I've got 99 vertices but a solution to Conway's problem ain't one

Caitlin Hutnyk

January 6, 2020

Abstract

Our project seeks to answer Conway's 99-graph problem, posed eloquently by Conway himself as: "Is there a graph with 99 vertices in which every edge belongs to a unique triangle and every nonedge to a unique quadrilateral?" Current spectral and combinatorial techniques for construction and disproof have failed in this case, and the space of possible graphs is too large for a brute-force computational search. Our approach is to use a hybrid evolutionary algorithm, which simulates evolution on a population of possible solutions while also performing local optimizations.

1 Conway's 99-graph problem

This paper describes the context and theory behind Conway's 99-graph problem, and explains a hybrid evolutionary algorithm based approach to try to construct a solution. We begin with the graph theoretic and spectral context and theory behind Conway's question, then discuss the basics of hybrid evolutionary algorithms. Finally we discuss the approach and the results.

In Conway's paper *Five $1000 problems* the 99-graph problem is stated as follows: "Is there a graph with 99 vertices in which every edge (i.e. pair of joined vertices) belongs to a unique triangle and every nonedge (pair of unjoined vertices) to a unique quadrilateral?" (Conway 2017). As stated the problem, and its importance, may seem unclear. The goal of the next few sections is to provide context to the problem, and to help explain the ideas behind our approach.

1.1 Strongly Regular Graphs

We begin with a discussion of strongly regular graphs.

Definition 1. *A graph $G = (V, E)$ is regular if $\exists d \in \mathbb{N}$ such that $\forall v \in V \ |\{uv \ s.t. \ uv \in E \}| = d$*

In other words, a graph is regular if every vertex has the same number of neighbours. We can say that a regular graph G has parameters (n, d) with $n = |V|$ and d being $d \in \mathbb{N}$ in the definition above.

Definition 2. *A regular graph is said to be strongly regular if $\exists a, c \in \mathbb{N}$ such that $\forall u, v \in V$*

$$|\{x \in V s.t. \ ux, \ vx \in E\}| = \begin{cases} a, \ if \, uv \in E \\ c, \ if \, uv \notin E \end{cases}$$

In other words, a graph is strongly regular if it is regular, and if every pair of adjacent vertices have a mutual neighbours, and every pair of non-adjacent vertices have c mutual neighbours. We say a strongly regular graph, or SRG, has parameters (n, d, a, c).

Proposition 1. *The parameters (n, d, a, c) of a strongly regular graph satisfy*

$$d(d - a - 1) = c(n - d - 1) \tag{1}$$

Proof. Fix a vertex x. Let Y be the set of x's d neighbours, and call Z the graph with x and Y removed, having size $n-d-1$. Count the edges between Y and Z in two ways. Each element in Y has d neighbours, a of which are shared with x. So each of the d elements in Y have $d-a-1$ neighbours in Z. On the other hand each element in Z has c mutual neighbours with x, all of which are in Y. So each of the $n-d-1$ vertices in Y is adjacent to c members of Z. Combining these two facts yields the equation above.

□

1.2 Spectral Theory and Eigenvalue Multiplicity

The adjacency matrix A of a graph $G=(V,E)$ is defined as follows.

$$A(u,v)=\begin{cases}1, \text{ if } uv\in E\\ 0, \text{ otherwise}\end{cases}$$

Proposition 2.

$$A^2(u,v)=\begin{cases}deg(u) \text{ if } u=v\\ u \text{ and } v\text{'s number of mutual neighbours otherwise}\end{cases}$$

Proof. $A^2(u,v)$ is the dot product of the u^{th} row and the v^{th} column. The k^{th} term in the dot product is

$$k=\begin{cases}1 \text{ if } uk, vk\in E\\ 0 \text{ otherwise}\end{cases}$$

So the matrix entry is equal to the number of neighbours that u and v share. If $u=v$ then u shares all of its neighbours with v, so the matrix entry is equal to its degree. □

Proposition 3. *The adjacency matrix of a regular graph has the simple eigenvalue d, with the constant function as its eigenfunction.*

