 Universally Hard Hamiltonian Cycle Problem Instances

Joeri Sleegers\textsuperscript{1\textcopyright a}, Sarah L. Thomson\textsuperscript{2\textcopyright b} and Daan van Den Berg\textsuperscript{3\textcopyright c}

\textsuperscript{1}Mice & Man Software and A.I. Development, Amsterdam, The Netherlands
\textsuperscript{2}Department of Computing Science & Mathematics, University of Stirling, U.K.
\textsuperscript{3}Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
jsleegers@hotmail.com, s.l.thomson@stir.ac.uk, daan@yamasan.nl

Keywords: Exact Algorithms, Instance Hardness, Evolutionary Algorithms, Phase Transition.

Abstract: In 2021, evolutionary algorithms found the hardest-known yes and no instances for the Hamiltonian cycle problem. These instances, which show regularity patterns, require a very high number of recursions for the best exact backtracking algorithm (Vandegriend-Culberson), but don’t show up in large randomized instance ensembles. In this paper, we will demonstrate that these evolutionarily found instances of the Hamiltonian cycle problem are hard for \textit{all} major backtracking algorithms, not just the Vandegriend-Culberson. We compare performance of these six algorithms on an ensemble of 91,000 randomized instances plus the evolutionarily found instances. These results present a first glance at universal hardness for this NP-complete problem. Algorithms, source code, and input data are all publicly supplied to the community.

1 INTRODUCTION

In computing, some problems are considered inherently difficult, while others are relatively easy. The distinction between the two is in the runtime increase for the most efficient algorithm on a problem. As an example, finding the minimum spanning tree on a graph of \(V\) vertices can be done by Kruskal’s algorithm in at most \(O(\log(V))\) time, concretely meaning that if the worst possible problem instance of \(V = 10\) vertices requires one second by your implementation of Kruskal’s algorithm, the worst possible problem instance of 100 vertices requires two seconds at most. Another example is finding a traveling salesman tour of at most 50\% worse than the optimal solution, which can be done by Christofides-Serdyukov algorithm in \(O(V^3)\) time, concretely meaning that if a solution for \(V = 10\) cities on a map takes one second, then a solution for \(V = 20\) cities takes not two but eight seconds. Both Kruskal’s algorithm and Christofides-Serdyukov are considered polynomial-time algorithms and if a problem has a polynomial-time algorithm, we say ‘the problem is in P’.

But even though \(O(V^3)\) might look like a bad increase compared to \(O(\log(V))\), it is nothing compared to the cost increase of an exponential-time algorithm. For example, an exhaustive depth-first search for finding a true-assignment to a 3CNFSAT Boolean formula like \((\neg a \lor b \lor \neg c) \land (\neg b \lor \neg c \lor d)\) takes \(O(2^n)\) time for \(n\) variables, as does finding the exact 2-part split of set 141,206,391,434,591,668,779,801 (Van Den Berg and Adriaans, 2021). These decision problems, where the size of the input problem instance appears in the exponent, are all harder than all polynomial-time problems, because even a very small exponent will eventually outgrow a very large polynomial as the instance size increases. These decision problems, which have no known deterministic polynomial-time algorithm, are said to be ‘NP-complete’\textsuperscript{1}. On account of these exponential influences, problems in NP (and more specifically for this paper in NP-complete) are considered hard, whereas problems in P are generally considered as easy, though one could easily discursively peregrinate along the details, which we will joyfully do next.

Because as disheartening as NP(-completeness) might seem, not all the instances of an NP-complete problem are equally hard. In fact, when given the right algorithm, only a small fraction of a randomized instance ensemble is actually hard – at least for SAT,

\textsuperscript{1}In fact, an additional requirement for NP-completeness is polynomial-time reduction of another NP-complete problem, which for narrative reasons, is omitted here. See e.g. (Garey and Johnson, 2002).

This paper is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 Licence (CC BY-NC-ND). https://creativecommons.org/licenses/by-nc-nd/4.0/
Figure 1: (Right-hand side, large inset: The hardest no-instance of the Hamiltonian cycle problem, $NonHam_0$, is a single highly regular graph, which is bipartite when $V$ is odd, and nearly so when $V$ is even. Left-hand side, small inset: The hardest yes-instances comprise a set of 27 non-isomorphic graphs ($Ham_0$), which have varying edge densities (shown are $E = \{55, 56, 57, 58, 59\}$). Although these yes-instances also show a high degree of regularity, the minimum Hamming distance from any of these graphs to $NonHam_0$ is 6.

the Hamiltonian cycle problem and graph colouring (Cheeseman et al., 1991). Remarkably enough, these hard instances were all found very close to the phase transition in solvability.

