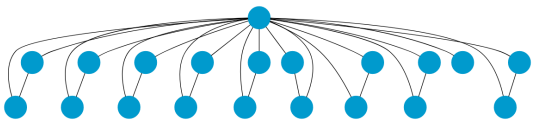


(b) Edges : 0-1, 0-2, 0-3, 0-4, 0-5, 0-6, 0-7, 0-8, 0-9, 0-10, 0-11, 0-12, 0-14, 0-15, 0-16, 0-17, 1-2, 1-9, 1-17, 2-3, 2-5, 3-4, 4-10, 5-6, 5-7, 6-7, 6-8, 7-8, 7-10, 8-9, 9-10, 11-12, 11-13, 12-14, 14-15, 15-16, 16-17

Figure 2. A counter-example of Conjecture 50 (left) and Conjecture 51 (right)

for trees.

Among the 11 conjectures disproved by MCTS but not GBFS, the latter is only able to refute 1 more conjecture with a 5-minute run, and 3 more with 15 minutes of allotted time. Conjecture 50 was refuted only by NMCS during a 1-hour run with a terminal size of 15, raising the total number of conjectures disproved by the algorithm to 28. As shown in Figure 2a, the counter-example is very structured but small, perhaps too small for NRPA and GRAVE to find it. We tried several 1-hour runs of NMCS with heuristic, NRPA and GBFS, both for any graphs and for trees, on the 3 remaining conjectures already refuted by subquartic graphs, without results. Moreover, NMCS, NRPA, GRAVE, and GBFS were not able to refute unproved conjectures with a 15-minute alloted time on terminal size 15, 20 and 50. However the bounds of those conjectures are often reached during exploration.



(a) Edges : 0-1, 0-2, 0-3, 0-4, 0-5, 0-6, 0-7, 0-8, 0-9, 0-10, 0-11, 0-12, 0-13, 0-14, 0-15, 0-16, 0-17, 0-18, 0-19, 2-3, 4-5, 6-7, 8-9, 10-11, 12-13, 14-15, 16-17, 18-19

Figure 3. A counter-example of Conjecture 65

Table 1. Time in seconds to obtain a refutation with each algorithm (maximum of 1 minute per conjecture).

	NMCS		NRPA		GRAVE		GBFS		CMA-ES	ILS
Conj.	Time	Graph type	Time	G. type	Time	G. type	Time	G. type		
3	0	tree	50	tree	0	tree	0	any	-	-
15	0	tree	0	tree	7	tree	1	any	-	-
28	0	tree	0	tree	-	-	0	any	-	-
29	0	tree	0	tree	0	tree	-	-	-	-
31	0	tree	1	tree	0	tree	25	any	-	-
36	0	tree	0	tree	0	tree	-	-	-	-
	43	any								
41	3	any	14	tree	7	any	-	-	-	-
43	0	tree	0	tree	0	tree	0	any	-	-
	2	any			1	any				
45	0	tree	3	tree	-	-	-	-	-	-
48	0	tree	0	tree	0	tree	0	tree	-	-
	0	any	0	any	0	any	0	any		
49	0	tree	0	tree	0	tree	0	any	-	-
	7	any			3	any				
50	422[b]	any	-	-	-	-	-	-	-	-
51	-	-	-	-	5	any	-	-	-	-
52	0	tree	0	tree	0	tree	0	any	-	-
	1	any			0	any				
53	0	tree	0	tree	0	tree	0	any	-	-
	1	any			0	any				
54	0	tree	0	tree	0	tree	-	-	-	-
	1	any			0	any				
55	0	tree	0	tree	0	tree	-	-	-	-
	3	any			0	any				
57	0	tree	0	tree	0	tree	0	tree	-	-
	1	any	0	any	0	any	0	any		
58	0	tree	0	tree	0	tree	0	any	-	-
	3	any	43	any	4	any				
59	1	tree	0	tree	0	tree	-	-	-	-
	11	any			19	any				
60	1	tree	2	tree	0	tree	-	-	-	-
	1	any			25	any				
61	0	tree	0	tree	0	tree	0	tree	-	-
	1	any	0	any	0	any	0	any		
62	1	tree	0	tree	0	tree	-	-	-	-
	2	any			4	any				
63	0	tree	0	tree	0	tree	0	any	-	-
	23	any								
64	1	tree	2	tree	0	tree	0	any	-	-
65	0	tree	0	tree	0	tree	-	-	-	-
	1	any			0	any				
66	48	tree	-	-	1	tree	0	any	-	-
67	1	tree	3	tree	0	tree	-	-	-	-
68	0	tree	0	tree	0	tree	0	any	-	-
	1	any	4	any	0	any				

