Spectral Graph Theory Conjecture Generation

Elie Duhamel, Tristan Cazenave

LAMSADE, Université Paris Dauphine - PSL, Paris, France

Abstract

This paper deals with the automated generation of mathematical conjectures. More specifically, we address spectral graph theory conjecture generation. We combine the automated generation of conjectures with the automated refutation of the generated conjectures to produce hundreds of thousands of new spectral graph theory conjectures that are difficult to refute.

Introduction

There are hundreds of conjectures in graph theory that have been proven, disproven, or remain open, but this represents only a tiny fraction of the billions of possible conjectures. These conjectures typically take the form of inequalities involving combinations of graph-theoretic variables.

We consider more than 12 different graph-theoretic variables, such as the radius of a graph, the largest eigenvalue of the Laplacian, the degree of a vertex, and others. The number of conjectures of length n grows exponentially due to the combinatorial explosion of mathematical expressions. However, this number can be significantly reduced by identifying isomorphic conjectures or removing unused ones.

Even after this reduction, the total number of conjectures remains very large. Further reduction is achieved through the refutation process.

This refutation work extends the results of Roucairol and Cazenave (Roucairol and Cazenave 2022, 2024), who were the first to apply Monte Carlo Search methods to construct counterexamples for spectral graph theory conjectures. They demonstrated the effectiveness of this approach compared to state-of-the-art neural methods.

Few works have focused on evaluating Monte Carlo Search algorithms for combinatorial problems in graph theory. Cazenave et al. applied them to graph coloring (Cazenave, Negrevergne, and Sikora 2021) followed by Grelier et al. (Grelier, Goudet, and Hao 2022), a topic previously explored only by Edelkamp et al. (Edelkamp et al. 2017). This approach proved competitive with state-of-the-art SAT solvers.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Our focus in this paper is again on undirected connected graphs, and the variables considered include rank, distance, eigenvalues, Laplacian, radius, weight, temperature, and degree.

The choice to restrict ourselves to undirected connected graphs is motivated by the fact that most conjectures in the literature are formulated for this class of graphs.

Similarly, the selection of variables is guided by their computational tractability, as all of them can be computed in polynomial time.

The paper is organized as follows. In the second section, we recall previous work on conjecture generation in graph theory and on Monte Carlo search algorithms. In the third section, we present the algorithm used for the refutation of the generated conjectures. In the fourth section, we describe the generation of conjectures and the associated reductions. Finally, the last section concludes.

Refutation of Graph Theory Conjectures

State Of The Art

"Written the wall" (W12 2012) lists thousands of conjectures on graph theory on undirected connected graphs. The conjectures are inequalities on graph classes (any graphs, trees, triangle-free...). The graph conjectures are propositions that are thought to be true and are awaiting proof or a refutation. They come from software for automated conjecture creation including Ingrid (Dutton, Brigham, and Gomez 1989), GRAPH (Cvetković and Simić 1994), Graffiti (DeLaVina 2005) and AutoGraphiX (Hansen and Caporossi 2000).

Graffiti generates and proposes conjectures in the form of inequalities between graph invariants. It uses an inference engine to attempt proofs or refutations. If a conjecture cannot be proven or refuted automatically, it is tested on a graph database and then submitted to graph theorists for further investigation.

Other tools have also been developed for conjecture refutation. For instance, AutoGraphiX uses Variable

Neighborhood Search (VNS) to disprove conjectures. Local search techniques like VNS are faster than naive exhaustive generation because they introduce constraints that reduce the search space. These techniques have been successfully applied to several combinatorial graph problems (Hertz and de Werra 1987; Mehrabian et al. 2023; Lidický, McKinley, and Pfender 2024).

More recently, deep learning has been applied to combinatorial problems on graphs (Khalil et al. 2017). A major breakthrough was Wagner's deep reinforcement learning approach (Wagner 2021), later improved in (Angileri et al. 2024), and used in Turán theory (Mehrabian et al. 2023) and Ramsey theory (Ghebleh et al. 2024), leading to the disproof of open conjectures (Wagner 2021; Ghebleh et al. 2024; Al-Yakoob et al. 2024). However, this approach often requires hours or more of computation time and is typically limited to graphs of fixed sizes.

Another significant advance is the use of Monte Carlo Search on trees and general graphs for spectral graph conjectures (Roucairol and Cazenave 2024; Taieb et al. 2025), which can disprove conjectures in minutes or even less, depending on the case.

