Plant Propagation & Hard Hamiltonian Graphs

Joeri Sleegers[1 [0000−0003−1701−6319]]
Daan van den Berg[1 [0000−0001−5060−3342]]

Universiteit van Amsterdam
joeri.sleegers@student.uva.nl, d.vandenberg@@uva.nl

Abstract. Although the Hamiltonian cycle problem is known to be NP-complete, only a few graphs are actually hard to decide for complete backtracking algorithms running on large ensembles of random graphs. Historically, these hard instances are found near the Komlós-Szemerédi bound, the average vertex degree where the Hamiltonian probability phase transition occurs. In this preliminary investigation, we take a different approach, generating hard graphs with two evolutionary algorithms. We find completely new and counterintuitive results.

Keywords: Hamiltonian cycle problem · instance hardness · phase transition · evolutionary algorithms · plant propagation algorithm

1 The Hamiltonian cycle problem

The undirected Hamiltonian cycle problem involves deciding whether a given graph of v vertices and average degree d contains a closed path that visits every vertex exactly once. Known to be NP-complete, quite a few complete algorithms exist for the problem, but none of those runs in subexponential time. The dynamic programming Help-Karp algorithm is quite memory intensive, but by $O(n^2 \cdot 2^n)$ still holds the lowest time complexity [3]. Depth-first based algorithms such as Cheeseman's, Van Horn's, Rubin's, and Vandegriend-Culberson's are far more memory efficient, but take more time in the theoretical worst case: $O(v!)$ [1][4][7][12]. Even for the least sophisticated of these algorithms however, the majority of randomly generated graphs is relatively easily decided. Low-degree graphs require few recursions, so an exhaustive search is quickly completed. High-degree graphs however, contain many Hamiltonian cycles, so one is easily found. The hardest graphs reside in between, right around the Komlós-Szemerédi bound of average degree $v \cdot ln(v) + \ v \cdot ln(ln(v))$ edges, where the probability of a random graph being Hamiltonian goes from almost zero to almost one as d increases [5].

More sophisticated backtracking algorithms such as Vandegriend-Culberson's comb out many of these hard graphs using early-decision techniques, clever pruning and sensible next-vertex heuristics. For this study, we will use a backtracking algorithm that prioritizes low-degree vertices over high-degree vertices, deploys two edge pruning techniques (path pruning and neighbor pruning) and several checks for non-Hamiltonicity (such as degree-one nodes). This algorithm, which

inherits most of its techniques directly from Vandegriend-Culberson's, outperforms all aforementioned backtracking algorithms on large ensembles of randomly generated graphs (results are to be published). For precise algorithmic details, we'll refer the reader to our open-source repository [11].

But even though all these advances reduce the average decision time dramatically, the hardest graphs still reside around the Komlós-Szemerédi bound in these random ensembles. In this preliminary investigation, we take a different approach, 'optimizing' graphs to be as hard as possible for this algorithm. Remarkably enough, the resulting graphs, which are indeed very hard, reside in a totally different part of the combinatorial state space.

