SN Computer Science (2022) 3: 372
https://doi.org/10.1007/s42979-022-01256-0

ORIGINAL RESEARCH

The Hardest Hamiltonian Cycle Problem Instances: The Plateau of Yes and the Cliff of No

Joeri Sleegers[1,2,3] · Daan van den Berg[1,2,3]

Received: 14 July 2021 / Accepted: 20 June 2022 / Published online: 15 July 2022

Abstract

We use two evolutionary algorithms to make hard instances of the Hamiltonian cycle problem. Hardness (or 'fitness'), is defined as the number of recursions required by Vandegriend–Culberson, the best known exact backtracking algorithm for the problem. The hardest instances, all non-Hamiltonian, display a high degree of regularity and scalability across graph sizes. These graphs are found multiple times through independent runs, and by both evolutionary algorithms, suggesting the search space might contain monotonic paths towards the global maximum. For Hamiltonian-bound evolution, some hard graphs were found, but convergence is much less consistent. In this extended paper, we survey the neighbourhoods of both the hardest yes- and no-instances produced by the evolutionary algorithms. Results show that the hardest no-instance resides on top of a steep cliff, while the hardest yes-instance turns out to be part of a plateau of 27 equally hard instances. While definitive answers are far away, the results provide a lot of insight in the Hamiltonian cycle problem's state space.

Keywords Hamiltonian cycle problem · Evolutionary algorithms · Plant propagation algorithm · Instance hardness · NP-complete

Introduction

The Hamiltonian cycle problem involves deciding whether an undirected and unweighted graph contains a path that visits every vertex exactly once and returns to the first vertex, closing the loop. In stark contrast to the closely related Euler cycle problem, which is easy, the Hamiltonian cycle problem is notoriously hard, and even has an entry (number 10) on Richard Karp's infamous list of NP-complete problems [21]. Under the common assumption that the classes of P and NP are not equal, the Hamiltonian cycle problem has no subexponential solving algorithm, but candidate solutions can still be verified in polynomial time [8, 17]. NP-complete problems are in some sense 'at the summit of NP': if a polynomial time algorithm is found for just one of these problems, it can be tailored to *all* NP-complete problems, vanishing the class NP completely into P. Unfortunately, such an algorithm is not known for any of the myriad problems in this class, which are therefore all intractable, even at very small instance sizes.

But the exponential runtime increase for solving algorithms[1] is not as crippling as it might appear on first sight. As it turns out, there are substantial differences in instance hardness for many NP-complete problems, and literature on the subject is widely available [6]. One example is graph

This is paper is an extended version of the paper "Looking for the Hardest Hamiltonian Cycle Problem Instances" [36] by invitation following ECTA's 2020 best paper award [13]. It thereby contains the original results as well as an entirely new section on neighbourhoods and a new discussion.

This article is part of the topical collection "Computational Intelligence" guest edited by Kurosh Madani, Kevin Warwick, Juan Julian Merelo, Thomas Bäck and Anna Kononova.

✉ Daan van den Berg
daan@yamasan.nl

Joeri Sleegers
jsleegers@hotmail.com

[1] Mice & Man Software and A.I. Development, Amsterdam, The Netherlands

[2] Yamasan Science & Education, Amsterdam, The Netherlands

[3] Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

[1] Whenever we say 'solvers' or 'solving algorithms', we always mean *exact* or *complete* algorithms, and never their heuristic or non-deterministic counterparts.

colourability, for which Daniel Brélaz' algorithm performs significantly better than the problem's exponential upper bound on many instances [5, 41]. For the satisfiability problem (SAT[2]), which could be considered 'the root of all NP-completeness', the hardness of individual instances, measured in computational effort required for solving, critically depends on the ratio of clauses to variables in the formula [24, 35]. Instances of SAT with many variables and relatively few clauses are generally speaking easy to decide, because they have many solutions. On the other hand, instances with few variables but many clauses are *also* easy to decide, because they can be quickly asserted to be unsolvable. But between these, around the clause-to-variable ratio of $\alpha \approx 4.26$, where a randomly generated formula has $\approx 50\%$ chance of being solvable, is where the hardest instances occur[3] [6, 20]. In this sense, the clause-to-variable ratio α functions as an 'order parameter', or 'predictive data analytic', indicating where to expect the worst runtimes when solving instances of randomly generated SAT-formulas.

For the Hamilton cycle problem, such an order parameter also exists, and it is again related to the solvability of the individual instance. For a randomly generated graph of V vertices and E edges, the probability of being Hamiltonian is given by

$$P_{\text{Hamiltonian}}(V, E) = e^{-e^{-2c}} \tag{1}$$

which is a strictly increasing function on any finite interval, and in which c depends on E and V as

$$E = 1/2\ V \cdot \ln(V) + 1/2\ V \cdot \ln(ln(V)) + cV \tag{2}$$

following the results of János Komlós and Endre Szemerédi [23]. In this equation, $P_{\text{Hamiltonian}}(V, E)$ has its steepest derivative at $c = 0$, where $E = 1/2\ V \cdot ln(V) + 1/2\ V \cdot \ln(\ln(V))$. Although this 'threshold point' happens to be at $e^{-1} \approx 0.368$, rather than a more intuitive 0.5 like in SAT,[4] this 'Komlós–Szemerédi bound' is considered as the central point for the hardest Hamiltonian problem instances. The number of edges (or equivalently: average vertex degree) is consequentially proposed as its order parameter [6, 45]. As a numerical example: for randomly generated undirected graphs of 120 vertices, the hardest Hamiltonian cycle problem instances would occur around 381 edges, where the probability of being Hamiltonian is $\approx 37\%$. Analogously to SAT, far more edges make for much easier problems instances, as dense graphs contain many Hamiltonian cycles, and one is quickly found. Graphs with far fewer edges are also easy, because they can be dismissed as non-Hamiltonian (i.e. not having a Hamiltonian cycle) relatively fast.

In an attempt to find the absolute hardest of the hardest problem instances, evolutionary algorithms were used to generate graphs requiring maximum computational effort for the best known backtracking algorithm [37]. Starting off from the Komlós–Szemerédi bound where traditionally the hardest problem instances were found, a stochastic hillClimber and plant propagation algorithm (PPA) produced graphs that required ever more recursions for the Vandegriend–Culberson (henceforth 'Vacul') algorithm, the most efficient backtracking algorithm for the problem known to date [46].

But NP-completeness had not surrendered all of its surprises yet: the resulting hardest graphs were found nowhere near the Komlós–Szemerédi bound. They were much denser, and displayed a high degree of structural regularity, which might be expressed as low Kolmogorov complexity [25]. The authors hypothesized that exactly for this reason, these very hard problem instances had never shown up in the large randomized ensembles of the aforementioned research endeavours. But the study raised more questions than answers: had the evolutionary algorithms actually converged? Are these almost-regular graphs really the hardest problem instances? In addition, why are they all non-Hamiltonian? What do the hardest *Hamiltonian* graphs look like? In addition, does the number of edges of the initial graph influence the outcome?

In this study, we will answer most of these questions and further deepen our knowledge of the problem. First, we will explain the algorithms involved: Vacul's algorithm for solving problem instances, the stochastic hillClimber and the plant propagation algorithms for evolving graphs. Then, the experiment is described; we significantly extend the scope of graph sizes, runs, starting points, and evaluations. We also conduct a 'Hamiltonian-bound' experiment, in which evolving graphs are *forced* to be Hamiltonian, to see how hard yes-instances can possibly get. Hard, but not nearly as hard as the non-Hamiltonian graphs, as we will shortly see. The paper then reaches an intermezzo, discussing the results so far, and reflecting on explanations for the performance of the evolutionary algorithms. After that, a deeper investigation into the neighbourhood structure of the hardest instances (both yes and no) is presented. The paper then ends with a lot of open ends, but also a treatment on how structure might be related to hardness, and some general implications for benchmarking practices.

[2] Whenever we refer to 'SAT', we implicitly mean random 3CNF-SAT, which is the satisfiability problem in its conjunctive normal form with three randomly chosen literals per clause.

[3] For further refinement on solver performance around the phase transition in SAT, see [1, 7].

[4] The origin of this specific value is that the threshold function becomes ever steeper exactly around e^{-1} as graph size increases, approaching a step function as $V \to \infty$.

Algorithms

Hamiltonian Cycle Problem Solver

Over the last century, a great number of deterministic exact solving algorithms have been developed for the Hamiltonian cycle problem. Rooted in dynamic programming, the Help–Karp algorithm is quite memory intensive, but by $O(n^2 \cdot 2^n)$ still holds the lowest time complexity [19]. Later algorithms by [6, 27, 31, 45, 46] are all exact backtracking algorithms, and therefore have a theoretical upper runtime bound of $O(v!)$, but perform significantly better on large ensembles of random graphs due to clever optimization strategies [38]. Traditionally, the hardest graphs for all these depth-first based algorithms are found around the Komlós–Szemerédi bound, where the probability of a random graph being Hamiltonian goes from almost zero to almost one as E increases (Eqs.1, 2).