[a]types of graphs we used in our searches, "any" means there was no restriction on the graph built, "tree" means the graph must be a tree. [b]Those results were obtained by NMCS while testing 1-hour runs per conjecture for Conjecture 2, 17, 32 and 50, the 4 conjectures refuted by subquartic graphs but not MCTS.

5. Discussion

All conjectures previously refuted with the deep cross entropy have been disproved very quickly by the Monte Carlo algorithms, especially NMCS and GRAVE. The difference of results between MCTS and GBFS highlights MCTS superior exploration capabilities. This study provides evidence that MCTS discovers potentially desirable structures much sooner than deep neural reinforcement learning and outperforms both the deep cross entropy and more naive local searches. MCTS strength comes from the gradual building of graphs coupled with smart detection of a search branch quality. It gives them an advantage to find counter-examples of any (reasonable) sizes, especially smaller and typically less dense counter-examples. An extreme illustration of this is Figure 1b, the very small path of 3 vertices, that refutes Conjectures 48, 57 and 61, one of which was still open. The simplicity of the counter-example exhibits a serious limitation of the exhaustive generation used to test to conjectures in (Brankov et al., 2006). But it is a very simple graph, directly and naturally found by MCTS algorithms.

The 4 subquartic graphs presented in (Al-Yakoob et al., 2024) that were able to refute Conjectures 2, 17, and 32 are very specific. MCTS is capable of generating specific graphs, such as the one in Figure 3a, which presents an almost windmill[1] graph as a counter-example to Conjecture 36, and the one in Figure 2a, which presents a "shoelace" graph as a counter-example to Conjecture 50. The fact that MCTS failed to produce counter-examples for Conjectures 2, 17, and 32 even after hour-long runs highlights the lack of guarantee in finding an optimal result. The bound of those conjectures are reached by our algorithms, but we believe this limitation is due to the high granularity of the score functions. We tried to remove arbitrarily one or two edges of the subquartic counter-examples in (Al-Yakoob et al., 2024) and spotted significantly lower scores when doing so, so much that no conjecture was refuted anymore. This is a general limitation of optimization methods.

CMA-ES and ILS have not been restricted to build trees, which may explain the poor results we obtained for them. Both methods start with a preconstructed graph. CMA-ES is the closest algorithm to the deep cross entropy, and also the hardest to train. We believe it got lost in the granularity of the score functions. ILS seems too naive to yield results.

6. Conclusion

The Monte Carlo approach performs well in many fields, and their versatility can effectively be applied to conjecture refutation in graph theory as well. We have shown that Monte Carlo search rapidly outperforms its rivals in refuting spectral graph theory conjectures. Its intelligence allows for quick and effortless research among many sizes of graphs. Roucairol and Cazenave (Roucairol & Cazenave, 2022) have shown that Monte Carlo methods can rapidly produce large counter-examples. Here, the methods yield smaller counter-examples than the state-of-the-art deep cross entropy method. It also refutes open conjectures which were not refuted by other methods.

Here, we only used Monte Carlo methods, as the conjectures we studied call for computationally inexpensive calculations, still on a par with what is done today. These methods are adaptable and can very well be combined with diverse heuristics to boost

[1] A windmill is an undirected graph constructed for $k \leqslant 2$ and $n \leqslant 2$ by joining n copies of the complete graph K_k at a shared universal vertex.

results. Monte Carlo search should be investigated further for more complex combinatorial graph problems.

Disclosure statement

The authors have no relevant financial or non-financial interests to disclose.

Funding

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

7. References

References

Al-Yakoob, S., Ghebleh, M., Kanso, A., & Stevanovic, D. (2024). Reinforcement learning for graph theory, i. reimplementation of wagner's approach. *arXiv preprint arXiv:2403.18429*.

Angileri, F., Lombardi, G., Fois, A., Faraone, R., Metta, C., Salvi, M., & Morandin, F. (2024). A systematization of the wagner framework: Graph theory conjectures and reinforcement learning. *arXiv preprint arXiv:2406.12667*.