Proof. Multiplying A by the column vector filled with ones we see that the i^{th} term in the vector is equal to the number of entries in the i^{th} row of A which are 1. For a regular graph this is d for every column. So the product is the column vector filled with d's, which is exactly d times the eigenfunction. □

Theorem 1. *(Nica 2018) The adjacency matrix of a strongly regular graph (n,d,a,c) has 3 distinct eigenvalues. The simple eigenvalue d and*

$$\frac{a-c\pm\sqrt{\Delta}}{2}, \text{ with } \Delta=(a-c)^2+4(d-c)>0 \tag{2}$$

having multiplicities

$$\frac{1}{2}\Big(n-1\pm\frac{(n-1)(a-c)+2d}{\sqrt{\Delta}}\Big) \tag{3}$$

Proof. Applying Proposition 2 to the adjacency matrix of a SRG with the above parameters, we get that $A^2=(a-c)A+(d-c)I+cJ$. For α a non-trivial eigenvalue of A with eigenfunction f we see that $Jf=0$. A is symmetric, and eigenvectors of symmetric matrices are orthogonal. Note that the constant function is an eigenvector of A for a regular graph, and so the same is true for a SRG, so the constant function and f are orthogonal. Each term in Jf is the constant function times f, so $Jf=0$. This means that $\alpha^2=\alpha(a-c)+(d-c)$. Applying the quadratic formula yields the equation above.

For the multiplicities, call the two eigenvalues above α and α' and their multiplicites m and m'. The sum of the multiplicities is the size of the matrix n, and the sum of all the eigenvalues counted

with multiplicity is the trace of the matrix. So $1 + m + m' = n$ and $d + m\alpha + m'\alpha' = 0$. Again combining these equations with the formulas for α and α' yields the desired equation.

□

The real power of the above theorem lies in the fact that the multiplicities must be integral. This requires that either $(n-1)(a-c)+2d = 0$ or $\exists\, t$ such that $t^2 = (a-c)^2+4(d-c)$ and $t \mid (n-1)(a-c)+2d$. This turns out to be very restrictive.

Theorem 2. *The only possible values of n and d for a strongly regular graph with parameters $a = 1$ and $c = 2$ are (9,4), (99,14), (243,22), (6273,112), and (494019,994).*

Proof. First we can consider the first case above, $(n-1)(a-c)+2d = 0$. Proposition 1 with $a = 1$, $c = 2$ shows $d(d-2) = 2(n-d-1)$. Combining these two equations gives

$$d^2 - 2d = 2(2d+1) - 2d - 2$$
$$d = \pm 4$$

This gives the unique set of parameters (9,4,1,2). For the second case let $t^2 = (a-c)^2 + 4(d-c) = 1 + 4(d-2) = 4d - 7$, where $t \mid 2d - n + 1$. Proposition 1 gives $\frac{d^2+2}{2} = n$. Applying the divisibility equation gives

$$t \mid 2d - \frac{d^2+2}{2} + 1$$
$$t \mid 4d - d^2$$

Then we can manipulate the definition of t to get $d = \frac{t^2+7}{4}$. Then

$$t \mid \left(\frac{t^2+7}{4}\right)^2 - t^2 - 7$$
$$t \mid t^4 - 63$$

So $t \mid 63$, which leaves only the possible values $t = 1, 3, 7, 9, 21, 63$. $t = 1$ gives $d < 0$, and $t = 3$ yields d non-integer. The rest of the possible values of t, along with Proposition 1 and $t^2 = 4d - 7$ yield the possible values of n and d.

□

In fact, constructions exist for (9,4,1,2) and (243,22,1,2): the Paley graph $P(9)$ and the *Berlekamp–van Lint–Seidel graph* (Berlekamp 1973). So the smallest graph for these parameters whose existence is unknown is that with parameters (99,14,1,2). This is the source of Conway's 99-graph problem. Every edge belonging to a unique triangle means that every adjacent pair of vertices has exactly one common neighbour, $a = 1$, and every non-edge being in a unique quadrilateral gives the other parameter $c = 2$. Then Proposition 1 implies that $d = 14$. While the initial phrasing of the question did not mention strongly regular graphs, we see that Conway's question is asking precisely about the existence of this smallest unknown but possible graph.