Interestingly enough though, all these are applied variations and subsets of just three optimization techniques: vertex degree preference, edge pruning, and non-Hamiltonicity checks. The more the better, it seems, as Vacul's algorithm, containing all three techniques, significantly outperforms all the others—even though its hardest instances are *still* near the Komlós–Szemerédi bound [38, 46]. It is this algorithm, the best backtracker available, that we use for measuring the hardness of Hamiltonian cycle problem instances in this study.

Vacul's algorithm is a depth-first search algorithm that uses edge pruning, non-Hamiltonicity checks and employs a low-degree first ordering while recursing over the vertices. Techniques for edge pruning and non-Hamiltonicity checks are employed both before and during recursion. The pruning subroutine removes edges that cannot be in any Hamiltonian cycle, based on 'required edges' that *must* be in a Hamilton cycle, given that a problem instance has one. An edge is required if it is connected to a vertex with degree two. The algorithm then uses two pruning methods; the first method seeks out vertices that have a degree higher than two and are connected to two required edges, rendering all other edges removable. The second method looks for paths of required edges that do not (yet) form a Hamilton cycle. If an edge exists that would close such a path prematurely, it is removed ('pruned').

The checks for non-Hamiltonicity examine whether the graph cannot contain a Hamilton cycle based on two global properties: having a vertex with degree smaller than two, or the graph being disconnected. Third, the algorithm checks whether the graph is 1-connected, using Tarjan's algorithm [40]. If any of these three conditions are met, the graph cannot be Hamiltonian and the recursive process can be skipped or be backtracked upon.

In these routines for checking Hamiltonicity, edge pruning and vertex degree preference, Vacul's algorithm combines many if not all best practices that have been developed for the recursive class of exact algorithms for the Hamiltonian cycle problem. For a more in-depth treatment, please consult [38].

Evolutionary Algorithms

Though making hard problem instances with evolutionary algorithms is not entirely new, it has become a lot easier during the last decade due to the enormous surge in computational power. This is necessary not because the evolutionary algorithms themselves consume so much budget, but as it finds harder and harder instances, its evaluation function, which is often an exact algorithm for solving the instances, does tend to get close to its dire upper bound ... for *every* function evaluation. A noteworthy example of such initiative is the work by Krzysztof Michalak, who evolved hard instances of the inventory routing problem, [28], while some earlier endeavours addressed TSP, SAT, and the binary constraint satisfaction problem [39, 44].

The evolutionary algorithms used for making the hard Hamiltonian cycle problem instances in this study are a stochastic hillClimber and an implementation of the plant propagation algorithm (PPA), a crossoverless population-based metaheuristic [48, 49]. It has multiple implementations, sometimes substantially different for the seminal form, but our PPA is directly adapted from an earlier application to the travelling salesman problem [18, 34]. By mutating the edge matrix of a graph, both algorithms try to iteratively increase its 'fitness', the computational hardness measured in number of recursions required by the Vacul-algorithm to solve the instance. The more recursions are required, the harder the problem instance, and the fitter the graph.

The evolutionary algorithms use three mutation types with equal probability: to *insert* an edge at a random unoccupied place in the graph, to randomly *remove* an existing edge from the graph, and to *move* an edge, which is effectively equal to a remove mutation followed by an insert mutation (on a *different* unoccupied place). In the hillClimber algorithm, one mutation is chosen at random after which the graph is reevaluated. The mutation is reverted iff the resulting graph is unfitter than its parent, and kept otherwise. This process is repeated for a predetermined number of evaluations (or 'iterations', for this algorithm).

The plant propagation algorithm is a population-based algorithm that tries to balance exploration and exploitation by letting the fitter individuals in the population produce many offspring with few mutations, and unfitter individuals in the population produce few offspring with many mutations. It can be applied to a broad spectrum of continuous,

discrete and mixed objective landscapes in scientific, industrial and even artistic optimization problems [12, 14–16, 18, 30, 33, 34, 47]. A most recent investigation suggested that one version of the algorithm might be largely parameter independent [10, 11].

The implementation of the plant propagation algorithm used in this experiment is closely related to a discrete adaptation that was earlier applied to the travelling salesman problem and the university timetabling problem [18, 34]. Each generation, the population is sorted on fitness after which each individual produces offspring by first copying itself, and then applying a number of mutations to the offspring. If any of a parent's offspring is fitter, it replaces the parent; if multiple offspring are fitter, the *fittest* replaces the parent. The exact numbers of offspring and mutations are predetermined for all ranks in the sorted population (see Table 1). So in this study, the population size is 10 and therefore the number of evaluated offspring is 25 in every PPA generation. These parameters are chosen intuitively, as they abide strongly by PPA's philosophy of balancing the powers of exploration and exploitation, but more efficient parameter settings are certainly not unthinkable. The algorithm's source code can be accessed through a public GitHub repository.[5]

Experiment

To obtain the hardest Hamiltonian cycle problem instances, we evolve 560 graphs of sizes $8 \leq V \leq 14$ in 560 evolutionary runs of 3000 evaluations. The investigation is split in two parts: an 'unbound' experiment, in which the evolutionary algorithms are free to modify all the edges, and a 'Hamiltonian-bound' experiment in which the evolutionary algorithms are free to modify all the edges *except* the edges $\{(1,2),(2,3)\cdots(v-1,v),(v,1)\}$, thereby enforcing the presence of a Hamiltonian cycle in the graph at all times.

For the hillClimber runs, 20 randomly generated graphs were evenly dispersed in terms of edge density, ranging from 0 to $1/2V \cdot (V-1)$ edges, corresponding to edge densities $\in \{0\%, 5\%, 10\% \cdots 95\%\}$. For the PPA runs, twenty initial populations were made along the same edge density intervals, with all graphs in one population having the same edge density. It should be noted that these densities are fixed only upon initialization, as the evolutionary algorithms are free to insert and remove edges from graphs at every step of a run. The rationale behind this choice of edge densities is that earlier results *could* have been biased from the initialization on the Komlós–Szemerédi bound. The current approach would cover a much wider area of the state space, at least as seen from the initial conditions. But for the results, it did not make much of a difference.

From these evenly distributed initial positions, both algorithms ran 3000 function evaluations. This means 3000 iterations for the stochastic hillClimber, but 120 generations of PPA, which produces 25 offspring, and therefore performs exactly 25 evaluations per generation (Table 1). These numbers might look small, as do the numbers of vertices in the graphs used, but the number of recursions required for Vacul's solving algorithm *in every function evaluation* can still easily run in the millions (see Fig. 3). In addition, as we are actively pushing towards the maximum, the entire unbound experiment of 280 runs (7 graph sizes with 20 starting points for two evolutionary algorithms) with 3000 function evaluations still took approximately 45 days on 16 cores of the LISA cluster computer at Amsterdam's Science Park.[6] The 280 runs of 3000 evaluations for the Hamiltonian-bound experiment took significantly less time, possibly because the Hamiltonian-bound instances require significantly fewer recursions to decide. Hamiltonian instances are *easier*, generally speaking. But if you want to see for yourself, all the experiment's resources are publicly available through an open repository.[7]

Results

Unbound Experiment

The results of the unbound experiment resoundingly suggest that the hardest problem instances are all non-Hamiltonian. Both evolutionary algorithms produced structurally similar graphs consisting of a 'clique' and a 'wall' for all vertex numbers (see Fig. 1). The clique is a fully connected subset consisting of $V_{c(\text{odd})} = \frac{V-1}{2}$ vertices in odd-sized graphs, and $V_{c(\text{even})} = \frac{V-2}{2}$ vertices in even-sized graphs. Every graph of size V is a subgraph of size $V+1$, even though the exact addition of edges differs from odd to even graphs. The edge number of these graphs is consequently given by

$$(V - V_c) \cdot V_c + 1/2V_c \cdot (V_c - 1) \tag{3}$$

with $V_c = V_{c(odd)}$ if V is odd, and

$$(V - V_c) \cdot V_c + 1/2V_c \cdot (V_c - 1) + 1 \tag{4}$$

with $V_c = V_{c(even)}$ if V is even. These quadratic results suggest that the larger the graph, the further away the hardest instances are from the Komlós–Szemerédi bound, which only increases (double) logarithmically in V. It should be

[5] https://github.com/Joeri1324/evolving-hard-hamilton-cycles.

[6] https://userinfo.surfsara.nl/systems/lisa.

[7] https://github.com/Joeri1324/evolving-hard-hamilton-cycles.