Aouchiche, M., & Hansen, P. (2010). A survey of automated conjectures in spectral graph theory. *Linear algebra and its applications*, *432*(9), 2293–2322.

Brankov, V., Hansen, P., & Stevanović, D. (2006). Automated conjectures on upper bounds for the largest laplacian eigenvalue of graphs. *Linear algebra and its applications*, *414*(2-3), 407–424.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., & Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 1–43.

Cazenave, T. (2009, June). Nested monte-carlo search. In *Twenty-first international joint conference on artificial intelligence.*

Cazenave, T. (2015). Generalized rapid action value estimation. In *24th international conference on artificial intelligence* (pp. 754–760).

Cazenave, T. (2017). Nested rollout policy adaptation with selective policies. In T. Cazenave, M. H. M. Winands, S. Edelkamp, S. Schiffel, M. Thielscher, & J. Togelius (Eds.), *Cgw/giga-2016* (Vol. 705, pp. 44–56). Springer, Cham. Retrieved from `https://doi.org/10.1007/978-3-319-57969-6`$_4$

Cazenave, T., Negrevergne, B., & Sikora, F. (2021). Monte carlo graph coloring. In *Monte carlo search: First workshop, mcs 2020, held in conjunction with ijcai 2020, virtual event, january 7, 2021, proceedings 1* (pp. 100–115). Springer International Publishing.

Cvetković, D., & Simić, S. (1994). Graph theoretical results obtained by the support of the expert system" graph". *Bulletin (Académie serbe des sciences et des arts. Classe des sciences mathématiques et naturelles. Sciences mathématiques)*, 19–41.

DeLaVina, E. (2005). Some history of the development of graffiti. In *Dimacs series in discrete mathematics and theoretical computer science* (Vol. 69, p. 81).

Dutton, R. D., Brigham, R. C., & Gomez, F. (1989). Ingrid: A graph invariant manipulator. *Journal of symbolic computation*, *7*(2), 163–177.

Edelkamp, S., Externest, E., Kühl, S., & Kuske, S. (2017). Solving graph optimization problems in a framework for monte-carlo search. In *Tenth annual symposium on combinatorial search.*

Edelkamp, S., Gath, M., Greulich, C., Humann, M., Herzog, O., & Lawo, M. (2016). Monte-carlo tree search for logistics. In *Commercial transport: Proceedings of the 2nd interdisciplinary conference on production logistics and traffic 2015* (pp. 427–440). Springer International Publishing.

Edelkamp, S., & Greulich, C. (2014). Solving physical traveling salesman problems with policy adaptation. In *2014 ieee conference on computational intelligence and games* (pp. 1–8).

Edelkamp, S., & Tang, Z. (2015). Monte-carlo tree search for the multiple sequence alignment problem. In *Proceedings of the international symposium on combinatorial search* (Vol. 6, pp. 9–17).

Ghebleh, M., Al-Yakoob, S., Kanso, A., & Stevanović, D. (2024). Reinforcement learning for graph theory, ii. small ramsey numbers. *arXiv preprint arXiv:2403.20055*.

Hansen, N. (2016). The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

Hansen, P., & Caporossi, G. (2000). Autographix: An automated system for finding conjectures in graph theory. *Electronic Notes in Discrete Mathematics*, *5*, 158–161.

Hertz, A., & de Werra, D. (1987). Using tabu search techniques for graph coloring. *Computing*, *39*(4), 345–351.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *Advances in Neural Information Processing Systems*, *30*.

Latest version of "written on the wall". (2012). (Accessed: https://independencenumber.wordpress.com/wp-content/uploads/2012/08/wow-july2004.pdf)

Lidický, B., McKinley, G., & Pfender, F. (2024). Small ramsey numbers for books, wheels, and generalizations. *arXiv preprint arXiv:2407.07285*.

Liu, L., & Ning, B. (2023). Unsolved problems in spectral graph theory. *arXiv preprint arXiv:2305.10290*.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2019). Iterated local search: Framework and applications. In *Handbook of metaheuristics* (pp. 129–168).

Mehrabian, A., Anand, A., Kim, H., Sonnerat, N., Balog, M., Comanici, G., & Wagner, A. Z. (2023). Finding increasingly large extremal graphs with alphazero and tabu search. *arXiv preprint arXiv:2311.03583*.