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 (Méhat and Cazenave 2010; Cazenave 2009) and non-game applications including biology (Portela 2018; Edelkamp and Tang 2015), logistics (Edelkamp et al. 2016; Edelkamp and 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 (Cazenave 2017; Cazenave, Negrevergne, and Sikora 2021).

Progressive Generalized Nested Rollout Policy Adaptation

The idea behind Progressive GNRPA is to run GNRPAs of gradually increasing size. This approach is particularly relevant in contexts where allocating the same amount of time to every execution is not as optimal as in timed chess games, where uniform time allocation may lead to inefficiencies. Progressive GNRPA launches multiple successive instances, each with greater search depth or size.

Technically, it is a variation of the standard GNRPA in which the first loop has been modified. In this version, the loop runs a lower-level GNRPA, records the best score obtained, and stores it in a list. The loop continues to iterate as long as meaningful improvements are observed. With each iteration, the size of the GNRPA increases, enabling it to explore progressively longer sequences.

To determine whether the progression is significant, we do not rely solely on local score variation, which can sometimes be null due to the stochastic nature of Monte Carlo exploration used by GNRPA. Identical results may occasionally occur by chance without indicating actual stagnation. Therefore, a global trend is considered, computed over several recent scores. In this work, we use a relative progression metric based on the ratio between the new score and the previous one, offset by a small constant. Special care is taken to avoid division by zero and to handle sign changes appropriately.

The same policy is maintained across iterations to ensure continuity in the learning process. Finally, increasing the level of GNRPA appears to offer limited benefit, as it leads to exponential growth in both search tree size and execution time.

The main Progressive GNRPA algorithm is given in algorithm 1 while the standard GNRPA is given in algorithm 2. The Adapt algorithm used in both is given in algorithm 3. The algorithm to verify the condition to continue the search for Progressive GNRPA is given in algorithm 4.

Algorithm 1: Progressive GNRPA Algorithm

```
$bestScore \leftarrow -\infty$
$scoreSequence \leftarrow \emptyset$
while ScoreSequence is good do
    $(score, state) \leftarrow$ GNRPA($level - 1$, $policy$, $heuristic$)
    Add $score$ to $scoreSequence$
    if $bestScore \leq score$ then
        $bestScore \leftarrow score$
        $bestState \leftarrow state$
    end if
    $policy \leftarrow$ Adapt($policy$, $bestState$, $heuristic$)
    Progress the $params$ {To increase N (iterations) and enable deeper exploration without changing level}
end while
return $bestState$, $bestScore$
```

Algorithm 2: GNRPA Algorithm

```
Input: level, policy
Output: bestState, bestScore
if level == 0 then
    return playout(policy, heuristic)
end if
bestScore $\leftarrow -\infty$
for iteration = 1 to N do
    (score, state) $\leftarrow$ GNRPA(level - 1, policy, heuristic)
    Add score to scoreSequence
    if bestScore $\leq$ score then
        bestScore $\leftarrow$ score
        bestState $\leftarrow$ state
    end if
    policy $\leftarrow$ Adapt(policy, bestState, heuristic)
end for
return bestState, bestScore
```

Algorithm 3: Adapt Algorithm

```
Input: policy, sequence, heuristic
Output: policy
polp $\leftarrow$ policy
state $\leftarrow$ root
for each best in sequence do
    moves $\leftarrow$ legalMoves(state)
    $z \leftarrow 0$
    for each m in moves do
        $z \leftarrow z + \exp(policy[\text{code}[m]])$
    end for
    for each m in moves do
        $polp[m] \leftarrow polp[m] - \alpha \cdot \frac{\exp(policy[m])}{z}$
    end for
    state $\leftarrow$ play(state, move)
end for
policy $\leftarrow$ polp
return policy
```

Algorithm 4: IsScoreSequenceGood

```
1: Input: sequence
2: Output: Boolean
3: if length(sequence) $\leq$ minTime then
4:     return true {Too early to judge — run GNRPA longer}
5: end if
6: derivative $\leftarrow$ sequence[last] / sequence[last - distance]
7: if derivative $< 0$ then
8:     return true {Sign changed — likely decreasing}
9: end if
10: return derivative $>$ minProgress {Checks if progress is significant}
```

Conjecture Generation

We generate a list of conjectures on graph theory, based on the conjectures of "Written on the wall". We select a number of definitions in the article. We used the following graph theory definition:

- The degree is the number of edges connected to the vertex.
- The temperature of a vertex v of graph G is $\frac{d}{n-d}$, where d is the degree of the vertex v and n is the number of vertices of G.
- The weight of an edge uv where u,v is vertices of graph G, is $\frac{1}{\sqrt{d(u)*d(v)}}$ where d is the degree of vertex.
- The Randic index of a graph is the sum of weights of its edges.
- The radius of a graph is the minimum over all vertices u of the maximum distance from u to any other vertex of the graph.