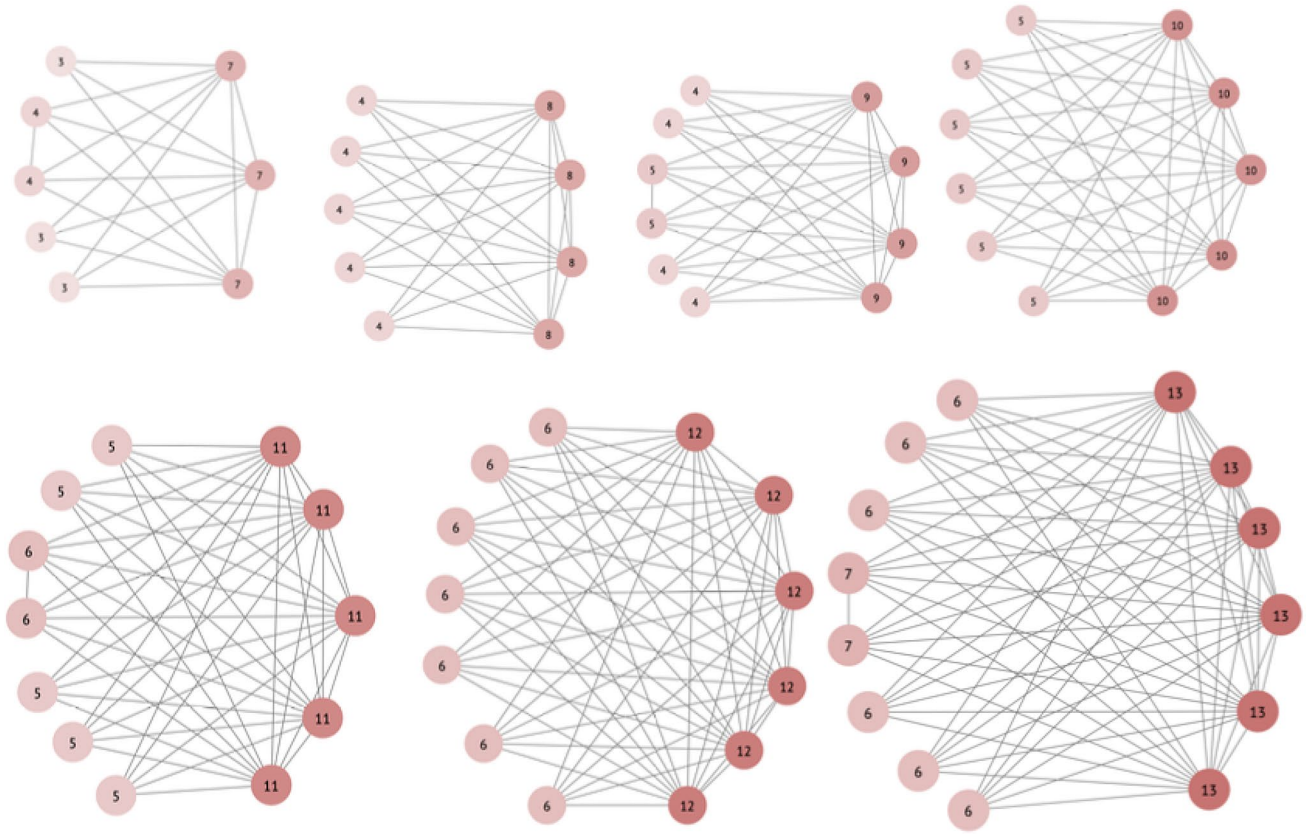

Fig. 1 The hardest instances of the Hamiltonian cycle problem are all non-Hamiltonian, highly structured, and maximally dense. Graphs were found with evolutionary algorithms, and the fitness measured in recursions needed for the Vandegriend–Culberson algorithm, the most efficient backtracker available. The dominant configuration of a 'wall' and a fully connected 'clique' was reached multiple times in independent runs and by both algorithms

noted that these results strongly contradict earlier findings that find the hardest instances close to the bound [6, 46]. This might be due to the random nature of earlier test sets, but for the related SAT-problem, two studies led by Moshe Vardi suggest that the hardness peak *itself* might also move, depending on the specific solver used in the experiments [1, 7].

Fitting an exponential curve through the recursions in Fig. 3 gives functional increase in computational cost of approximately $0.24 \cdot 3.22^V$ ($R^2 = 0.99$) in the number of vertices for the unbound experiment. The base number of 3.22 appears a bit high, even for an NP-complete problem, but significantly lower than the 'plain' complexity $O(V!)$ of an exhaustive enumeration. The number of recursions wobbles a bit in V, which is likely due to discrepancy between odd- and even-sized graphs. In the odd-sized graphs, there are slightly more vertices in the clique, which results in a higher edge density, and possibly more required recursions.

In the unbound experiment, both the plant propagation algorithm and the stochastic hillClimber converged multiple times onto the same graph. HillClimber produced the same instance between 4 and 16 times (11.3 on average) out of 20 for different V (see the bars in Fig. 3, left). In its operation, the stochastic hillClimber is prone to get stuck in local maxima, but the plant propagation algorithm is better equipped for navigating large non-convex search spaces with its highly mutative offspring at the bottom of its population. Maybe that is why the algorithm did solidly better, with all values between 11 and 18 same instances (14.9 on average) out of 20 runs converging to the (same) wall-and-clique graph for different V. It should be noted though, that PPA only outperforms the hillClimber after approximately 2000 evaluations, an effect that was also witnessed in other problems [18]. Because of the consistent convergence through independent runs of both algorithms, and PPA's ability to escape from local maxima, it is possible that the wall-and-clique graphs are indeed the hardest instances of the Hamiltonian cycle problem for Vacul's algorithm. Moreover, this maximum appears to be connected through a state path of monotonically increasing fitness values, the details of which will receive further investigation in "Larger neighbourhoods". A last slightly eyebrow raising observation is that both algorithms converge somewhat better for even numbers of V. Reasons for this, if any, remain unknown.

One-Bit Neighbourhoods

The highly structured results of the unbound experiment allow for an exhaustive mapping of the one-bit neighbourhood of the most difficult instances.[8] For the odd-sized graphs, there is only one possible graph type resulting from edge insertion. For the even-sized graphs there are two neighbouring graph types from inserting an edge, due to the extra edge in the wall. Both these edge insertions immediately make the graph Hamiltonian and very easily decidable, within just V recursions (Table 2). It is a remarkable finding, that the hardest non-Hamiltonian instances and the easiest Hamiltonian instances are separated by just one bit of information.

In odd-sized graphs, removal of an edge can create two different one-bit neighbouring non-Hamiltonian graph types, either from removal inside the clique, or removal of a wall-clique edge. In even-sized graphs, a third and a fourth removal are possible, from the single wall-wall edge, and from the bridge to the clique. All edge-removal operations lower the number of recursions needed to decide the graph, but the effect is much less dramatic than for inserting edges. Even though the number of recursions from edge removal drops between 6 and 63% for the smallest instances, the difference is only between 5 and 33% for the largest instance in this study, and is expected to become ever smaller for larger instances, simply because larger graphs have more edges, so the removal of one could have a smaller impact on recursion.

These neighbourhood results do show however, that the wall-and-clique graphs are at the very least a local maximum of instance hardness. But since both algorithms repeatedly and independently converged to the same graph, and PPA is not sensitive to local maxima, it might well be that these graphs are the hardest instances of the Hamiltonian cycle problem (for Vacul's algorithm). These results could be taken as a suggestion that harder problem instances for the Vandegriend–Culberson algorithm do not exist.

Hamiltonian-Bound Experiment

For the Hamiltonian-bound experiment, results are much less uniform than for the unbound experiment.[9] The hardest Hamiltonian graphs found by the evolutionary algorithms are still roughly a magnitude easier than the non-Hamiltonian graphs (see Fig. 3, right), with the number of recursions increasing as approximately $1.99 \cdot 10^{-7} \cdot 6.90^{V}$ in the number of vertices ($R^2 = 0.99$). Again, this exponent is a fit on only seven data points, needs future refinement, but still serves as a rough indication. The acute reader will notice the unlikeliness that the Hamiltonian exponent actually exceeds the non-Hamiltonian exponent, even if accompanied by a very small multiplicative factor but for now, these are the facts. The authors consider it well possible though that fits through larger numbers of data points give different exponents and factors.

The structural resemblance between graphs of different sizes is also much lower (Fig. 2). For graphs of size $V = 8$, the maximum number of recursions was identical in two graphs, reached in 10 out of 40 runs. For $V = 9$, only 7 out of 40 runs reached any of 4 graphs with maximum recursions, and for larger V, the hardest Hamiltonian instance was unique throughout 40, with just a single PPA run producing that graph. These results suggest that the hardest possible Hamiltonian instances might not yet have been found, and that harder graphs are still possible. An extensive neighbourhood mapping was made for this graph, the results will be presented in "Larger neighbourhoods".

Intermezzo

If the problem instances found in the unbound experiment are indeed global maxima, it could indicate that the problem space is largely convex, since the stochastic hillClimber acquires similar results to the PPA. In this sense, the wall-and-clique graph might be sitting on the top of mount hardness, with very easy Hamiltonian instances and very hard non-Hamiltonian instances in its immediate vicinity.

For the Hamiltonian-bound experiment, such observations are less expedient, because convergence of the algorithms appears much less convincing. So what makes these algorithms perform so bad on the Hamiltonian-bound problem instances? Surely, less freedom from fixing immutable edges would make a problem easier, right? The converse might actually be true, and the argument is a somewhat bewildering and counterintuitive numerical elaboration emanating from Komlós and Szemerédi's early results and some basic combinatorics.

As presented in Eq. 1, the probability of a random graph being Hamiltonian sigmoidally depends on the number of edges. But for a complete edge-independent search space such as ours, this probability might also be seen as a *frequency*. As a numerical example: for $V = 8$ and $E = 14$, Komlós and Szemerédi's equations predict an approximate 61% chance of Hamiltonicity. Equivalently, one could say that 61% of all existable graphs with $V = 8$ and $E = 14$ are Hamiltonian. Now the *number* of graphs is equivalent to the number of options for placing the E edges between V vertices:

[8] Some of the results from this section have been expanded in "Larger neighbourhoods".