Méhat, J., & Cazenave, T. (2010). 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.

Portela, F. (2018). An unexpectedly effective monte carlo technique for the rna inverse folding problem. *BioRxiv*, 345587.

Rosin, C. D. (2011, July). Nested rollout policy adaptation for monte carlo tree search. In *Ijcai* (Vol. 2011, pp. 649–654).

Roucairol, M., & Cazenave, T. (2022). Refutation of spectral graph theory conjectures with monte carlo search. In *International computing and combinatorics conference* (pp. 162–176). Springer International Publishing.

Roucairol, M., & Cazenave, T. (2024). *Refutation of spectral graph theory conjectures with search algorithms).* Retrieved from `https://arxiv.org/abs/2409.18626`

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Wagner, A. Z. (2021). Constructions in combinatorics via neural networks. *arXiv preprint arXiv:2104.14516*.

8. Appendices

Appendix A. Algorithms

Algorithm 1 The NMCS algorithm.

```
function NMCS(current-state, level)
    if level = 0 then
        ply ← 0
        seq ← {}
        while current-state is not terminal do
            move ← randomChoice(LegalMoves_current-state)
            current-state ← play(current-state, move)
            seq[ply] ← move
            ply+ = 1
        end while
        return score(current-state), seq
    else
        best-score ← −∞
        best-seq ← []
        ply ← 0
        while current-state is not terminal do
            for each move in M_current-state do
                next-state ← play(current-state, move)
                (score, seq) ←NMCS(next-state, level − 1)
                if score ≥ best-score then
                    best-score ← score
                    best-sequence[ply..] ← move + seq
                end if
            end for
            next-move ← best-sequence[ply]
            ply ← ply + 1
            current-state ← play(current-state, next-move)
        end while
        return (best-score, best-sequence)
    end if
end function
```

Algorithm 2 The NRPA algorithm.

function NRPA(*policy*, *level*)
  if $level = 0$ **then**
    $current\text{-}state \leftarrow root()$
    $ply \leftarrow 0$
    $seq \leftarrow \{\}$
    while *current-state* is not terminal **do**
      $move \leftarrow softmaxChoice(LegalMoves_{current\text{-}state}, policy)$
      $current\text{-}state \leftarrow play(current\text{-}state, move)$
      $seq[ply] \leftarrow move$
      $ply+ = 1$
    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
end function

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

Algorithm 3 The GRAVE algorithm.

Input: N tree-walks, initial state s_0, reference state constant ref
Output: A search tree

Initialize an empty transposition table
for $i = 1$ to N **do**
  $s \leftarrow s_0$, $S \leftarrow \{s\}$, $s_{\text{ref}} \leftarrow s$
  while s is not a leaf state and is not simulatable **do**
    if $n(s) >$ ref **then**
      $s_{\text{ref}} \leftarrow s$
    end if
    for each $a \in s$.children **do**
      $\beta \leftarrow \frac{\text{sref.pAMAF}}{\text{sref.pAMAF}+s.p+\text{bias}\times sref.pAMAF\times s.p}$
      grave $\leftarrow (1-\beta) \times s.\text{mean} + \beta \times sref.\text{AMAF}$
    end for
    Select $a \leftarrow \text{argmax}\{\text{GRAVE}(s,a) \mid a \in s.\text{children}\}$
    Transition to the new state resulting from action a, $S \leftarrow S \cup \{s\}$
  end while
  Sample a new action a from the available moves of s
  Add the state resulting from action a as a child node of s
  while s is not a terminal state **do**
    Sample a from the available moves of s based on the default policy
    Transition to the new state resulting from action a
  end while
  score $\leftarrow$ evaluate(s)
  for each $s \in S$ **do**
    Update s with score
  end for
end for

Algorithm 4 The GBFS algorithm.

```
1: function GBFS(ini-state, max-iter)
2:     open-states ← [ini-state]
3:     state ← ini-state
4:     iter ← 0
5:     best-state ← ini-state
6:     best-score ← score(ini-state)
7:     while not optimal(state) and open-states ≠ [] and iter < max-iter do
8:         iter ← iter + 1
9:         state ← pop( open-states, 0)
10:        for each move in LegalMoves_state do
11:            new-state ← play(state, move)
12:            score ← score(new-state)
13:            insert(open-states, score, new-state)
14:            if score ≥ best-score then
15:                best-state ← state
16:                best-score ← score
17:            end if
18:        end for
19:    end while
20:    return best-state
21: end function
```

Algorithm 5 The CMA-ES algorithm.