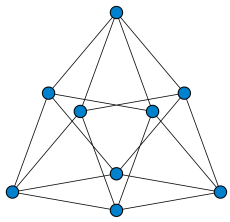


Figure 1: The strongly regular graph (9,4,1,2): the Paley graph $P(9)$

2 Evolutionary Algorithms

The basic idea behind an evolutionary algorithm is to simulate evolution on a population of potential solutions. This requires an evaluation function, some notion of random mutation, and some way of having the potential solutions "reproduce." A basic implementation would look like

Algorithm 1 Evolutionary Algorithm

randomly initialize population
Evaluate current generation
while Solution not found **do**
  Select best from population for reproduction
  Create new individuals by reproduction and random mutation
  Evaluate fitness of new individuals
  Replace weakest individuals in population with new individuals
end while
Return solution or best found

In general, an evolutionary algorithm alone is not effective enough to find good solutions to problems, so a hybrid algorithm is used: an evolutionary algorithm combined with a local search algorithm. Different types of search algorithms have seen success with different types of NP-hard graph problems. Some opt for a more heuristically motivated algorithm (Sidi Mohamed Douiri 2013), while others use a Tabu algorithm (Philippe Galinier 1999), while others use a more generic local search algorithm (Keiko Kohmoto 2003).

For reasons that will become clear in the next section, the local search algorithm we used is simulated annealing. The idea behind simulated annealing is to avoid getting stuck in local minima by accepting worse solutions at each step with probability proportional to a cooling temperature. The temperature starts very high, allowing lots of exploration, and is gradually lowered to zero, where the algorithm behaves like a simple local search. The algorithm then, depends on the initial and final temperatures T and FT, as well as the cooling ration $0 < \alpha < 1$. A basic implementation would look like

Algorithm 2 Simulated annealing

Given initial solution x, and evaluation ex
while $T > FT$ **do**
  Pick random neighbour x', and find evaluation ex'
  $\delta = ex' - ex$
  if $\delta \leq 0$ then $x = x'$, $ex = ex'$
  else set $x = x'$, $ex = ex'$ with probability $e^{-\delta/T}$
  $T = \alpha T$
end while
Return x

Usually the algorithm is run on any new individual just before it is added to the population, be it on one of the individuals of the initial population, or the result of the reproduction operator.

3 Approach

With all of the background out of the way, we can finally talk about the details of our approach. The algorithm as a whole follows the same structure as the generic outline above, but several details need to be explained. The first question is how to initialize the population of potential solutions.

Since generating random regular graphs has well-known algorithms, and evaluating for regularity and strong regularity would be tricky, it made sense to have every graph in the population be regular with $n = 99$ and $d = 14$. This makes evaluation clear and simple, but as we will see, it also makes the step function and reproduction algorithm slightly more complicated.

3.1 Fitness

The idea behind the fitness function is simple: for every pair of vertices find the distance between their number of mutual neighbours and the number they are suppose to have. Using the spectral identity from before this is

$$\text{fitness} = \sum_{i \neq j \in V} \begin{cases} |A^2(i,j) - 1| \text{ if } ij \in E \\ |A^2(i,j) - 2| \text{ otherwise} \end{cases} \tag{4}$$

So the desired SRG would have fitness 0. Manually computing this for every $\binom{99}{2} = 4851$ pair of vertices is slow. We can use the speed of matrix operations and the spectral identities seen before to improve the speed. But first, another useful fact from linear algebra.

Proposition 4.

$$tr(A^T A) = \sum_{i=1}^{n} \sum_{j=1}^{n} A(i,j)^2 \tag{5}$$

Proof. The trace is the sum of the dot products of each of the rows of A. Each term in each of those dot products will be the square of a unique $A(i,j)$, so the above formula holds.