- The diameter of a graph is the maximum distance between two vertices.
- The rank of the matrix.
- The distance of the vertices u, v is the minimum length path between u and v.
- The Laplacian of a graph is the diagonal of degree of the vertex v minus the adjacency matrix.
- The eigenvalue of the matrix.

The generator uses the definitions to make conjectures so that a conjecture is a mathematical combination of operators and definitions. An example:

$$\mathrm{RandicIndex}(G) \leq \mathrm{rank}(L_G) \cdot \max_{(u,v)\in E(G)} \mathrm{weight}(u,v)$$

Beyond the definitions introduced earlier, we also use the natural logarithm, square root, powers, and basic arithmetic operators in the formulation of conjectures.

We use the generator for the generation of conjectures of 12 or fewer words. A word is "<=", "+", "-","log", "RandInc" for the Randic index... For the previous conjecture, the output is "`<= IndRand Adj max-v-u Adj * rank laplac Adj weight u v`". It can also be written as "`<= IndRand Adj * rank laplac Adj max-v-u Adj weight u v`". We generated 36 million conjectures and removed duplicates so as to have in the end a list of 2.5 million conjectures.

The conjecture generator works as follows: to generate conjectures of size n, the process starts with the root node "<=". The "<=" node takes two numerical expressions as arguments. The generator therefore produces all possible pairs of expressions whose combined size is n−1 and which both evaluate to a number. All such combinations are used to build the left and right sides of the inequality. This operation is repeated recursively for each sub-expression until size 1 is reached, at which point a terminal node (such as a variable or constant) is placed.

Experimental Results

The list of conjectures is extensive. We use Progressive GNRPA for conjecture refutation. The values used for minTime, distance, minProgress are 7, 7 and 1.00005. The values had conditions:

- $minTime \geq distance \geq 1$
- $minProgress > 1$

Out of the 2.5 million conjectures, 1.9 million were refuted, 0.3 million were marked as false (due to computations producing NaNs) and 0.3 million were found to be open.

Example of an open conjecture:

$$\lambda_{\max}(G) \leq \lambda_{\max}(L_G) - 1 \tag{1}$$

is interesting,because the algorithm assigned it an almost zero score. The score is defined as the maximum value of the difference between the left part and the right part. A conjecture is considered refuted when its score is greater than 0.001, due to floating-point calculation errors. Examples of conjectures:

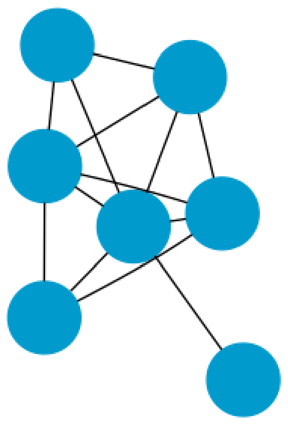
$$\lambda_{\max}(L_G) \leq \mathrm{rad}(G) \cdot \max_{v\in V(G)} \deg(v) \tag{2}$$

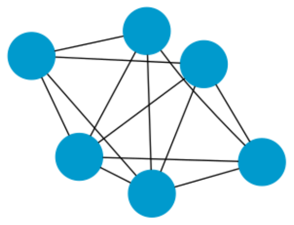
$$\lambda_2(L_G) \leq \text{Randić index} \cdot \sqrt{\mathrm{rank}(G)} \tag{3}$$

$$\lambda_{\max}(L_G) \leq \text{Randić index} \cdot \mathrm{rank}(G) \tag{4}$$

$$\text{Randić index} \leq \mathrm{rank}(G) + 2 \tag{5}$$

The refutation graphs are given in figures 1a, 1b and 1c.


(a) Refuted graph for conjecture (2)


(b) Refuted graph for conjecture (3)

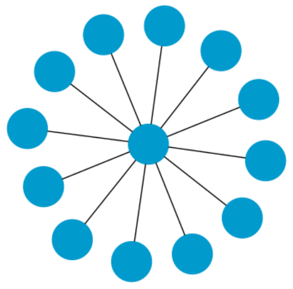

(c) Refuted graph for conjecture (4)

Figure 1: Refutation graphs for conjectures (2), (3) and (4)

Among the 300,000 open conjectures, some are barely usable because their scores are very low or their formulas are too complex.