[9] Some of the results from this section (too) have been expanded in "Larger neighbourhoods".

Fig. 2 The hardest yes-instances of the Hamiltonian cycle problem (these forcibly *do* contain a Hamiltonian cycle). Structure is much less obvious than for the non-Hamiltonian instances, although some premature tendencies towards cliquing might be discerned

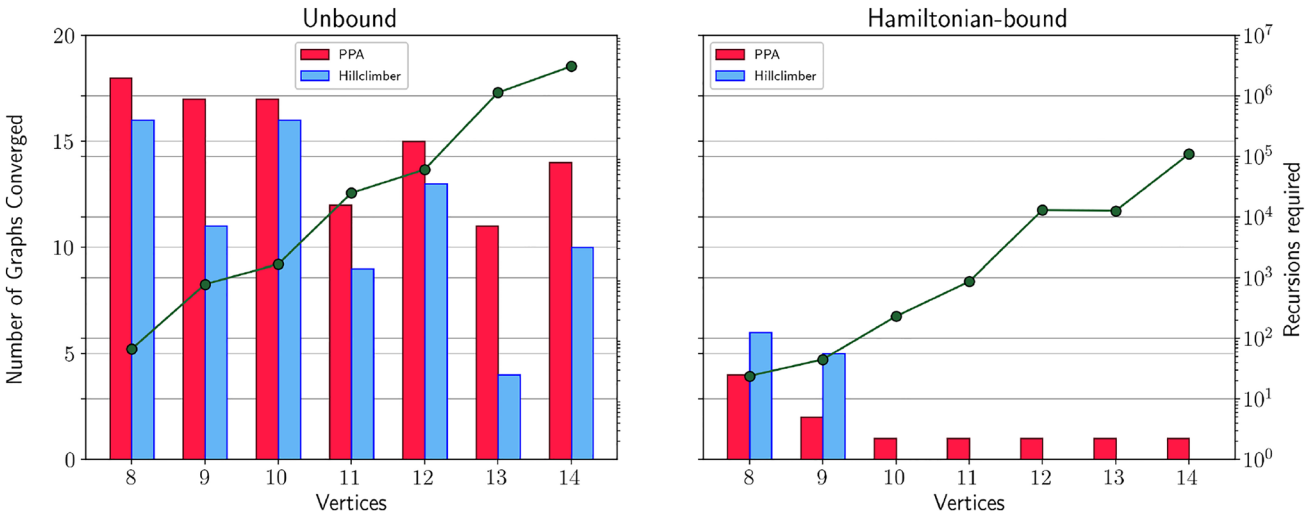

Fig. 3 Recursions required for the hardest graph on the right-side vertical axis versus their corresponding graph size on the horizontal axis. The left graph shows results of the experiment without restrictions on edge mutation, the right graph shows the results of the experiment in which graphs forcibly retained an immutable Hamiltonian cycle at all times. The bars represent the number of multiple times a graph of maximum recursions was found

Table 1 The number and mutability of offspring produced by PPA's individuals are based on its fitness rank (1 = fittest)

Rank	1	2	3	4	5	6–10
# offspring	6	5	4	3	2	1
# mutations	1	2	5	5	10	20

Table 2 The smallest distance between hardest and easiest problem instances for the Hamiltonian cycle problem is (in at least one place) just one bit: inserting an edge on either of the two possible insertion point types makes the hardest (non-Hamiltonian) instance trivially Hamiltonian

Graph size	8	9	10	11	12	13	14
Most difficult	67	785	1673	25,061	61,051	1,139,785	3,091,141
Insert wall #1	8	9	10	11	12	13	14
Insert wall #2	8	9	10	11	12	13	14
Remove clique-clique	63	717	1577	23,261	57,799	1,071,037	2,943,549
Remove wall-wall	43	–	1081	–	39,591	–	2,016,877
Remove wall-clique	25	529	1015	18,561	43,513	894,861	2,387,791
Remove bridge-clique	49	–	1267	–	47,655	–	2,478,947

Removing an edge from either of the four possible types will make for a (just slightly) easier non-Hamiltonian instance. Only six different one-bitflip operations are possible, due to the highly structured nature of the results. Instance hardness is measured in number of recursions required by Vacul's algorithm

Table 3 The edge-independent search space increases faster than exponential in the number of vertices, but the percentage of Hamiltonian instances increases also

Vertices	8	9	10	11	12	13	14
Graphs	$2.68 \cdot 10^8$	$6.87 \cdot 10^{10}$	$3.52 \cdot 10^{13}$	$3.60 \cdot 10^{16}$	$7.38 \cdot 10^{19}$	$3.02 \cdot 10^{23}$	$2.48 \cdot 10^{27}$
Hamiltonian	57.9%	66.5%	74.4%	81.0%	86.3%	90.4%	93.4%

This results in an ever denser volume of Hamiltonian graphs, which might explain a possible lack of convergence in the evolutionary algorithms for the Hamiltonian-bound experiment. Numbers are rounded

$$\binom{1/2 \cdot V \cdot (V-1)}{E} \tag{5}$$

which for $V = 8$ and $E = 14$, amounts to 40,116,600 graphs. Of these, approximately 61%, or 24,274,846 graphs are Hamiltonian, the remaining 39%, or 15,841,754 graphs, are non-Hamiltonian. Summing these results over all possible values of E for a given V gives us the number (or percentage) of Hamiltonian graphs in the entire edge-independent search space (Table 3).

As it turns out, the number of Hamiltonian instances ever more outweigh the number of non-Hamiltonian instances as graphs get larger. So by forcing the evolutionary algorithms into the Hamiltonian part of the combinatorial state space, we actually make it harder to navigate landscapes for increasing V, as all runs have identical numbers of evaluations. This observation might also account for the slightly diminishing returns as V increases, for both algorithms in both experiments but contrarily, these numbers do not account for graph isomorphism. It is an interesting and non-trivial question to see whether other (meta)heuristic algorithms such as a properly parameterized simulated annealing [9, 22] or genetic algorithms [2] do better for this problem. It is also plausible that metaheuristic parameter tuning and/or control might set some serious sods to the dyke, as the problem space clearly changes rapidly as V increases.

On a final note, these graphs might be difficult for Vacul's solving algorithm because its efficiency heavily depends on pruning off edges that cannot be in any Hamilton cycle, which only occurs when a vertex is connected by two required edges. Because of the compact structure of the wall-and-clique graph, this will only happen near the full depth of the search tree, when all but two vertices of the maximum clique are included in a partial solution. But just the ubiquity of pruning techniques throughout history does not spell much good for other exact algorithms either when it comes to these graphs. The non-Hamiltonian instances in this study might thereby actually be the hardest around, but more evidence, or perhaps even a proof, is needed to solidify this conjecture. One way to move forward is to have a look at larger neighbourhoods.

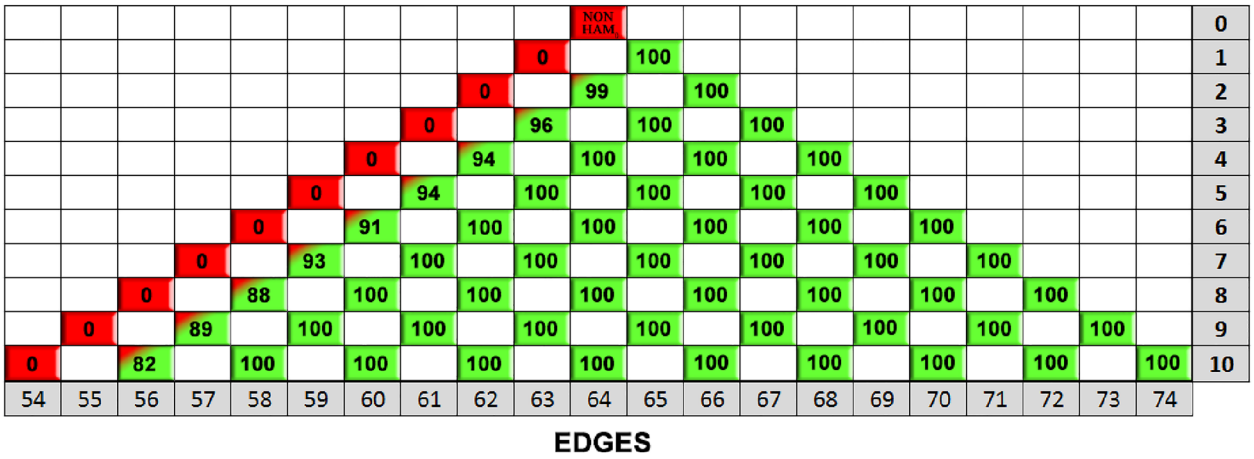

Fig. 4 In the neighbourhood of the hardest non-Hamiltonian graph ($nonHam_0$, on top), nearly all instances are Hamiltonian. Neighbourhoods produced by only removing edges are non-Hamiltonian (primary diagonal, completely red), as is a very small fraction of neighbours which also received 1 edge insert (secondary diagonal, slightly red). Numbers on the neighbourhood blocks are Hamiltonian instances, out of 100, in the generated neighbourhood. The vertical axis has the number of bitflips to produce a neighbourhood, the horizontal axis has the number of edges of graphs therein. For example: right above the central '64' is a cohort of 100 Hamiltonian graphs, made by 10 bitflips, 5 random edge inserts and 5 edge random removes from $nonHam_0$—because it has the same number of edges

Larger Neighbourhoods

These graphs found by the evolutionary algorithms provide us some food for thought. Clearly, the hardest no-instances of the problem (for Vacul's algorithm) are extremely edge dense, and found far beyond the Komlós–Szemerédi bound, in a region where they are theoretically least likely to exist. The situation with the hardest yes-instances though, is much less conclusive as convergence is much less uniform, possibly due to the vast majority of Hamiltonian graphs over non-Hamiltonian graphs within the combinatorial state space. Nonetheless, the discovered Hamiltonian graphs are also situated far beyond the Komlós–Szemerédi bound.