Initialization:
Number of generations N, number of parents λ.
for each graph size n **do**
    Load graphs from previous iterations (curriculum) and select the best parents.
    if the number of parents is insufficient **then**
        Create additional parents randomly.
    end if
    while N is not reached **do**
        Encode the graphs of the parents as vectors.
        Calculate the mean μ and the covariance matriX Σ of the encoded vectors.
        Write $\Sigma = A^\top A$ using SVD.
        Generate vectors $Y = AX + \mu$ where $X \sim \mathcal{N}(0, I_{n^2})$.
        Construct the graphs of the children from the vectors Y.
        for each generated child **do**
            Calculate the score of the child.
            if the score of the child is higher than the previous best score **then**
                Update the best graphs and the best score.
            else if the score of the child is equal to the previous best score **then**
                if the child is not isomorphic to a previously found graph **then**
                    Add the child to the best graphs.
                end if
            end if
        end for
        Select the best candidates among the parents and children to form the next generation.
    end while
    Increment n, record the best scores and graphs.
end for

Appendix B. Conjectures

The upper bounds of the 68 conjectures that were attempted in this paper. The largest eigenvalue μ of the laplacian matrix of some graph must exceed the bound to refute the conjecture. O means the conjecture is still open to this day, X means it has been refuted.

1. O $\max_{v\in V} \sqrt{\frac{4d_v^S}{m_v}}$
2. X $\max_{v\in V} \frac{2m_v^2}{d_v}$
3. X $\max_{v\in V} \frac{m_v^2}{d_v} + m_v$
4. O $\max_{v\in V} \frac{2d_v^2}{m_v}$
5. O $\max_{v\in V} \frac{d_v^2}{m_v} + m_v$
6. O $\max_{v\in V} \sqrt{m_v^2 + 3d_v^2}$
7. O $\max_{v\in V} \frac{d_v^2}{m_v} + d_v$
8. O $\max_{v\in V} \sqrt{d_v\left(m_v + 3d_v\right)}$
9. O $\max_{v\in V} \frac{m_v+3d_v}{2}$
10. O $\max_{v\in V} \sqrt{d_v\left(d_v + 3m_v\right.}$
11. O $\max_{v\in V} \frac{2m_v^3}{d_v^2}$
12. O $\max_{v\in V} \sqrt{2m_v^2 + 2d_v^2}$
13. O $\max_{v\in V} \frac{2m_v^4}{d_v^3}$
14. O $\max_{v\in V} \frac{2d_v^3}{m_v^2}$
15. X $\max_{v\in V} \sqrt{\frac{4m_v^3}{d_v}}$
16. O $\max_{v\in V} \frac{2d_v^4}{m_v^3}$
17. X $\max_{v\in V} \sqrt[4]{5d_v^4 + 11m_v^4}$
18. O $\max_{v\in V} \sqrt{\frac{2m_v^3}{d_v} + 2d_v^2}$
19. O $\max_{v\in V} \sqrt[4]{4d_v^4 + 12d_v m_v^3}$
20. O $\max_{v\in V} \frac{\sqrt{7d_v^2+9m_v^2}}{2}$
21. O $\max_{v\in V} \sqrt{\frac{d_v^3}{m_v} + 3m_v^2}$
22. O $\max_{v\in V} \sqrt[4]{2d_v^4 + 14d_v^2 m_v^2}$
23. O $\max_{v\in V} \sqrt{d_v^2 + 3d_v m_v}$
24. O $\max_{v\in V} \sqrt[4]{6d_v^4 + 10m_v^4}$
25. O $\max_{v\in V} \sqrt[4]{3d_v^4 + 13d_v^2 m_v^2}$
26. O $\max_{v\in V} \frac{\sqrt{5d_v^2+11d_v m_v}}{2}$
27. O $\max_{v\in V} \sqrt{\frac{3d_v^2+5d_v m_v}{2}}$
28. X $\max_{v\in V} \sqrt{\frac{2m_v^4}{d_v^2} + 2d_v m_v}$
29. X $\max_{v\in V} \sqrt{m_v^2 + \frac{3m_v^3}{d_v}}$
30. O $\max_{v\in V} \frac{m_v^3}{d_v^2} + \frac{d_v^2}{m_v}$
31. X $\max_{v\in V} \frac{4m_v^2}{m_v+d_v}$
32. X $\max_{v\in V} \frac{\sqrt{m_v^3(m_v+3d_v)}}{d_v}$
33. O $\max_{v_i\sim v_j} 2\left(d_i + d_j\right) - \left(m_i + m_j\right)$