These results show the efficiency of the GNRPA-based method in exploring large spaces of conjectures. It is difficult to run a big GNRPA for each conjecture since it takes too much time. Some conjectures that were not refuted could possibly be refuted by a longer search. We used a progressive version of GNRPA, where the size of the exploration tree is increased or the search is stopped depending on the evolution of the score. This adaptive strategy allowed us to reduce the theoretical computation time from 200 days to just 8 hours on

Name	word,symbol	entry	output
The Randic index	IndRand	graph	number
The radius	rad	graph	number
Degree	degree	vertex	number
Temperature	temp	vertex	number
Diameter	dia	graph	number
Graph (adjacency matrix)	Adj	None	graph
Laplacian	laplac	graph	matrix
The rank of matrix	rank	matrix	number
The max of vertex	max-v	number	number
The max of edge	max-v-u	number	number
The weight of Edge	weight	edge	number
Vertex v,u	u — v	None	vertex

Table 1: Table of symbols

Algorithm Name	**Time**	**Refuted**
GNRPA	~10 min	85
BFS	~10 min	81
NMCS	~30 min	61
Progressive GNRPA	~2 s	82

Table 2: Results on 100 conjectures (the results may vary slightly).

the trees. The timings were measured on an Intel i7-13650HX CPU 20 cores using low-level, performance-oriented Rust code.

This gain is not without consequences: some conjectures are not explored, and the program wrongly assumes they are true because their scores do not improve. The algorithm is randomized, so the results are never the same. A conjecture may be refuted once and then not refuted again in ten attempts.

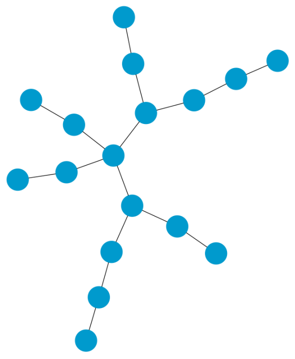
Another optimization involved tuning the compilation parameters to improve performance. For example, we disabled error-handling mechanisms that were not essential during evaluation, such as exception management or runtime checks.

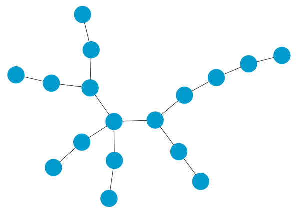
Other algorithms such as NMCS, BFS, and classic GNRPA, could be used but the resolution time is expected to be much longer. Table 2 is the result of running the various algorithms. The GNRPA-progressive is 300 times faster than GNRPA but it refuted 5 percent less conjectures. For large lists of conjectures, GNRPA is the most effective approach.

The Progressive GNRPA can also refute ungenerated conjectures like conjectures (6) and (7) from (Roucairol and Cazenave 2024). Where $mv(v_i)$ is the average of the degrees of v_i's neighbors.

$$\lambda_{\max}(L_G) \leq \max_{v \in V(G)} \frac{4mv(v)^2}{mv(v) + deg(v)} \tag{6}$$

$$\lambda_{\max}(L_G) \leq \max_{v \in V(G)} \sqrt{mv(v)^2 + \frac{3mv(v)^3}{deg(v)}} \tag{7}$$


(a) Refuted graph for conjecture (6)


(b) Refuted graph for conjecture (7)

Conclusion

To process the exhaustive list of conjecture candidates, we developed a modified version of the GNRPA algorithm. This choice was motivated by the fact that GNRPA proved to be the most efficient among the three algorithms evaluated.

The proposed version introduces a modulation mechanism in the execution, enabling the computational effort to be adapted according to the likelihood that a given conjecture can be refuted. This strategy significantly improved the efficiency of the filtering phase by focusing computational resources on the most promising conjectures. As a result, we were able to refute a substantial portion of the initial candidates and generate a total of 300,000 new conjectures.

Table 3: Selected Conjectures. Runtimes are in seconds.