In an extension to these results, we set out to explore the neighbourhoods of these hardness optima, but by a different approach for each instance type. For the largest ($V = 14$) non-Hamiltonian instance, which we will call '$nonHam_0$' (see Fig. 1, bottom-right subfigure) we stochastically sample neighbourhoods with Hamming distances between 1 to 10 bits. Neighbourhood exploration on the hardest (again $V = 14$) Hamiltonian instance (see Fig. 2, bottom-right subfigure) requires a different approach, in recursively making all 1-bit neighbours of the instance at hand. Looking forward, these differences stem from the structural properties of the optima for both instance types: the hardest no-instances form a *cliff* with a narrowing ascending path of 1-bit mutations leading to a single pinnacle with an extremely hard instance, whereas the hardest yes-instance turns out to be just one of 27 equally hard graphs, that are all reachable by making 1-bit mutations, thereby forming a sizeable *plateau*. The forthcoming subsections, which were produced as a later extension to the earlier results, encompass a complete structural treatment of both hardness-neighbourhoods and the methods deployed to discover them.

The Hardest 'No'-Instances

A bitwise mutation in the Hamiltonian cycle problem is the smallest mutation one can make, and it comes in the form of either an edge-insert or an edge-remove (to 'move' an edge, such as done by our PPA-algorithm, is equivalent to simultaneously removing an edge and inserting another, thereby creating a graph with maximum Hamming distance two, and is therefore omitted in the neighbourhood sampling). To explore the neighbourhood of the hardest non-Hamiltonian instance $nonHam_0$, we create 100 new instances for every possible combination of between 1 and 10 mutations, either insert or remove, thereby sampling all neighbourhoods of Hamming distance 1 through 10.

There are 65 such mutation sequences, like 'remove 10 edges' through 'remove 3 and insert 2 edges' and 'remove 3 and insert 6 edges' all the way to 'insert 10 edges', all creating one 100-instance sampled neighbourhood from $nonHam_0$. The maximum Hamming distance from $nonHam_0$ for any 100-instance neighbourhood we created is thereby 10. There are 11 such maximum-distance neighbourhoods (bottom row of Fig. 4), ranging from '10 edge removals', which creates a purely non-Hamiltonian neighbourhood, through neighbourhoods like '7 removals and 3 inserts' or '1 removal and 9 inserts' which both yield 100% Hamiltonian

neighbourhoods. There is an asymmetry in this method, as odd mutation numbers can only produce odd Hamming distances and odd edge number shifts, which reflects in the chequeredness of Fig. 4. Furthermore, the resulting graphs are not uniformly distributed over the various connectivities; there is only one way of making graph neighbourhoods with 10 edges less than $nonHam_0$: by 10 removal mutations. But there are five ways of making graph neighbourhoods with 2 edges more than $nonHam_0$, ranging from 2 insert mutations, to 6 inserts and 4 removals. Finally, mutations were *exclusive*, in the sense that the same edge can not be removed and inserted by the same series of mutations. In other words: when m mutations are made, m distinct locations in the edge matrix are flipped, and the Hamming distances of all graphs inside that neighbourhood is exactly m, and no less.

The results of this experiment are quite uniform: of the 6500 new graphs we made out of $nonHam_0$, the vast majority was Hamiltonian (Fig. 4). More particular, not a single non-Hamiltonian graph was found in areas with higher edge density than $nonHam_0$. We know they must exist though; one example is a fully connected $V-1$ clique and a loose vertex, or a connected vertex with degree one. Such graph types however, are not obtainable from $nonHam_0$ within 10 mutations and are not Vacul-hard, but they exist. Different non-Hamiltonian graph types might also exist in this region of higher density, and it is technically speaking even possible that such graphs are *harder* than $nonHam_0$, but considering the strongly coherent convergence results in Fig. 3, this seems highly unlikely.

In regions sparser than or equally sparse to $nonHam_0$, again almost every generated graph was Hamiltonian. In fact, the only non-Hamiltonian instances were found in two places. The upper left 'primary' diagonal, completely red in Fig. 4, consisting of neighbourhoods made by between 1 and 10 removal mutations, and the secondary diagonal directly below it (slightly reddish), which consists of neighbourhoods made by 1 insert-mutation, and between 1 and 9 removal mutations. We will further explore the top diagonal shortly, but it is important to note that these results are closely related to Komlós and Szemerédi's theoretical results for Hamiltonicity in random graphs. In the span of these neighbourhoods, where $V = 14$ and $54 \leq E \leq 74$, the chance of a random graph being Hamiltonian ranges from 99.39 to 99.97%. So from a theoretical standpoint, it is not a complete surprise that so many graphs in the vicinity of $nonHam_0$ are Hamiltonian. Apparently, the very small fraction of graphs that are non-Hamiltonian are mostly, if not all, structurally very closely related to $nonHam_0$.

Summarizing, these 65 neighbourhoods in the immediate vicinity of $nonHam_0$ contain 6500 graphs, 5426 of which are Hamiltonian and 1074 are non-Hamiltonian. Of these non-Hamiltonians, 1000 were found in the top-left diagonal, the other 74 were found in the secondary diagonal directly below it. Of the 5426 Hamiltonian instances, 5423 were decided in 14 recursions only, which could be accredited either to the efficiency of the Vandegriend–Culberson algorithm, or to the high number of Hamiltonian cycles in these instances. Only 3 Hamiltonian instances required more than 14 recursions, and they were all found in the secondary diagonal: 106 recursions (a graph made from $nonHam_0$ by 1 insert and 7 removes), 2128 recursions and 69,370 recursions (both made from $nonHam_0$ by 1 insert and 6 removes).

The Non-Hamiltonian Cliff

These results show that the hardest non-Hamiltonian instances narrow up along an ascending path of increasing edges density ending in a peak of hardness, much like a Dover cliff. At the pinnacle sits $nonHam_0$, a non-Hamiltonian graph requiring over 3 million recursions, surrounded by very easy Hamiltonian instances on almost all sides. Closely related hard graphs are found on and near the path only, and are almost exclusively non-Hamiltonian instances made by removing one or more edges from $nonHam_0$. A very low number of somewhat hard Hamiltonian instances resulting from a number of removals plus one insert also exist near the cliff path, in the secondary diagonal.

Upon closer inspection of the path itself, its structure shows an intriguing hierarchical pattern of increasing diversity as we descend from the cliff (Fig. 5). The 100 instances emanating from $nonHam_0$ by removal of a single edge are between 4.8 and 34.8% easier than $nonHam_0$, but there are only 4 distinct values in this range, possibly reflecting large isomorphic graph clusters. This would be not completely surprising, as $nonHam_0$ itself is highly regular, and many different edge removal mutations could result in the same (isomorphic) graph. The extent to which isomorphism and hardness equality coincide in the combinatorial state space is as yet unknown.

Descending further down the path by removing more edges, the resulting hardnesses become ever more diverse (Fig. 5). Non-Hamiltonian graphs with 59 edges or fewer have over 90 different hardness values, which are very probably structurally different. Non-Hamiltonian graphs with 57, 55 or 54 edges have 100 different hardness values; 56 edges has 99 different values. These numbers suggest that the path *widens* in instance diversity on decreasing edge numbers. Still, 'widening' should be considered a very relative term. For 14 vertices with 54 edges, $4.26 \cdot 10^{23}$ graphs exist, and even though Komlós and Szemerédi's results predict that only 0.61% of these are non-Hamiltonian, that still accounts for a staggering number of $2.60 \cdot 10^{21}$ graphs. How many of these are isomorphic, and how precise Komlós and Szemerédi's theoretical results are on such a small scale is all open for further analysis, but it surely gives some kind of indication. Besides, a significant number of