□

Returning to the fitness function, replace the absolute values with squares, and note that the desired number of mutual neighbours is $2 - A(i,j)$. Also, since the adjacency matrix of a graph is symmetric for simple graphs, we can compute a similar sum to that from (4), namely

$$\text{fitness} = tr((A^2 - 2J + A)^2) \tag{6}$$

However, the above equation includes the sum of the squares of the diagonals $i = j$. This can be removed by seeing that for any regular graph $(A^2 - 2J + A)(i,i) = d - 2$. The sum of the diagonal entries will be a constant for any graph in our population, so the above formula evaluates strong regularity effectively, but we want a solution to have fitness 0 so that our function behaves like an optimization problem. So our final fitness function is

$$\text{fitness} = tr((A^2 - 2J + A)^2) - 12^2 \times 99 \tag{7}$$

3.2 Reproduction

Generally in an evolutionary algorithm the goal of the reproduction operator is to take some of the traits of each of the parents and instill them in the offspring. What exactly these traits might be in our case is not so clear: the evaluation for each vertex relies on the configuration of every other vertex, and we have yet to develop any sort of heuristic for what sort of substructure is positive or negative. Further, we cannot just take two subgraphs and reconnect them carelessly: we need the offspring to be regular before we can add it to the population.

Because of these difficulties, the reproduction operator is currently motivated by quite a simple heuristic: take the induced subgraph on some k vertices from the first parent, and the induced subgraph on the remaining $n - k$ vertices from the second parent. We reconnect these two subgraphs using as many edges from the parents as possible, and as many edges which cross the cut as possible. Unfortunately, we almost always have to insert some edges which are in neither of the parents in order

to preserve the regularity condition. Ideally we want to keep as much of the subgraph structure from each of the parents as possible, although because of the restrictions on the problem space and the inter-relatedness of the evaluation, this is difficult.

The other aspect of the algorithm is how to determine k. We could always pick $k = n/2$, but it makes sense to pick k based on the fitness of each of the parents. So we pick k from a normal distribution centered around $n/2 + w$ with w some number representing the difference in fitness evaluation of the parents.

Our current implementation of the above algorithm succeeds about half the time. So the question arises: is it always possible to reconnect the two subgraphs? More formally, given two induced subgraphs g_1 and g_2 of two regular graphs with parameters (n, d) with $|g_1| + |g_2| = n$, can we always produce a regular graph with the original parameters with g_1 and g_2 as subgraphs by only adding edges?

It is relatively easy to construct examples where one of the parents is disconnected, or where one of g_1 or g_2 is already d regular. Take both parent graphs to be copies of the 2-regular graph on 7 vertices from Figure 2. Taking g_1 with vertices 0, 1, 2 and g_2 with vertices 0, 1, 2, 3 we see that the reproduction requirements are met and it is impossible to connect them back up into a regular graph $(6, 2)$.

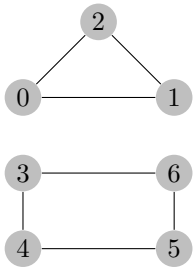
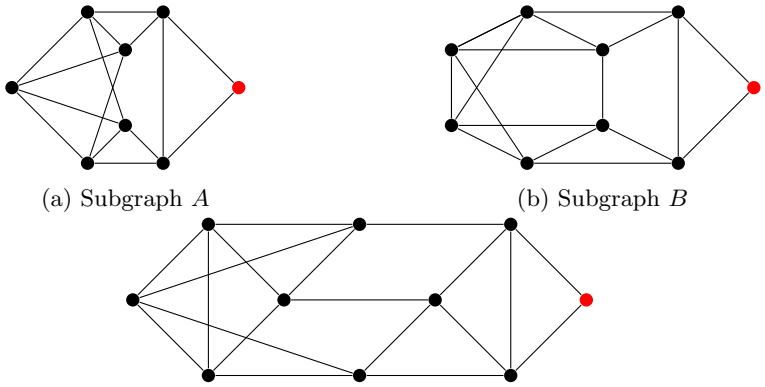


Figure 2: Disconnected regular graphs which may not be able to reproduce

Cases like the one above do not seem very realistic given the size and relatively high degree of the graphs in question. So we can refine the question by qualifying that both parents must be connected and that both subgraphs must not already be d regular. Although it may not be immediately obvious, the answer to this question is also no. We construct three subgraphs, each with a designated special vertex which we will use to connect them up after. The idea is to construct two connected regular graphs with two subgraphs g_1 and g_2 such that both are d regular except for one vertex of degree $d-2$. Since double edges are not allowed in simple graphs, this means that the two subgraphs cannot be stitched back together.