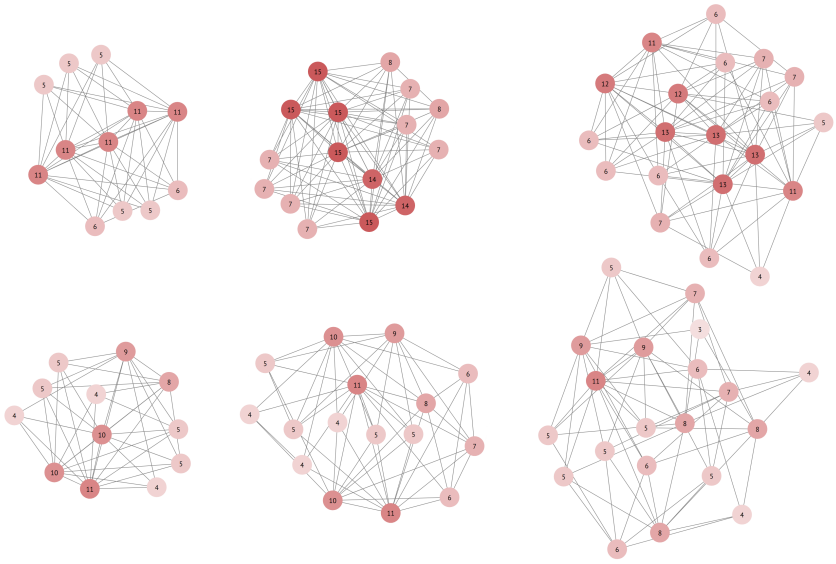

Fig. 1. The hardest Hamiltonian graphs of 12, 16 and 20 vertices found by a hillclimber (top row) and a plant propagation algorithm (bottom row). Graphs appear to converge to a 'hamburger structure', with a 'fat layer' of high degree vertices in the middle, flanked by two 'lighthweight' layers of low degree vertices on either side.

2 Two Evolutionary Algorithms

We use two evolutionary algorithms to make hard graphs: a hillclimber and a plant propagation algorithm [10][8]. For both, the fitness (or objective value) is given by the number of recursions needed by the backtracking algorithm described in Section 1. So the more recursions, the longer the decision time, the harder the graph, and the higher its fitness.

The hillclimber starts off from a random graph of $v = \{12, 16, 20\}$ vertices, and edge degree $d = \lfloor\ v \cdot ln(v) + v \cdot ln(ln(v))\rfloor$. Its inception thereby lies exactly on the Komlós-Szemerédi bound, where we would historically expect the hardest graphs to reside [1][4][12]. Each iteration, one of three possible mutation types is randomly chosen and applied: to **insert** an edge at a random unoccupied place in the graph, to randomly **delete** an existing edge from the graph, and to **move** and edge, which is effectively equal to a delete-mutation followed by an insert-mutation (on a *different* unoccupied place). The mutated graph is kept iff fitter, and the mutation is reverted otherwise. An important observation herefrom is that the graphs do not necessarily retain their initial connectivity throughout the evolutionary process, which is somewhat uncommon in this kind of study. But it's exactly this relaxation that provides us with some surprising results.

The plant propagation algorithm is a population-based evolutionary algorithm that can be applied to a broad spectrum of continuous, discrete and mixed objective landscapes in scientific, industrial and even artistic optimization problems [8][9][2][13][6]. To meet these different requirements, various adaptations have been implemented but the core of the algorithm is always the same: a population of solutions from which fitter individuals spawn many offspring with few mutations, and unfitter individuals spawn few offspring with many mutations, all in an effort to balance the powers of exploration and exploitation in a problem's state space.

The version in this experiment is most closely related to the variants used for optimizing the traveling salesman and timetabling problems [9][2]. Maintaining a constant-sized population of 10 individuals (i.e. undirected graphs) descendingly sorted to fitness (i.e. number of recursions needed to decide the graph), the two fittest individuals each spawn five offspring (i.e. new graphs) which all receive one random mutation. Iff any of these offspring is fitter than its parent, it replaces it. The eight unfitter individuals each spawn one offspring which receives 20 mutations, and again replaces its parent when fitter.

3 Experiment and Results

Both algorithms get 30 random initializations, 10 for each $v \in \{12, 16, 20\}$ and a corresponding $e \in \{15, 23, 31\}$ – exactly on the Komlós-Szemerédi bound – after which they are run for exactly 500 function evaluations of either evolutionary algorithm. These numbers might appear small, but as we are pushing towards the bounds of an NP-complete problem, a single evaluation can easily take up millions of recursions, even for graphs this small.

Both algorithms find hard graphs in all size categories. In most ensembles, more than half of the evolutionary runs generates graphs requiring over 10,000 recursions, sometimes even ranging in the millions (Table 1). The hillclimber appears to outperform plant propagation in finding hard graphs which is in some sense remarkable as it is prone to getting stuck in local maxima. This might indicate that the graph hardness landscape is largely convex, but it is also not unthinkable that in longer runs, PPA would eventually outperform the

hillclimber, as has been witnessed before [2]. What is quite remarkable though, is that all configurations appear to converge to a 'hamburger structure', with approximately 40% 'fat' nodes of high degree being sandwiched between two layers of 30% 'light' nodes of low degree. (Fig. 1).