Conjecture	Runtime	Score
$\mathrm{RandicIndex}(G) \leq (2 + (\sqrt{2})^{\mathrm{rank}(G)})$	44	-0.55025253169417
$\mathrm{RandicIndex}(G) \leq (1 + (2)^{\lambda_{\max -1}(L_G)})$	51	-0.28993647761945684
$\mathrm{RandicIndex}(G) \leq (1 + (\lambda_{\max}(L_G))^2)$	59	-8.31451569390679
$\mathrm{RandicIndex}(G) \leq 2 * (\mathrm{rank}(G) + (1 + 2))$	44	-8.55025253169417
$\mathrm{RandicIndex}(G) \leq \frac{2}{(\mathrm{rank}(G))^{(1-2)}}$	42	-2.550252531694169
$\mathrm{RandicIndex}(G) \leq \max_{v \in V(G)} (2)^{(2+\mathrm{degree}(v))}$	50	-7.585786437626906
$\mathrm{RandicIndex}(G) \leq \max_{v \in V(G)} ((1 + 2))^{\mathrm{degree}(v)}$	48	-0.585786437626906
$\mathrm{RandicIndex}(G) \leq \max_{v \in V(G)} ((2 + \mathrm{degree}(v)))^2$	43	-7.585786437626906
$\mathrm{RandicIndex}(G) \leq \max_{v \in V(G)} (2 * \mathrm{degree}(v))^2$	53	-7.585786437626906
$\mathrm{RandicIndex}(G) \leq (\sqrt{2} + \sqrt{(\mathrm{rank}(G))^2})$	46	-0.0536761712699656
$\mathrm{RandicIndex}(G) \leq ((1 + \max_{v \in V(G)} \mathrm{degree}(v)))^2$	49	-0.585786437626906
$\mathrm{RandicIndex}(G) \leq ((1 + \lambda_{\max -1}(L_G)))^2$	53	-5.823839986008376
$\mathrm{RandicIndex}(G) \leq (\frac{2}{\frac{1}{\mathrm{rank}(G)}} - 2)$	42	-0.5502525316941691
$\lambda_{\max -1}(G) \leq (1 + (2 + \lambda_{\max}(G)))$	52	-3.0111435201085848
$\lambda_{\max -1}(G) \leq (1 + (\lambda_{\max}(G) - 1))$	61	-0.012046099581604963
$\lambda_{\max -1}(G) \leq (2 + (\lambda_{\max}(G) - 2))$	56	-0.014917065241241367
$\lambda_{\max -1}(G) \leq 2 * \log(\lambda_{\max}(L_G))$	45	-0.15645678373681005
$\lambda_{\max -1}(G) \leq (2 + \sqrt{\lambda_{\max -1}(L_G)})$	42	-0.9869274393702367
$\lambda_{\max -1}(G) \leq (\sqrt{\mathrm{rank}(G)})^{\sqrt{\mathrm{rank}(G)}}$	47	-1.8564340080028612
$\lambda_{\max -1}(G) \leq (\log(\mathrm{rank}(G)))^{\sqrt{\mathrm{RandicIndex}(G)}}$	45	-0.15310717617681946
$\lambda_{\max -1}(G) \leq (\lambda_{\max}(L_G) - \sqrt{2})$	50	-0.6815995760277991
$\lambda_{\max -1}(G) \leq ((2 + \lambda_{\max}(G)) - 1)$	61	-1.01114352010859
$\lambda_{\max -1}(G) \leq (2 * \lambda_{\max}(G) - 1)$	62	-1.0501441756363505
$\lambda_{\max -1}(G) \leq \frac{2 * \lambda_{\max}(G)}{2}$	45	-0.012708926281729926
$\lambda_{\max -1}(G) \leq \sqrt{(\mathrm{rank}(G) + \lambda_{\max -1}(G))}$	46	0
$\lambda_{\max -1}(G) \leq \sqrt{\mathrm{RandicIndex}(G) * (2 + 2)}$	42	-1.206912251116893
$\lambda_{\max -1}(G) \leq \sqrt{(\max_{v \in V(G)} \mathrm{degree}(v))^2}$	44	-0.25955477614946876
$\mathrm{rank}(G) \leq (1 + (2)^{2 * \lambda_{\max}(G)})$	53	-0.3400471496615367
$\mathrm{rank}(G) \leq (1 + ((\lambda_{\max}(G))^2)^2)$	50	-0.04962985252513619
$\mathrm{rank}(G) \leq (2 - (\mathrm{RandicIndex}(G) - 2 * \mathrm{rank}(G)))$	48	-0.5502525316941691
$\mathrm{rank}(G) \leq (2 + ((\lambda_{\max}(G))^2)^2)$	56	-1.0496298525252001
$\mathrm{rank}(G) \leq \max_{v \in V(G)} \mathrm{RandicIndex}(G) * (2 + \mathrm{temperature}(v))$	48	-0.6979092531226989
$\mathrm{rank}(G) \leq \max_{(v,u) \in E(G)} (\mathrm{degree}(v))^{(1+\mathrm{degree}(u))}$	49	-10
$\mathrm{rank}(G) \leq \max_{(v,u) \in E(G)} ((1 + \mathrm{degree}(u)))^{\mathrm{degree}(v)}$	51	-10
$\mathrm{rank}(G) \leq ((\sqrt{2})^2)^{\lambda_{\max}(L_G)}$	46	-6.2210532465756785
$\mathrm{rank}(G) \leq ((1 + (2 + 2)))^{\lambda_{\max}(G)}$	53	-7.806862234846115
$\mathrm{rank}(G) \leq ((1 + (2)^{\lambda_{\max}(G)}))^2$	57	-8.173321598999152
$\mathrm{RandicIndex}(G) \leq \frac{1}{\frac{(\sqrt{2}-1)}{\mathrm{rank}(G)}}$	43	-4.207106781186547
$\mathrm{RandicIndex}(G) \leq 2 * (2)^{\sqrt{2 * \mathrm{rank}(G)}}$	42	-8.7562391343262
$\mathrm{RandicIndex}(G) \leq 2 * (\mathrm{rank}(G) + (2 - \sqrt{2}))$	42	-3.7218254069479793
$\mathrm{RandicIndex}(G) \leq \mathrm{rank}(G) * \sqrt{2}$	42	-0.20710678118654968
$\mathrm{RandicIndex}(G) \leq (2 + ((\sqrt{2} - \mathrm{rank}(G)))^2)$	45	-3.2365440327094097
$\mathrm{RandicIndex}(G) \leq \max_{v \in V(G)} \mathrm{rank}(G) * (2 + \mathrm{temperature}(v))$	42	-6.1058080872497245
$\mathrm{RandicIndex}(G) \leq (\mathrm{rank}(G))^{\frac{1}{(2-\sqrt{2})}}$	44	-5.210829102455075
$\mathrm{RandicIndex}(G) \leq (\sqrt{2} + (2 + (2 + \mathrm{rank}(G))))$	46	-3.964466094067265
$\mathrm{RandicIndex}(G) \leq \sqrt{2} * ((\mathrm{rank}(G) - 2))^2$	45	-0.20710678118654968
$\mathrm{RandicIndex}(G) \leq \lambda_{\max}(G) * \log((2)^{\mathrm{rank}(G)})$	44	-2.8680186984135085
$\mathrm{RandicIndex}(G) \leq ((1 - (2)^{\sqrt{2}}))^{\mathrm{rank}(G)}$	49	-2.23814563708841