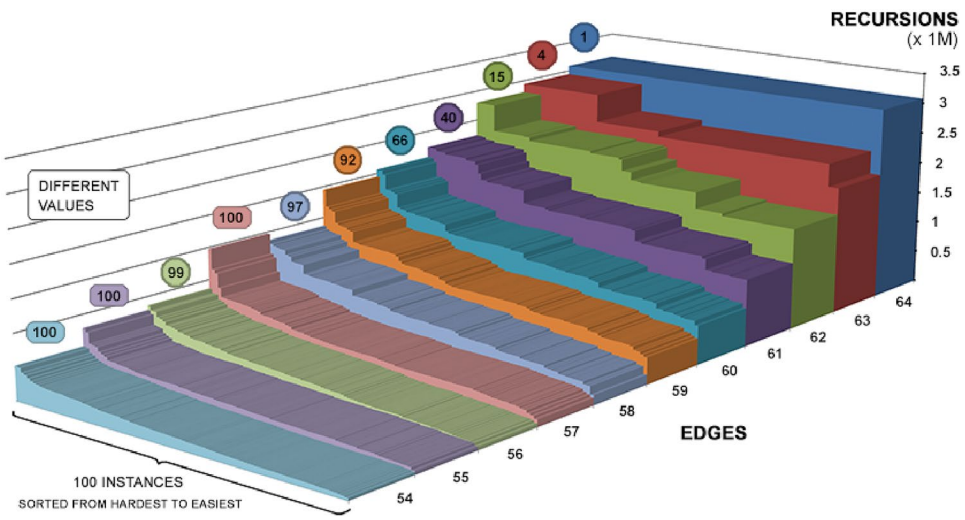

Fig. 5 Except the rightmost, each slice in this figure represents 100 graphs generated by removing 1 or more edges from $nonHam_0$ (which is the rightmost slice). In general, the hardness of these neighbourhoods decreases, while the variety of hardness values widens. Adding an edge nearly anywhere in these graph neighbourhoods however results in very easy Hamiltonian instances. As such, $nonHam_0$ can be thought of as being perched on top of a steep narrow cliff of non-Hamiltonian hardness, surrounded by a deep abyss of easiness

existable non-Hamiltonian graphs in this region is *not hard* for Vacul's algorithm—disconnected graphs, or degree-one graphs, for instance. On the other hand, the convergence of the evolutionary algorithms onto $nonHam_0$ seems to suggest that this *is* truly the hardest instance, but the magnitude of these numbers are too compelling to make any definitive statements as yet.

The Hardest 'Yes'-Instances

For the Hamiltonian instances, the approach was a little different. After all, whereas the hardest non-Hamiltonian graph was quite consistently converged upon by our evolutionary algorithms, the hardest Hamiltonian graph for $V = 14$ (which we named "Ham_0") was found only once in the 40 runs (Fig. 3, right-hand side, rightmost column). Reasons for this could be purely numerical, as discussed in "Intermezzo": the number of yes-instances (or: Hamiltonian graphs) greatly outweighs the no-instances (or non-Hamiltonian graphs) in the combinatorial state space. For $V = 10$ vertices, already $\approx 74.4\%$ of all graphs are Hamiltonian, and this fraction increases as graphs get bigger. Our focus lies on the Ham_0, the hardest Hamiltonian instance of $V = 14$, where the balance of yes- to no-instances is approximately 93.4% versus 6.6%. That is, of all existable undirected graphs of $V = 14$, irrespective of the number of edges, approximately 93.4% is Hamiltonian—a huge portion.

Whether these numbers are *one* explanation or *the* explanation for the poor convergence of the evolutionary algorithms onto Ham_0 is still unclear, but it does prescribe a somewhat different approach in order to get meaningful results from its immediate vicinity. So for the hardest Hamiltonian graph for $V = 14$, we first made all $7 \cdot 13 = 91$ graphs which can be obtained by flipping exactly one bit of information in Ham_0, coinciding with either inserting exactly one edge or a removing exactly one edge. This operation did not yield any harder graphs. But it did yield two different graphs of the *exact same* hardness of Ham_0, being 109,632 recursions for Vacul's algorithm. Both of these were Hamiltonian, and we dubbed them Ham_1 and Ham_2. We progressed by generating all $2 \cdot 91 = 182$ neighbouring graphs for both Ham_1 and Ham_2, which yielded another 4+3=7 equally hard graphs besides Ham_0. Continuing on in similar fashion, we found 160 graphs of hardness equal to Ham_0, all reachable by a single bitflip from earlier found graphs. Considering the importance of this group, we performed a check for isomorphism, which is conveniently provided in an off-the-shelf function from the *networkx* package [29]. After filtering out the isomorphs without damaging the connectivity, an interconnected network of 27 equally hard graphs emerged, all separated by one bitflip from between 4 and 7 neighbours. Such a connected equal-fitness-neighbour-area is appropriately enough called a 'plateau' in evolutionary computing, and a notorious hassle for iteratively improving algorithms such as ours [26, 32]. In the next section, we'll set out to give some structural insights into this plateau. The lookahead conclusion however is: instances Ham_0 through Ham_{26} are currently the hardest known yes-instances for the Hamiltonian cycle problem under Vacul's algorithm for $V = 14$, and they form a 1-bit connected plateauic network of equally hard instances (Fig. 7).

The Hamiltonian Plateau

The plateau of 27 non-isomorphic yes-instances of the Hamiltonian cycle problem for Vacul's algorithm all require 109,632 recursions, making them approximately 28 times easier than the corresponding hardest no-instances for $V = 14$. Although the convergence results in Fig. 3 are somewhat inconclusive, the exponentiality of the trend suggests that this difference might increase for larger V. So while the yes-instances of this decision problem vastly outnumber the no-instances for any

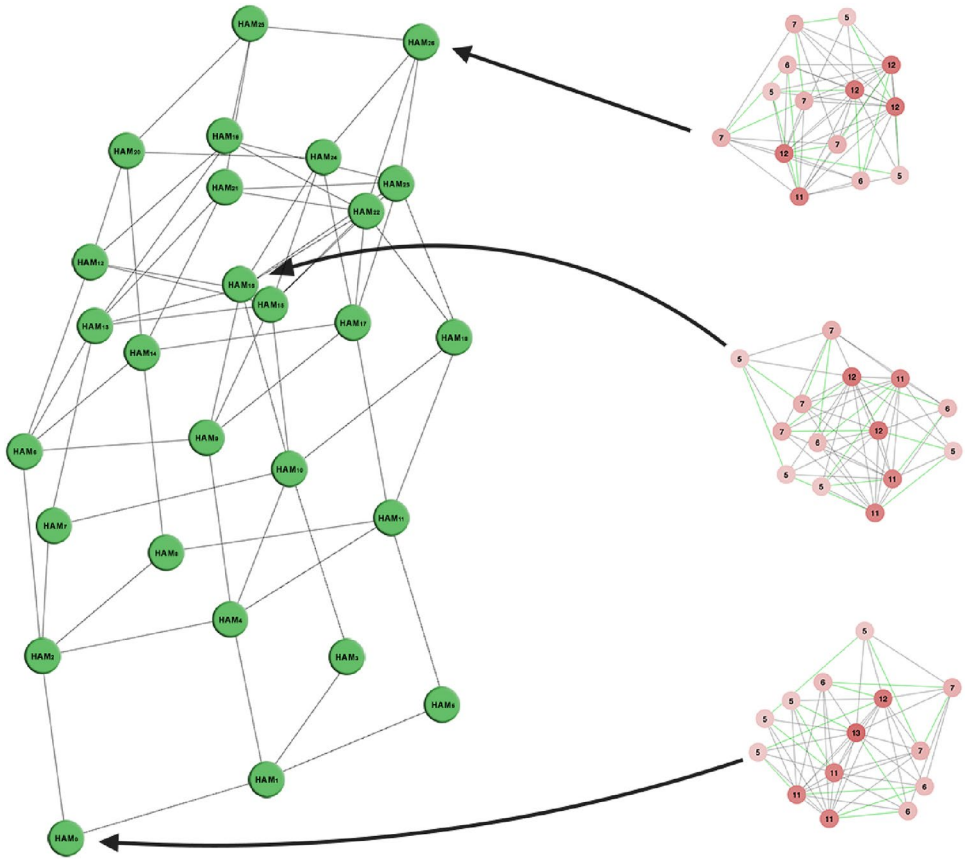

Fig. 6 Left: the 27 hardest yes-instances of the Hamiltonian cycle problem all require 109,632 recursions. When linked by their 1-bit differences, they constitute a plateau with diameter 6 and average path length 2.55. Right: three members of the plateau are shown in detail. Although some resemblance with $nonHam_0$ is apparent, the Hamming distance towards the non-Hamiltonian cliff is still quite significant

reasonable V, the hardest 'no'-instances (non-Hamiltonian graphs) are apparently exponentially harder than the hardest yes-instances; a remarkable observation.

The 27 Hamiltonian graphs on the plateau can be thought of as a network[10] with problem instances embedded in its nodes, and its links symbolizing a 1-bit mutation (edge-insert or edge-removal). This plateau network is connected, and has 64 links between its 27 nodes with connection degrees ranging from 2 to 7. Remarkably enough, Ham_0 itself has only degree 2 and is on the edge of the plateau. This could merely be an unlikely coincidence, as the convergence results were not very consistent for the yes-instances, but it could also be that mutations from lower hardness Hamiltonian instances funnel the algorithm towards Ham_0, in similar fashion to the non-Hamiltonian cliff.

The structure itself has a somewhat three dimensional feel to it (Fig. 6, left-hand side), but to what extent it can actually be mapped onto a mesh is unknown. Every problem instance inside a plateau node is quite dense, having $55 \leq E \leq 59$ edges between its $V = 14$ vertices. On the eye, these hardest yes-instances (Fig. 6, thrice in the right-hand side) indeed look very much like the hardest no-instances , sporting a highly connected core of 5 vertices, and a 'wall' of 9 vertices having degree 5, 6, 7 or 8. Though it is tempting to think that the hardest yes-instances and the hardest no-instances might therefore be closely intertwined, current evidence does not support this idea. Even for the densest yes-instance (59 edges), the Hamming distance to the hardest no-instance (64 edges for $nonHam_0$) is *at least* 5, but likely higher. What can be said though, is that the hardest yes- *and* the hardest no-instance are both located in a very edge dense region, far beyond the Komlós–Szemerédi bound, which was previously thought to hold the hardest instances. Another open issue is that we can not rule out the existence of other (plateaus of) hard(er) yes-instances anywhere in the combinatorial state space.