But what is even more remarkable, is that these graphs all have an edge degree that lies considerably higher than the Komlós-Szemerédi bound where previous investigations by Cheeseman et al., Van Horn et al. and Vandegriend & Culberson found the hardest graphs. There might be several not-so-trivial explanations for this.

4 Discussion

A first explanation for these surprising results is that these results are specific for the backtracking algorithm we used. This is unlikely however, as the algorithm minimizes most other backtracking algorithms found in literature (yet unpublished results). Put differently: chances are very high that these graphs are *also* hard for other complete backtracking algorithms but evidence pending, this possibility can not be completely ruled out.

A second explanation might be that in most studies on Hamiltonian cycle backtracking algorithms, runs are cutoff after a preset number of recursions, as even small graphs can take up significant decision time. However, these cutoff points are usually situated near the Komlós-Szemerédi bound, and not in edge-dense regions where the 'hamburger graphs' would be located.

The most tempting explanation might therefore come from the fact that on first glance, these graphs have low Kolmogorov complexity – they are *structured*. As unstructured objects in any randomly generated ensemble vastly outnumber the structured objects, the chances of being created by a stochastic process (which is the case in most large-scale comparative studies) are microscopic. One would simply not find them unless knowing where to look. These graphs are an isolated island of structured hardness in an ocean of unstructured ease. Whether more such islands exist, and what they look like, awaits further exploration.

Table 1. Both the hillclimber and plant propagation are succesful in finding hard-to-decide graphs within 500 function evaluations on 6x10 graphs of different sizes v. The two last columns are the number of graphs which required over 10,000 recursions, their average degree, and in brackets their *expected* average degree based on the Komlós-Szemerédi bound.

size	**algorithm**	**max recursions**	**avg recursions**	**over 10K**	**avg deg**
12	HC	61,051	36,413	6/10	7.61 (2.56)
12	PPA	12,479	3,395	2/10	6.67 (2.56)
16	HC	198,557,095	19,908,439	6/10	7.84 (2.84)
16	PPA	211,475	43,475	5/10	6.38 (2.84)
20	HC	99,742,171	10,063,317	7/10	6.47 (3.05)
20	PPA	1,130,923	357,725	8/10	6.08 (3.05)

References

1. Cheeseman, P.C., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In: IJCAI. vol. 91, pp. 331–337 (1991)
2. Geleijn, R., van der Meer, M., van der Post, Q., van den Berg, D.: The plant propagation algorithm on timetables: First results. EVO* 2019 p. 2
3. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied mathematics **10**(1), 196–210 (1962)
4. van Horn, G., Olij, R., Sleegers, J., van den Berg, D.: A predictive data analytic for the hardness of hamiltonian cycle problem instances. DATA ANALYTICS 2018 p. 101 (2018)
5. Komlós, J., Szemerédi, E.: Limit distribution for the existence of hamiltonian cycles in a random graph. Discrete Mathematics **43**(1), 55–63 (1983)
6. Paauw, M., Van den Berg, D.: Paintings, polygons and plant propagation. Springer (2019)
7. Rubin, F.: A search procedure for hamilton paths and circuits. Journal of the ACM (JACM) **21**(4), 576–580 (1974)
8. Salhi, A., Fraga, E.S.: Nature-inspired optimisation approaches and the new plant propagation algorithm (2011)
9. Selamoğlu, B.İ., Salhi, A.: The plant propagation algorithm for discrete optimisation: The case of the travelling salesman problem. In: Nature-inspired computation in engineering, pp. 43–61. Springer (2016)
10. Skiena, S.S.: The algorithm design manual: Text, vol. 1. Springer Science & Business Media (1998)
11. Sleegers, J.: Source code. `https://github.com/Joeri1324/What-s-Difficult-About-the-Hamilton-Cycle-Poblem-` (2020)
12. Vandegriend, B., Culberson, J.: The gn, m phase transition is not hard for the hamiltonian cycle problem. Journal of Artificial Intelligence Research **9**, 219–245 (1998)
13. Vrielink, W., van den Berg, D.: Fireworks algorithm versus plant propagation algorithm. In: IJCCI. pp. 101–112 (2019)