Most if not all NP-complete problems have a solvability phase transition through some ‘order parameter’: a predictive data analytic that bestows an a priori probability of solvability upon the set of all instances of size $n$ for a problem. Satisfiability has one; if the number of clauses in the formula outweigh the number of variables by approximately $\alpha \approx 4.26$, it probably has no solution, whereas lower values of $\alpha$ mean it probably has (Larrabee and Tsuji, 1992; Hayes, 1997). For Hamiltonian cycle problem instances of size $V$, if the number of edges exceeds the value of $\frac{1}{2} V \ln(V) + \frac{1}{2} V \ln(\ln(V))$ then Hamiltonicity is highly likely; for lower values, the instance most likely has no Hamiltonian cycle (Komlós and Szemerédi, 1983)(van Horn et al., 2018). This sudden transition in solvability turned out to be quite ubiquitous for NP-complete problems, or as Ian Gent and Toby Walsh put it: “Indeed, we have yet to find an NP-complete problem that lacks a phase transition” (Gent and Walsh, 1996). But an additional the quality of Hamiltonian cycle problem’s phase transition, is that its phase transition is exactly characterized, both by shape and location, in early work by János Komlós and Endre Szemerédi (Komlós and Szemerédi, 1983).

So why care about these solvability phase transitions? Because that is where the hardest problem instances were found; a famous result by Cheeseman, Kanefsky & Taylor’s (henceforth: ‘Cetal’). For instances close to the solvability phase transition, computational resources abolutely skyrocketed, making NP-completeness live up to its reputation. SAT formulas with 100 variables and 426 clauses were typically hard to solve. Hamiltonian cycle problem instances with 100 vertices and 307 edges were also hard, as were 3-colorability graphs with a connectivity around 5.4 (Cheeseman et al., 1991). (The figures in Cetal’s paper are quite poor; for somewhat better figures on SAT, please see (Hayes, 1997), for the Hamiltonian cycle problem see (van Horn et al., 2018) and for their traveling salesman experiment, please see (Sleegers et al., 2020);). But the inverse was also true: for problem instances far away from the solvability phase transition, required computation time was low, either because one of its many solutions was quickly found, or a ‘no-solution’ dismissal was readily reached. But for one NP-complete problem, all this was about to change.

2 EXACT ALGORITHMS

The Hamiltonian cycle problem is a quintessential example of an NP-complete problem. It comes in many different shapes and forms, but in its most elementary formulation involves finding a path (a sequence of distinct edges) in an undirected and unweighted graph that visits every vertex exactly once, and forms a closed loop.

A true classic, the problem already appeared on Richard Karp’s infamous list of “21 NP-complete problems” from 1972 (Karp, 1972) but interestingly enough, a later edition of the same paper contains a striking stipulation from Karp himself: “[I failed to prove the NP-completeness of the undirected Hamiltonian cycle problem; that reduction was provided independently by Lawler and Tarjan]” (Karp, 2008;
Figure 2: According to traditional views, the hardest Hamiltonian cycle problem instances lie close to its solvability phase transition, the Komlós-Szemerédi-bound. Indeed this is true for large random ensembles, but a deliberate evolutionary search for the hardest yes/no-instances found instances that are hard for all these algorithms, sometimes by several orders of magnitude than random ensembles (see markers for $Ham_0$ and $NonHam_0$).

Karp, 1972) – the latter being Robert Tarjan, another mastodont of computer science. His original publication seems hard to find, but Garey and Johnson provide a proof of the Hamiltonian cycle problem’s NP-completeness from the vertex cover problem. A direct transformation to the satisfiability problem (SAT) is also possible, as shown for planar Hamiltonian cycle problems in cubic graphs (which have maximum vertex degree three)(Garey et al., 1976). Although clever, the direct practicality could be compromised by the large sizes of the Hamiltonian cycle problem instances that can arise even from the transformation of a single small SAT-instance. This principle, that instance sizes need not hold under NP-complete transformations is colloquially known as the ‘TommyGun Theorem’.

Being NP-complete, the Hamiltonian Cycle problem has no known polynomial-time exact algorithm\(^2\).

\(^2\)An ‘exact algorithm’ for a decision problem such as the Hamiltonian cycle problem, will always return a solution, given enough runtime.
In fact, the best upper bound was set by Richard Karp himself, together with Michael Held, with a dynamic programming algorithm that runs in $O(n^2 2^n)$ (Held and Karp, 1962). Independently discovered and published by Richard Bellman, it is still the fastest algorithm for the worst-case instance, and hence has the lowest known complexity – which is still exponential (Bellman, 1962).