A seemingly insignificant frolic is that the plateau network *itself* is non-Hamiltonian—the four nodes at the bottom of Fig. 6's left-hand side cannot be in any Hamiltonian cycle together.

Around the Hamiltonian Plateau

At this stage, we can not rule out that other yes-instances of the same or even higher hardness exist, but if they do, they are not connected to this plateau. In the direct 1-bit

[10] We intentionally omit the terms 'graph', 'vertex' and 'edge' here, to distinguish the plateau structure from the structure of the problem instances.

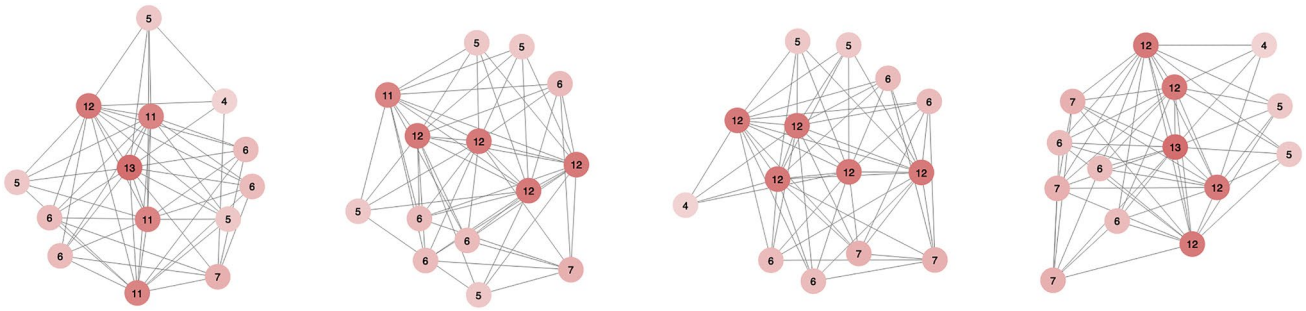

Fig. 7 Directly besides the Hamiltonian plateau lie 24 non-Hamiltonian instances, all of which are much harder than the plateau itself. They come in 4 groups, all with identical edge numbers, diameter, average path length and strongly similar (sometimes even identical) degree distributions. Still, within these 24 graphs, not one pair is isomorphic. Counterintuitively, their minimum Hamming distance towards $nonHam_0$, on the Hamiltonian cliff, is greater than that of the plateau instances

vicinity of the 27-plateau are exactly $27 \cdot 91 = 2430$ graphs that make up the plateau's 1-bit neighbourhood. The vast majority of these, 2406 instances ($\approx 99.0\%$) is Hamiltonian too. Of these 2406 plateau-adjacent Hamiltonian instances, 1507 (61.3 %) graphs are trivially easy, requiring 14 recursions only. In such cases, Vacul's algorithm finds a Hamiltonian cycle almost instantly, without backtracking upon a single branch. The other 1070 plateau-adjacent Hamiltonian instances require between 2914 and 109,632 recursions, with an average and median relatively close to the maximum, (90,457 and 101,504 recursions respectively). There are 27 different values, which may or may not be a coincidence as the plateau itself also consists of 27 nodes. Some of these values occur only once, like 51,644, 51,760 and 58,694 recursions, and some values occur more often, most prominently 109,632 recursions, which was found 161 times—these are all plateau graphs, or isomorphs thereof. The remaining 909 instances come in 26 different recursion values and might therefore also contain some degree of isomorphism, but that has yet to be seen.

A somewhat more puzzling finding is the finding of exactly 24 non-Hamiltonian instances directly connected to the plateau, mainly because *these are all significantly harder than the plateau itself*. The number of required recursions ranges from 229,251 to 718,745 thereby being between 2.09 and 5.56 times as hard as the plateau instances. Moreover, every recursion value is unique, and a check yields that these 24 instances do not contain a single isomorphic pair. Furthermore, these 24 graphs come in only four groups of 6, 6, 6, and 2 instances with $E = 54$ edges and $APL = 1.41$,[11] $E = 55$ edges and $APL = 1.40$, $E = 56$ edges and $APL = 1.38$, and $E = 57$ edges and $APL = 1.37$. Furthermore, many identical degree distributions are to be found, most notably within one group. So the takeaway here is that even with exactly identical values for number of vertices, edges, diameter, APL and even degree distributions, graphs might still be very different in terms of hardness, and definitely not isomorphic.

This find is remarkable, because the plateau-connected Hamiltonian graphs shows strong homogeneity in recursion values, while the adjacent non-Hamiltonian instances show maximum diversity. Although all 24 adjacent non-Hamiltonians are unique, and their recursion numbers vary a lot, they cannot be *wildly* different, because they are all just one bit away from the plateau, which has a diameter of 6, and an APL of only 2.55. In addition, there's another remarkable fact: although the increased hardness and non-Hamiltonicity might suggest that these problem instances might be closer to the non-Hamiltonian cliff, this could very well not be the case. Judging by edge numbers, the minimal Hamming distance from these instances to $nonHam_0$ is between 7 and 10. The Hamiltonian instances on the plateau's Hamming distances to $nonHam_0$ is between 5 and 9.

So although the adjacent non-Hamiltonians' recursion values are closer to the cliff, and they look structurally alike, they are actually further away than the plateau itself. At this point, we have no idea how to interpret these findings (Fig. 7).

Discussion

Is Hardness Related to Structure?

It's time for a redefinition. Clearly, the paradigm that "the hardest instances of the Hamiltonian cycle problem reside around the phase transition (or: Komlós–Szemerédi bound)" is no longer unequivocally true. For at least one algorithm, the highly efficient backtracker by Vandegriend–Culberson, the hardest instances, both Hamiltonian and non-Hamiltonian, are situated in a very edge dense region of the combinatorial

[11] Average path length (APL) is the minimum number of links to traverse from one randomly picked node to another.

state space, far beyond the Komlós–Szemerédi bound. Judging from the generality of the structure, these instances could well be very hard for *all* reasonable backtracking algorithms and all instances sizes, but this is yet to be investigated.

So do these results invalidate Cheeseman et al.'s results then? Not at all. Although the hardest instances, both yes and no, lie very far away from the Komlós–Szemerédi bound, they are probably very rare. By an argument from Kolmogorov complexity, these graphs are to some extent structured, and can be recreated by a small computer program containing some for loops and a few explicit edge definitions. This is very different from unstructured graphs, whose smallest computer program must explicitly contain its entire edge matrix to accurately recreate it. In addition, by a simple counting argument, the number of computer programs that is significantly shorter (in bits) than the number of graphs is extremely small. Therefore, structured graphs must be rare too in the combinatorial state space. In addition, if structure is indeed related to hardness, this also means that the easy problem instances greatly outnumber the hard problem instances.

Highly problematic with these conjectures though, is that they are nearly untestable due to a number of serious obstacles. First, exhaustively enumerating the graphs is not an option for any number of vertices over 20 or so (that's already $\approx 10^{35}$ graphs). Pure random generation or random generation in columns of fixed edge degree is not viable either, because it likely produces relatively easy instances, as hard instances, if indeed related to structure, could well be extremely rare. Evolutionary algorithms provide some solution to finding these, but take a very long time to converge because of the extremely high costs of individual evaluations, the ginormous state space, and the lack of *guarantee* in finding the hardest instances. But thirdly, and this is not unimportant: Kolmogorov complexity is practically incomputable. There is absolutely no way of telling how short the shortest program for any given graph is, or conversely: what short programs are amenable to producing interesting (hard) instances at all. So if structure is indeed related to hardness, an exact approach is almost certainly doomed to fail.

There is a bit of hope for an *a posteriori* approach though: a short program (of say, length b) for the wall-and-clique graph, which likely exists, produces some bound on the complexity, a program for an arbitrary random graphs likely needs around 91 bits (one bit for each vertex pair) for $V = 14$, the difference must be at least $91 - b$, but could well be greater. Assuming that such a program also exists for the Hamiltonian instances, and given we can find it a posteriori, and it is significantly short, then this would not only confirm that structure is related to hardness, but also enable us to *create* hard instances for any number of V, both Hamiltonian and non-Hamiltonian.

Taking this argument one step further, if either enumeration, algorithmics or randomness could account for making a *cloud* of hard yes- or no-instances, it would be interesting to see how those output clouds mingle. Looking at the structures presented in this paper, it is imaginable that at some point, the hardest yes- and no-instances might be close together, both in Hamming distance and in computational hardness (for a similar discussion, see [4, 42]).