34. O $\max_{v_i \sim v_j} \frac{2\left(d_i^2+d_j^2\right)}{d_i+d_j}$

35. O $\max_{v_i \sim v_j} \frac{2\left(d_i^2+d_j^2\right)}{m_i+m_j}$

36. X $\max_{v_i \sim v_j} \frac{2\left(m_i^2+m_j^2\right)}{d_i+d_j}$

37. O $\max_{v_i \sim v_j} \sqrt{2\left(d_i^2+d_j^2\right)}$

38. O $\max_{v_i \sim v_j} 2+\sqrt{2\left(d_i-1\right)^2+2\left(d_j-1\right)^2}$

39. O $\max_{v_i \sim v_j} 2+\sqrt{2\left(d_i^2+d_j^2\right)-4\left(m_i+m_j\right)+4}$

40. O $\max_{v_i \sim v_j} 2+\sqrt{2\left(\left(m_i-1\right)^2+\left(m_j-1\right)^2\right)+\left(d_i^2+d_j^2\right)-\left(d_i m_i+d_j m_j\right)}$

41. X $\max_{v_i \sim v_j} 2+\left(m_i+m_j\right)-\left(d_i+d_j\right)+\sqrt{2\left(d_i^2+d_j^2\right)-4\left(m_i+m_j\right)+4}$

42. O $\max_{v_i \sim v_j} \sqrt{d_i^2+d_j^2+2 m_i m_j}$

43. X $\max_{v_i \sim v_j} 2+\sqrt{3\left(m_i^2+m_j^2\right)-2 m_i m_j-4\left(d_i+d_j\right)+4}$

44. O $\max_{v_i \sim v_j} 2+\sqrt{2\left(\left(d_i-1\right)^2+\left(d_j-1\right)^2+m_i m_j-d_i d_j\right)}$

45. X $\max_{v_i \sim v_j} 2+\sqrt{\left(d_i-d_j\right)^2+2\left(d_i m_i+d_j m_j\right)-4\left(m_i+m_j\right)+4}$

46. O $\max_{v_i \sim v_j} 2+\sqrt{2\left(d_i^2+d_j^2\right)-16 \frac{d_i d_j}{m_i+m_j}+4}$

47. O $\max_{v_i \sim v_j} \frac{2\left(d_i^2+d_j^2\right)-\left(m_i-m_j\right)^2}{d_i+d_j}$

48. X $\max_{v_i \sim v_j} \frac{2\left(d_i^2+d_j^2\right)}{2+\sqrt{2\left(d_i^2+d_j^2\right)-4\left(m_i+m_j\right)+4}}$

49. X $\max_{v_i \sim v_j} 2+\sqrt{2\left(m_i^2+m_j^2\right)+\left(d_i-d_j\right)^2-4\left(d_i+d_j\right)+4}$

50. X $\max_{v_i \sim v_j} 2 \frac{d_i^2+d_j^2+m_i m_j-d_i d_j}{d_i+d_j}$

51. X $\max_{v_i \sim v_j} 2\left(m_i+m_j\right)-4 \frac{m_i m_j}{d_i+d_j}$

52. X $\max_{v_i \sim v_j} 2+\sqrt{\sqrt{8\left(m_i^4+m_j^4\right)-8\left(d_i^2+d_j^2\right)+4}-4\left(d_i+d_j\right)+6}$

53. X $\max_{v_i \sim v_j} 2+\sqrt{\sqrt{8\left(m_i^4+m_j^4\right)-8\left(d_i m_i+d_j m_j\right)+4}-4\left(d_i+d_j\right)+6}$

54. X $\max_{v_i \sim v_j} 2+\sqrt{2\left(m_i^2+m_j^2\right)+\left(d_i m_i+d_j m_j\right)-\left(d_i^2+d_j^2\right)-4\left(d_i+d_j\right)+4}$