Although well-regarded, Held-Karp is by no means the only exact algorithm for the Hamiltonian cycle problem. In fact, a number of recursive backtracking algorithms have been devised through the years. Plain depth-first search was invented far before any modern day computer in 1882. As early as 1974, the far more sophisticated Rubin’s algorithm already contained a surprising number of efficient subprocedures for non-Hamiltonicity checks (Rubin, 1974). Nine years later, an algorithm was proposed by Sylvano Martello (Martello, 1983), who has a wide track record with other NP-complete problems such as (perfect) rectangle packing (Iori et al., 2021b; Iori et al., 2021c)(Braam and van den Berg, 2022)(van den Berg et al., 2016). Martello’s algorithm ’595’ is not particularly efficient, but not nearly as bad as Cetal’s algorithm, probably the most inefficient of them all (Cheeseman et al., 1991)(van Horn et al., 2018). Nevertheless, Cetal’s work had value: as an enormous contribution to the awareness of hardness variation. An important note however, is that their experiment on the traveling salesman problem instance hardness was critically flawed due to a roundoff error (Sleegers et al., 2020). Even so, Cetal’s assessment on the other three NP-complete problems (in particular the Hamiltonian cycle problem) that hard instances reside close to the yes/no phase transition still, holds up for large randomized sets. If only they had used Frank Rubin’s algorithm from 20 years earlier, or even just Van Horn’s from 2018 (which is just a basic inversion of Cetal’s (van Horn et al., 2018)), their results might have taken a much larger step forward.

But the best performance award turned out to be reserved for Basil Vandegriend & Joe Culberson (abbreviated to “Vacul”), whose Vacul’s algorithm, even better than Frank Rubin’s, was published in 1998, and corroborated that even for that algorithm, the hardest problem instances were close to the phase transition (Vandegriend and Culberson, 1998). Finally, a comprehensive comparative study by Joeri Sleegers showed that all the algorithms are quite similar: backtracking algorithms with degree priority, pruning and check subroutines switched on or off. Sleegers, aided by modern day computation power, generated large ensembles of random graphs, which he ran through all six backtracking algorithms\(^3\), tying up earlier findings, showing that the hardest instances for all these algorithms are close to the yes/no phase transition (Sleegers and Van den Berg, 2021).

### 3 EVOLUTIONARY HARDNESS

So until 2020 the case for the Hamiltonian was crystal clear: like all NP-complete problems, it has a solvability phase transition, in this (rare) case exactly characterized by the Komlós-Szemeredi bound (or KS-bound), and the hardest instances, which are significantly harder than a given random instance, are located very close to that bound for all major backtracking algorithms (Sleegers and Van den Berg, 2021).

But 21st century computing power was about to throw a wrench in the machine, as Joeri Sleegers was about to evolve Hamiltonian cycle problem instances, making them as hard as possible using two evolutionary algorithms. First, a fully stochastic hillClimber was implemented; the algorithmic process was as follows: a single mutation on the incumbent hardest instance, which was retained if and only if the mutant was harder or equally hard. Otherwise, the mutation was reverted. Second, the population-based plant propagation algorithm (PPA), whose central idea is that fitter individuals produce many offspring with few mutations whereas unfitter individuals produce fewer offspring with high mutability, thus allegedly “[balancing the forces of exploration and exploitation]” (Vrielink and van den Berg, 2019). Since its introduction by Abdellah Salhi and Eric Fraga (Salhi and Fraga, 2011), the paradigm has seen a number of applications (Sulaiman et al., 2018)(Sleegers and van den Berg, 2020a)(Sleegers and van den Berg, 2020b)(Fraga, 2019)(Rodman et al., 2018), as well as some spinoffs (Sulaiman et al., 2016)(Paauw and van den Berg, 2019; Dijkzeul et al., 2022)(Selamoglu and Salhi, 2016)(Haddadi, 2020)(Geleijn et al., 2019). For the experiment, a constant popSize = 10 was adopted, which produced 25 offspring every generation, which were distributed and mutated as shown in table 1.

<table>
<thead>
<tr>
<th>Rank</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6 – 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>#offspring</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>#mutations</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

\(^3\)Source code is publicly available (Source, 2022).
Both evolutionary algorithms used three mutation
types with equal probability:
1. to insert an edge at a randomly chosen unoccu-
pied place in the graph,
2. to remove a randomly chosen existing edge from
the graph,
3. and to move an edge, which is effectively equal to
a removal mutation followed by an insert mutation
(on a different unoccupied place).