Using the subgraphs of Figure 3 we can construct our counterexample. Let G_1 be the graph obtained by merging the red vertices of A and C into one, and G_2 from merging the red vertices of two copies of B. Both graphs are 4-regular on 17 vertices, so our initial conditions for the two parent graphs are satisfied. Picking g_1 as A, the subgraph from G_1, and g_2 as B, the subgraph from G_2, we see that the sum of the vertices in the subgraphs is 17, but there is no way to reconnect them into a 4-regular graph without deleting edges.

This counterexample is also quite contrived, but speaks more to the main problem when reconnected subgraphs into a regular graph: having two vertices which need more edges than there are available. This is why the reproduction algorithm as it is currently implemented maintains an ordering on the vertices in the last stage, so as to add an edge to the vertex with the lowest degree. This helps avoid cases like the one in the counterexample.



(a) Subgraph A (b) Subgraph B

(c) Subgraph C

Figure 3: Each subgraph is 4-regular except for one red vertex with degree 2

3.3 Step operator and local search

Necessary for any sort of local search algorithm is the notion of a step function. This defines what the neighbours of an individual are in the problem space. Since every graph in the population needs to be regular, we need the step function to preserve regularity. This means that simply adding or removing an edge will not suffice. Instead we have a slightly more complicated step function, shown in figure 4: find two pairs of adjacent vertices $(a, b), (c, d)$ such that each element of both pairs is disjoint from the other pair. In other words the only edges between these four vertices are ab and cd. Then delete those two edges and add either ac and bd or ad and bc. This preserves the regularity of the graph.

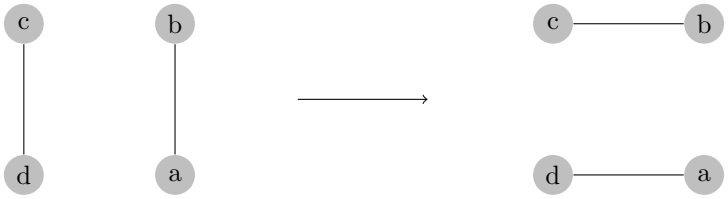


Figure 4: The step function

Now that we have a well-defined notion of locality, we can discuss the choice of local search algorithm. As previously discussed, the problem is that a small change in the graph, like one step in the problem space, affects the evaluation of every single vertex. This means that a single step away from a minimum can increase the evaluation by a large amount. For this reason it is difficult to motivate any sort of heuristic algorithm. On the other hand, the regularity condition for all the graphs in the population means that the step function is less simple. This means that the neighbourhood around an individual is large and not very well defined, so iterating through all of the neighbours is difficult. The problem also has lots of symmetry in its current definition: two graphs could be identical up to renaming of vertices. For these reasons remembering which graphs we have already seen, even just in a neighbourhood, is not realistic, so using something like a Tabu search is not feasible. What is left is local search algorithms like simulated annealing, which is what is used. Simulated annealing is successful at finding local minima quickly, and has seen success with evolutionary algorithms in the past (Philippe Galinier 1999).

4 Results

The results from two runs of the most recent version of the algorithm can be seen in Figure 5. The first point shows the population before any local search has been performed to show some indication of what a completely random regular graph would evaluate to. Both runs were done with population size 50.