55. X $\max_{v_i \sim v_j} 2+\sqrt{3\left(m_i^2+m_j^2\right)-\left(d_i^2+d_j^2\right)-4\left(m_i+m_j\right)+4}$

56. O $\max_{v_i \sim v_j} \frac{\left(d_i^2+d_j^2\right)\left(m_i+m_j\right)}{2 d_i d_j}$

57. X $\max_{v_i \sim v_j} 2+\sqrt{2\left(m_i^2+m_j^2\right)-8 \frac{d_i^2+d_j^2}{m_i+m_j}+4}$

58. X $\max_{v_i \sim v_j} 2+\sqrt{2\left(m_i^2+m_i m_j+m_j^2\right)-\left(d_i m_i+d_j m_j\right)-4\left(d_i+d_j\right)+4}$

59. X $\max_{v_i \sim v_j} \frac{2\left(m_i^2+m_i m_j+m_j^2\right)-\left(d_i^2+d_j^2\right)}{m_i+m_j}$

60. X $\max_{v_i \sim v_j} 2 + \sqrt{2\left(m_i^2 + m_i m_j + m_j^2\right) - \left(d_i^2 + d_j^2\right) - 4\left(d_i + d_j\right) + 4}$

61. X $\max_{v_i \sim v_j} \frac{2\left(m_i^2+m_j^2\right)}{2+\sqrt{2\left((d_i-1)^2+(d_j-1)^2\right)}}$

62. X $\max_{v_i \sim v_j} 2 + \sqrt{m_i^2 + 4m_i m_j + m_j^2 - 2d_i d_j - 4\left(d_i + d_j\right) + 4}$

63. X $\max_{v_i \sim v_j} d_i + d_j + m_i + m_j - 4\frac{d_i d_j}{m_i+m_j}$

64. X $\max_{v_i \sim v_j} \frac{m_i m_j (d_i+d_j)}{d_i d_j}$

65. X $\max_{v_i \sim v_j} \frac{(m_i+m_j)(d_i m_i+d_j m_j)}{2m_i m_j}$

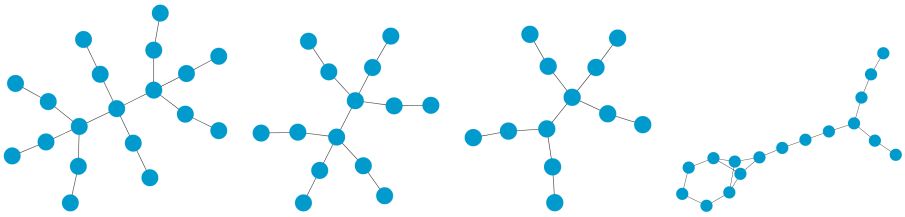
66. X $\max_{v_i \sim v_j} \frac{m_i^2+4m_i m_j+m_j^2-(d_i m_i+d_j m_j)}{d_i+d_j}$

67. X $\max_{v_i \sim v_j} \frac{(m_i+m_j)(d_i m_i+d_{jj} m_j)}{2d_i d_j}$

68. X $\max_{v_i \sim v_j} 2 + \sqrt{\left(m_i - m_j\right)^2 + 4d_i d_j - 4\left(m_i + m_j\right) + 4}$

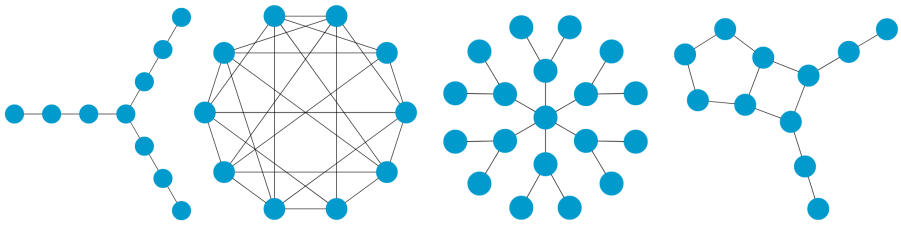
Appendix C. Counter-examples

Some counter-examples for each of the 29 refuted conjectures.