Continued on next page

Table 3 − following

Conjecture	Runtime	Score
$\mathrm{RandicIndex}(G) \leq ((\frac{1}{\mathrm{RandicIndex}(G)} - \mathrm{rank}(G)))^2$	44	-9.115964923966015
$\mathrm{RandicIndex}(G) \leq (((\log(1) - \mathrm{rank}(G)))^2 - 1)$	42	-9.55025253169417
$\mathrm{RandicIndex}(G) \leq \log((2 + (2)^{(\mathrm{rank}(G))^2}))$	42	-5.640637937765768
$\mathrm{RandicIndex}(G) \leq \log(2 * ((\mathrm{rank}(G))^2)^2)$	43	-0.7885771567336768
$\lambda_{\max}(G) \leq \max_{(v,u)\in E(G)} \frac{\mathrm{weight}(u,v)}{\mathrm{temperature}(u)}$	43	-2.366196697460232
$\lambda_{\max -1}(G) \leq \max_{v\in V(G)}(\mathrm{rank}(G) + \log(\mathrm{degree}(v)))$	42	-3.433690230615244
$\mathrm{diameter}(G) \leq (\mathrm{RandicIndex}(G) + (\mathrm{rank}(G) - 2))$	42	-2
$\mathrm{diameter}(G) \leq ((\mathrm{rank}(G) + \mathrm{RandicIndex}(G)) - 2)$	44	-2
$\mathrm{radius}(G) \leq (\mathrm{rank}(G) - \frac{\mathrm{diameter}(G)}{\mathrm{RandicIndex}(G)})$	42	-0.5
$\mathrm{diameter}(G) \leq (1 + (\mathrm{rank}(G) + \sqrt{\mathrm{RandicIndex}(G)}))$	44	-3
$\mathrm{diameter}(G) \leq (\mathrm{rank}(G) + (2)^{\lambda_{\max -1}(G)})$	50	-1
$\mathrm{diameter}(G) \leq \mathrm{rank}(G) * \frac{\mathrm{RandicIndex}(G)}{\mathrm{radius}(G)}$	42	-0.8284271247461881
$\mathrm{diameter}(G) \leq (\mathrm{rank}(G) + (\mathrm{RandicIndex}(G) - \sqrt{2}))$	45	-2.585786437626905
$\mathrm{diameter}(G) \leq (\mathrm{rank}(G) + \log((2)^{\mathrm{RandicIndex}(G)}))$	43	-3.3759630583824354
$\mathrm{diameter}(G) \leq (\mathrm{rank}(G) + \log(\lambda_{\max}(L_G)))$	44	-1.3777444645410952
$\mathrm{diameter}(G) \leq (\lambda_{\max}(G) + (\mathrm{rank}(G) - 1))$	44	-0.9696155060244145
$\mathrm{diameter}(G) \leq (\lambda_{\max -1}(G) + 2 * \mathrm{rank}(G))$	43	-2
$\mathrm{diameter}(G) \leq (\sqrt{\mathrm{RandicIndex}(G)} - (1 - \mathrm{rank}(G)))$	47	-1
$\mathrm{diameter}(G) \leq \log(\mathrm{RandicIndex}(G) - 1 + \mathrm{rank}(G)$	43	-0.3862943611198908
$\lambda_{\max -1}(G) \leq (\mathrm{diameter}(G))^{\frac{2}{\mathrm{radius}(G)}}$	44	-0.038354871558563985
$\mathrm{radius}(G) \leq (\mathrm{rank}(G) + \sqrt{2} * \lambda_{\max}(G))$	42	-4
$\mathrm{radius}(G) \leq \frac{\mathrm{diameter}(G)}{\log((2+\lambda_{\max -1}(G)))}$	42	-0.0652968125593576
$\mathrm{radius}(G) \leq ((\mathrm{rank}(G) - 2) + \sqrt{\lambda_{\max}(G)})$	42	-1.7320508075688776