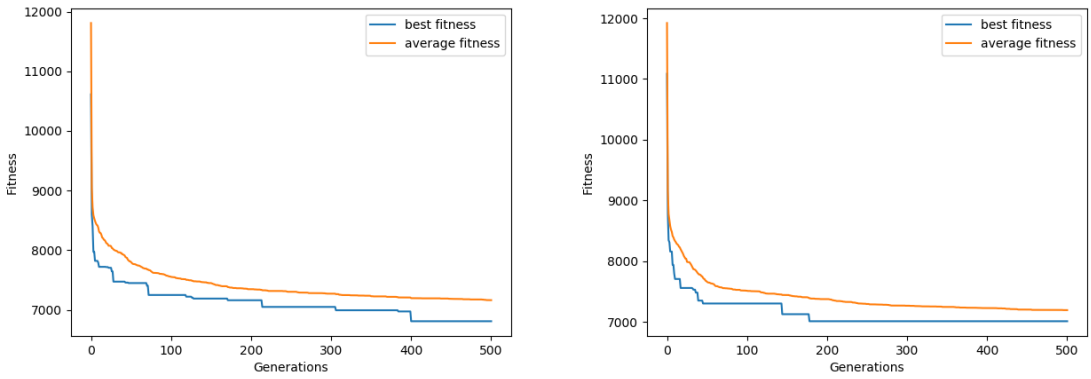


Figure 5: Two runs of the algorithm

The progress reaches a plateau after around 100 generations. The reason for this is likely that every species has reached a significant local minimum, and most offspring cannot reach a similar quality through local search so they do not end up in the population. A simple solution to this would be to accept something into the generation with probability proportional to its evaluation and the number of generations that it has been in the population. This could help preserve some diversity. Some other options will be explored in the next section.

5 Conclusion and Future Work

The approach described in this paper is simple. There is no heuristic or understanding of the structure of the graph instilled in the algorithm, and the only improvement comes from random reproduction or the local search algorithm. We think that there is huge room for improvement in the project, both in terms of the algorithm and the underlying mathematics.

On the mathematics side, there are many techniques for reducing the size of the graph we need to evaluate. The simplest one would be to fill in what necessarily exists in the graph: a single vertex surrounded by 7 triangles, with each of the other vertices adjacent to two vertices among those triangles, and not both on the same triangle. As can be seen in one of the references, this gives us a way to construct the adjacency matrix of the graph and have a system of equations describe its constituents (Brandfonbrener 2017). Similarly we could construct an induced subgraph of the matrix called the star complement. These graphs are much smaller but if found lead to constructions of the graph. In either case we need a way to evaluate fitness based on adjacency matrix equations rather than strong regularity alone. Either of these approaches would drastically reduce the size of the graph and improve search results.

On the other side, the algorithm used is incredibly simple. Modern evolutionary algorithms use genetic codes to represent traits, which results in much more effective crossover and reproduction, while related techniques like NEAT (Kenneth Stanley 2002) are much more complex and effective. Implementing either of these, or some other form of machine learning, would likely yield much better results.

This paper began with a discussion of strongly regular graphs and spectral techniques surrounding them, them discussed the basics of evolutionary algorithms and our approach to solve Conway's 99-problem using one. While the algorithm has been relatively ineffective so far, we think that with improved mathematical foundation and a more modern algorithm it could prove fruitful.

References

Berlekamp, Seidel, Lint. 1973. "A strongly regular graph derived from the perfect ternary Golay code".

Brandfonbrener, David. 2017. "Algebraic Graph Theory, Strongly Regular Graphs, and Conway's 99 Problem".

Conway, John H. 2017. "Five $1000 Problems (Update 2017)".

Keiko Kohmoto, Hiroyuki Narihisa, Kengo Katayama. 2003. "Performance of a Genetic Algorithm for the Graph Partitioning Problem".

Kenneth Stanley, Risto Miikulainen. 2002. "Efficient Evolution of Neural Network Topologies".

Nica, Bogdan. 2018. "A Brief Introduction to Spectral Graph Theory".

Philippe Galinier, Jin-Kao Hao. 1999. "Hybrid evolutionary algorithms for graph coloring".

Sidi Mohamed Douiri, Souad Elbernoussi. 2013. "Solving the graph coloring problem via hybridgenetic algorithms".