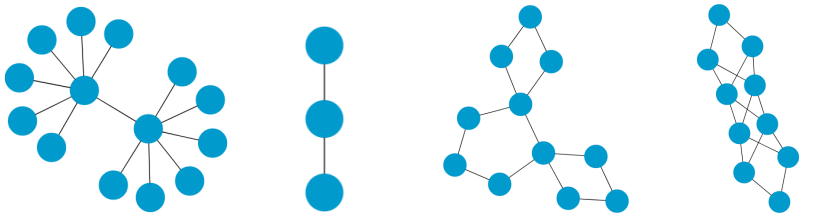
(a) Conjectures 3, 28 and 31 (b) Conjecture 15 (c) Conjecture 29 (d) Conjecture 36

Figure C1. Conjectures 3, 15, 28, 31, 29, and 36



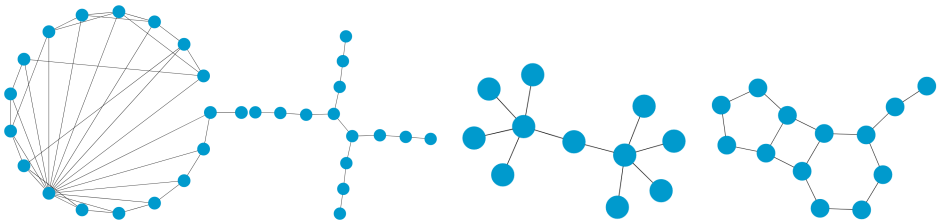
(a) Conjectures 36, 43, 49, 55, 58, 59, 60, 63, 66 and 67 (b) Conjecture 41 (c) Conjecture 41 (tree) (d) Conjecture 43

Figure C2. Conjectures 36, 41, 41 (tree), 43, 45 and others



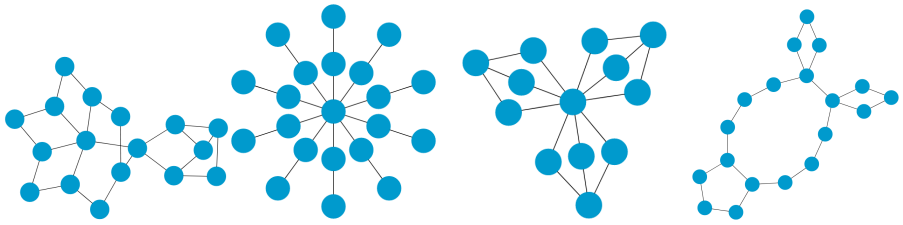
(a) Conjecture 45 (b) Conjectures 48, 57 and 61 (c) Conjectures 49 (d) Conjecture 50

Figure C3. Conjectures 45, 48, 57, 61, 49, and 50



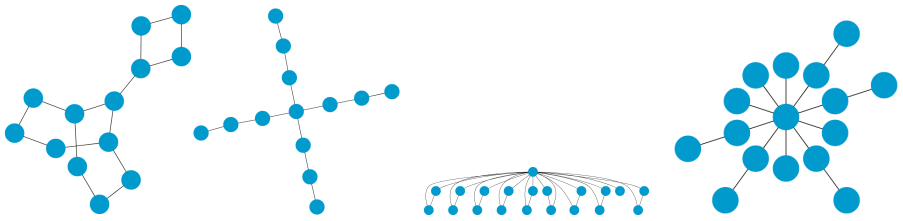
(a) Conjecture 51 (b) Conjectures 52, 53, 62 (c) Conjecture 54 (d) Conjectures 58

Figure C4. Conjectures 51, 52, 53, 62, 54, and 58



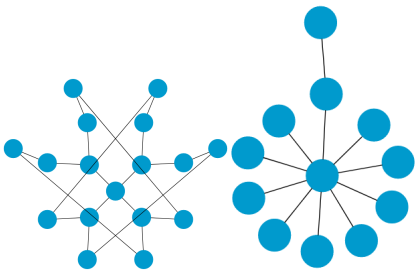
(a) Conjecture 58 (bigger) (b) Conjecture 60 (c) Conjectures 63 (d) Conjectures 63 (bigger)

Figure C5. Conjectures 58 (bigger), 60, 63 and 63 (bigger)



(a) Conjecture 64 (b) Conjectures 64 (tree) (c) Conjecture 65 (d) Conjecture 65 (tree)

Figure C6. Conjectures 64, 64 (tree), 65, and 65 (tree)



(a) Conjectures 66 (b) Conjectures 68

Figure C7. Conjectures 66 and 68