$\mathrm{diameter}(G) \leq (1 + (\mathrm{rank}(G) + \frac{\mathrm{RandicIndex}(G)}{2}))$	42	-3
$\mathrm{diameter}(G) \leq -1 + \mathrm{rank}(G) + \mathrm{RandicIndex}(G)$	43	-3
$\mathrm{diameter}(G) \leq (2 + (\mathrm{rank}(G) - \sqrt{\lambda_{\max}(G)}))$	43	0
$\mathrm{diameter}(G) \leq (2 + (\mathrm{RandicIndex}(G) + (\mathrm{rank}(G) - 2)))$	44	-4
$\mathrm{diameter}(G) \leq (2 - (\sqrt{\lambda_{\max}(G)} - \mathrm{rank}(G)))$	42	-0.06281113512294301
$\mathrm{diameter}(G) \leq (\mathrm{rank}(G) - (2 - (\mathrm{RandicIndex}(G) - 2)))$	45	0
$\mathrm{diameter}(G) \leq (\mathrm{rank}(G) + \frac{\mathrm{RandicIndex}(G)}{(1+2)})$	43	-1.333333333333333
$\mathrm{diameter}(G) \leq (\mathrm{rank}(G) - (\sqrt{\mathrm{rank}(G)} - \mathrm{RandicIndex}(G)))$	47	-2.585786437626905
$\mathrm{diameter}(G) \leq (\mathrm{rank}(G) + \log((2)^{\lambda_{\max}(G)}))$	42	-1.3652334347879744
$\mathrm{diameter}(G) \leq \mathrm{RandicIndex}(G) + \mathrm{rank}(G)$	44	-4
$\mathrm{diameter}(G) \leq (\mathrm{RandicIndex}(G) - (\log(\mathrm{radius}(G)) - \mathrm{rank}(G)))$	42	-4
$\mathrm{diameter}(G) \leq ((2 + \mathrm{rank}(G)) + \frac{\mathrm{RandicIndex}(G)}{2})$	46	-4
$\mathrm{diameter}(G) \leq (2 * \mathrm{rank}(G) + \sqrt{\lambda_{\max -1}(G)})$	43	-2
$\mathrm{diameter}(G) \leq ((\mathrm{rank}(G) - 1) + \lambda_{\max}(L_G))$	42	-2.9659461993677994
$\mathrm{diameter}(G) \leq ((\mathrm{rank}(G) - 2) - (1 - \mathrm{RandicIndex}(G)))$	47	-1
$\mathrm{diameter}(G) \leq ((\mathrm{rank}(G))^2 - (2 + \mathrm{RandicIndex}(G)))$	42	-4.55025253169417
$\mathrm{diameter}(G) \leq (\frac{\mathrm{RandicIndex}(G)}{2} - (2 - \mathrm{rank}(G)))$	47	0
$\mathrm{diameter}(G) \leq ((\mathrm{rank}(G) + \mathrm{RandicIndex}(G)) - \log(\mathrm{radius}(G)))$	43	-4
$\mathrm{diameter}(G) \leq ((\mathrm{RandicIndex}(G) + (2 + \mathrm{rank}(G))) - 1)$	43	-5
$\mathrm{diameter}(G) \leq (((\mathrm{rank}(G) + \mathrm{RandicIndex}(G)) - 1) - 2)$	42	-1
$\mathrm{diameter}(G) \leq \sqrt{((\mathrm{rank}(G) + \lambda_{\max}(G)))^2}$	44	-1.9696155060244145
$\mathrm{RandicIndex}(G) \leq (\mathrm{rank}(G) + \frac{\mathrm{RandicIndex}(G)}{\sqrt{\mathrm{radius}(G)}})$	44	-2.4038059222874413
$\mathrm{RandicIndex}(G) \leq \mathrm{diameter}(G) + \mathrm{rank}(G) - 1$	42	-0.7406084689826562
$\mathrm{RandicIndex}(G) \leq ((1 + 2))^{(\mathrm{rank}(G) - \mathrm{radius}(G))}$	43	-3.550252531694169