Implications for Benchmarking

Another consequence of these results is that we might add a serious consideration to our best benchmarking practices [3]. At the very least, we could ask ourselves the question "what is it that constitutes a good benchmark set?". If "being representative" is considered a requirement, than we should consider whether random generation of instances is a good idea in the first place. Random generation, even random generation in degree columns, has almost zero chance of creating structured instances, which might (partially) coincide with the class of hard instances. For the Hamiltonian cycle problem, but possibly also for other ((harder than) NP-complete) problems, they are simply too rare to show up in randomly generated benchmark sets.

But an immediate deeper question here is: do you *want* to have the hardest instances in your benchmark set, and if yes, *which* hardest instances? For the Hamiltonian cycle problem, generating random graphs in degree columns, a column with a degree near the Komlós–Szemerédi bound may well contain the hardest instances *on average*, but the hardest *single individuals* might be in the edge-dense regions far away—possibly in a column together with a host of very easy instances, considering Cheeseman's results. So a redefinition of "where the *really* hard problem instances are" might incorporate maxima, minima, standard deviations or possibly a complete hardness frequency spectrum of degree columns for better answers.

It is well known that many "real world graphs" have a 'small-world' structure, which translates to relatively low Kolmogorov complexity [43]. Therefore, data on randomly generated benchmark sets might produce absolutely zero useful information for real-world graph applications. Similarly, for finding the hardest possible instances for a class of algorithms, random generation does not seem suitable either. Thankfully, evolutionary algorithms which provide us with a quantum of solace and a way forward. Even though we have few definitive answers yet, there is some indication of direction and as for future work, one of the tasks will be to more sharply define the hardest instances, and if possible, describe them with small computer programs.[12]

[12] ECTA 2020 is part of the larger conference IJCCI 2020, see http://www.ecta.ijcci.org/.

Acknowledgements ECTA's reviewers really made an effort to understand and thoroughly annotate the predecessor of this paper which led to this extended version. For the journal paper, two reviewers of Springer Nature Computer Science did an excellent job by not only reading, but recalculating, interpreting and discussing our findings. Thank you all, for adding your fruits of thought to our pie.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Aguirre ASM, Vardi M. Random 3-sat and bdds: the plot thickens further. In: International conference on principles and practice of constraint programming. Springer; 2001. p. 121–36.
2. Bäck T, Fogel DB, Michalewicz Z. Handbook of evolutionary computation. Release. 1997;97(1):B1.
3. Bartz-Beielstein T, Doerr C, Berg D, Bossek J, Chandrasekaran S, Eftimov T, Fischbach A, Kerschke P, La Cava W, Lopez-Ibanez M et al (2020) Benchmarking in optimization: best practice and open issues. arXiv:2007.03488.
4. Braam F, van den Berg D. Which rectangle sets have perfect packings? Oper Res Perspect. 2022;9: 100211.
5. Brélaz D. New methods to color the vertices of a graph. Commun ACM. 1979;22(4):251–6.
6. Cheeseman P, Kanefsky B, Taylor WM. Where the really hard problems are. In: Proceedings of the 12th international joint conference on artificial intelligence, vol 1, IJCAI'91. San Francisco: Morgan Kaufmann Publishers Inc.;1991. p. 331–37.
7. Coarfa C, Demopoulos DD, Aguirre AS M, Subramanian D, Vardi MY. Random 3-sat: the plot thickens. In: International conference on principles and practice of constraint programming. Springer; 2000. p. 143–59.
8. Cook SA. The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on theory of computing, STOC '71. New York: ACM; 1971. p. 151–58.
9. Dahmani R, Boogmans S, Meijs A, van den Berg D. Paintings-from-polygons: simulated annealing. In: International conference on computational creativity (ICCC'20). 2020.
10. De Jonge M, van den Berg D. Parameter sensitivity patterns in the plant propagation algorithm. In: IJCCI. 2020. p. 92–9.
11. de Jonge M, van den Berg D. Plant propagation parameterization: offspring & population size. In: Evo* 2020. 2020. p. 19 .
12. Dijkzeul D, Brouwer N, Pijning I, Koppenhol L, van den Berg D. Painting with evolutionary algorithms. In: International conference on computational intelligence in music, sound, art and design (Part of EvoStar). Springer; 2022. p. 52–67.
13. ECTA (2020) Ecta website. http://www.ecta.ijcci.org/PreviousAwards.aspx. Accessed 18 June 2021.
14. Fraga E. Fresa: a plant propagation algorithm for black-box single and multiple objective optimization. Int J Eng Tech Inf (Skeena). 2021;2(4):110–1.
15. Fraga ES. An example of multi-objective optimization for dynamic processes. Chem Eng Trans (AIDIC). 2019;74:601–6.
16. Fraga ES. Multiple simultaneous solution representations in a population based evolutionary algorithm. 2021. arXiv:2106.05096. Accessed 21 Feb 2022.
17. Garey MR. Johnson DS. Computers and intractability; a guide to the theory of NP-completeness. New York: W. H. Freeman & Co.; 1990.
18. Geleijn R, van der Meer M, van der Post Q, van den Berg D. The plant propagation algorithm on timetables: first results. In: EVO* 2019. 2019. p. 2.
19. Held M, Karp RM. A dynamic programming approach to sequencing problems. J Soc Ind Appl Math. 1962;10(1):196–210.
20. Hutter F, Xu L, Hoos HH, Leyton-Brown K. Algorithm runtime prediction: methods & evaluation. Artif Intell. 2014;206:79–111.
21. Karp RM. Reducibility among combinatorial problems. In: Miller RE., Thatcher JW, Bohlinger, JD, editors. Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department, Boston. Springer US; 1972.
22. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science. 1983;220(4598):671–80.
23. Komlós J, Szemerédi E. Limit distribution for the existence of Hamiltonian cycles in a random graph. Discrete Math. 1983;43(1):55–63.
24. Larrabee T, Tsuji Y. Evidence for a satisfiability threshold for random 3cnf formulas. Technical report. 1993.
25. Li M, Vitányi P, et al. An introduction to Kolmogorov complexity and its applications, vol 3. Springer; 2008.
26. Malan KM, Engelbrecht AP. A survey of techniques for characterising fitness landscapes and some possible ways forward. Inf Sci. 2013;241:148–63.
27. Martello S. Algorithm 595: an enumerative algorithm for finding Hamiltonian circuits in a directed graph. 1983.
28. Michalak K. Generating hard inventory routing problem instances using evolutionary algorithms. In: Proceedings of the genetic and evolutionary computation conference. 2021. p. 243–51.
29. NetworkX. Python's NetworkX package. 2021. https://networkx.org/. Accessed 18 June 2021.
30. Paauw M. van den Berg D. Paintings, polygons and plant propagation. In: International conference on computational intelligence in music, sound, art and design (Part of EvoStar). Springer; 2019. p. 84–97.
31. Rubin F. A search procedure for Hamilton paths and circuits. J ACM. 1974;21(4):576–80.
32. Russell S. Norvig P. Artificial intelligence: a modern approach. 2002. p. 123.
33. Salhi A. Fraga ES. Nature-inspired optimisation approaches and the new plant propagation algorithm. 2011.
34. 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. Springer; 2016. p. 43–61.
35. Selman B, Mitchell DG, Levesque HJ. Generating hard satisfiability problems. Artif Intell. 1996;81(1):17–29 (**Frontiers in Problem Solving: Phase Transitions and Complexity**).
36. Sleegers J, vanden Berg D. Looking for the hardest Hamiltonian cycle problem instances. 2020.

37. Sleegers J, van den Berg D. Plant propagation & hard Hamiltonian graphs. In: Evo* 2020. 2020. p. 10.
38. Sleegers J, Van den Berg D. Backtracking (the) algorithms on the Hamiltonian cycle problem. Int J Adv Intell Syst. 2021;14:1–13.
39. Smith-Miles K, van Hemert J, Lim XY. Understanding tsp difficulty by learning from evolved instances. In: International conference on learning and intelligent optimization. Springer; 2010. p. 266–280.
40. Tarjan R. Depth-first search and linear graph algorithms. SIAM J Comput. 1972;1(2):146–60.
41. Turner JS. Almost all k-colorable graphs are easy to color. J Algorithms. 1988;9(1):63–82.
42. Van Den Berg D, Adriaans P. Subset sum and the distribution of information. In: Proceedings of the 13th international joint conference on computational intelligence. 2021. p. 135–141.
43. van den Berg D, Gong P, Breakspear M, van Leeuwen C. Fragmentation: loss of global coherence or breakdown of modularity in functional brain architecture? Front Syst Neurosci. 2012;6:20.
44. van Hemert JI. Evolving combinatorial problem instances that are difficult to solve. Evol Comput. 2006;14(4):433–62.
45. van Horn G, Olij R, Sleegers J, van den Berg D. A predictive data analytic for the hardness of Hamiltonian cycle problem instances. In: Data Analytics 2018: the seventh international conference on data analytics. 2018.
46. Vandegriend B. Culberson JC. The gn,m phase transition is not hard for the Hamiltonian cycle problem. arXiv:1105.5443 [CoRR]. 2011. Various sources report reversals of author order. Basil Vandegriend is the first author of this paper.
47. Vrielink W, van den Berg D. Fireworks algorithm versus plant propagation algorithm. 2019.
48. Vrielink W, van den Berg D. A dynamic parameter for the plant propagation algorithm. 2021.
49. Vrielink W, van den Berg D. Parameter control for the plant propagation algorithm. 2021.