References

2012. Latest version of ”Written on the wall”. Accessed: https://independencenumber.wordpress.com/wp-content/uploads/2012/08/wow-july2004.pdf.

Al-Yakoob, S.; Ghebleh, M.; Kanso, A.; and 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.; and Morandin, F. 2024. A Systematization of the Wagner Framework: Graph Theory Conjectures and Reinforcement Learning. *arXiv preprint arXiv:2406.12667*.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; and 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. Nested Monte-Carlo search. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Cazenave, T. 2017. Nested rollout policy adaptation with selective policies. In Cazenave, T.; Winands, M. H. M.; Edelkamp, S.; Schiffel, S.; Thielscher, M.; and Togelius, J., eds., *CGW/GIGA-2016*, volume 705 of *Communications in Computer and Information Science (CCIS)*, 44–56. Springer, Cham.

Cazenave, T.; Negrevergne, B.; and 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*, 100–115. Springer International Publishing.

Cvetković, D.; and 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*, volume 69, 81.

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

Edelkamp, S.; Externest, E.; Kühl, S.; and 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.; and Lawo, M. 2016. Monte-Carlo tree search for logistics. In *Commercial Transport: Proceedings of the 2nd Interdisciplinary Conference on Production Logistics and Traffic 2015*, 427–440. Springer International Publishing.

Edelkamp, S.; and Greulich, C. 2014. Solving physical traveling salesman problems with policy adaptation. In *2014 IEEE Conference on Computational Intelligence and Games*, 1–8. IEEE.

Edelkamp, S.; and Tang, Z. 2015. Monte-Carlo tree search for the multiple sequence alignment problem. In *Proceedings of the International Symposium on Combinatorial Search*, volume 6, 9–17.

Ghebleh, M.; Al-Yakoob, S.; Kanso, A.; and Stevanović, D. 2024. Reinforcement learning for graph theory, II. Small Ramsey numbers. *arXiv preprint arXiv:2403.20055*.

Grelier, C.; Goudet, O.; and Hao, J.-K. 2022. On monte carlo tree search for weighted vertex coloring. In *European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar)*, 1–16. Springer.

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

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

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

Lidický, B.; McKinley, G.; and Pfender, F. 2024. Small Ramsey numbers for books, wheels, and generalizations. *arXiv preprint arXiv:2407.07285*.

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

Méhat, J.; and 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.

Roucairol, M.; and Cazenave, T. 2022. Refutation of spectral graph theory conjectures with monte carlo search. In *International Computing and Combinatorics Conference*, 162–176. Springer International Publishing.

Roucairol, M.; and Cazenave, T. 2024. Refutation of Spectral Graph Theory Conjectures with Search Algorithms. arXiv:2409.18626.

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

Taieb, L.; Roucairol, M.; Cazenave, T.; and Harutyunyan, A. 2025. Automated Refutation with Monte Carlo Search of Graph Theory Conjectures on the Maximum Laplacian Eigenvalue. In *LION 19*, LNCS.

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