For the hillClimber runs, 20 randomly generated
graphs were evenly dispersed in terms of edge density,
ranging from 0 to \( \frac{1}{2}V \cdot (V - 1) \) edges, corresponding to
to edge densities \( \in \{0\%,5\%,10\%...95\%\} \). For the PPA
runs, twenty initial populations with \( popSize = 10 \)
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 evolution-
ary algorithms are free to insert and remove edges
from graphs at every step of a run. The rationale be-
hind this choice of edge densities is that earlier results
could have been biased from the initialization on the
Komlós-Szemerédi bound. This spread out approach
would cover a much wider area of the state space – at
least in terms of initial conditions. For the end results
however, it did not made much of a difference.

From these evenly distributed initial positions,
both algorithms ran 3000 function evaluations, corre-
sponding to 3000 generations for the stochastic hill-
Climber, but 120 generations of PPA. For the hardest
instance, both algorithms converged on the same in-
stance, a highly regular wall-and-clique-graph (Fig-
ure 1), not having a Hamiltonian cycle. For yes-
instances however (graphs which do have a Hamil-
tonian cycle) the hardest was in fact not a solitary
graph, but rather a constellation of 27 equihard non-
isomorphic graphs which were connected by one-
bite mutations: the Hamiltonian plateau (Figure 1)
(Sleegers and van den Berg, 2022). In this paper,
we will show that these evolutionarily found instances
are not only hard for Vacul’s algorithm, but possess a
more universal hardness.

4 EXPERIMENT & RESULTS

The hardest evolutionarily found instances (as seen
in Figure 1) are all 14-vertex graphs. The reason
that Sleegers & Van den Berg originally used such
instances, is that because the evolutionary algorithm
constantly pushes for hardness, measured through the
number of recursions for the most efficient known
backtracking algorithm, even a single function eval-
uation of the evolutionary algorithm can require
millions of recursions, which consumes incredible
amounts of computational power over the evolution-
ary trajectory, especially for population-based algo-
rithms.

For a fair comparison, we generated an ensemble
of random graph which also have \( V = 14 \) vertices.
The maximum number of edges that a \( V = 14 \) graph
instance can hold is \( \frac{1}{2} \cdot 14 \cdot 13 = 91 \), so for the ran-
dom ensemble, we generated 1000 graphs for each
\( E \in [1,91] \), meaning we covered the entire edge den-
sity range with a set of 91,000 graphs. We will call
this ensemble \( Rand91K \), and all graphs are publicly
available (Source, 2022). Then, we added the single
nonHam graph and the 27 Ham graphs, the hardest
no- and yes-instances as found by our evolutionary al-
gorithms. After that, each of these 91028 problem in-
stances was decided in six experiments, one for each
recursive instance, and the results can be found in
Figure 2.

Of the 91,000 instances in \( Rand91K \), 61720 were
Hamiltonian, whereas 29280 were not. These num-
bers might seem counterintuitive, but one should re-
alize that the KS-bound for Hamiltonicity in a graph
of \( V \) vertices scales as \( \frac{1}{2} \cdot V ln(V) + \frac{1}{2} \cdot V ln(ln(V)) \),
whereas possible number of edges scales as \( \frac{1}{2} \cdot V (V - 1) \). The combinatorial peak lies somewhat lower at
\( \frac{1}{2} \cdot V (V - 1) \), but even for very modest numbers of
\( V \), there are far more TRUE-instances than FALSE-
instances for the Hamiltonian cycle problem. For
\( V = 10 \), approximately 73% of all existible instances
are Hamiltonian; for \( V = 14 \), the percentage is al-
ready 93\%\(^4\). It is paradoxical but true that for this NP-
complete decision problem, nearly every existable in-
stance is a yes-instance.

Nonetheless, NP-complete classification relies on
the guarantee of exactness; it follows that all 91,000
instances were decided upon by all six algorithms, the
results of which can be found in Table 2 in columns
2-5. Results for the evolutionarily found hard in-
stances are in the final two columns. In line with pre-
vious results, the three algorithms with pruning and
check routines (bottom three rows) outperform the
more basic algorithms. In addition, the evolution-
arily found Ham and NonHam are very hard for all
backtracking algorithms, even though the difference
is less acute for the basic algorithms, which are in-
efficient on a large part of the \( Rand91K \) set to begin
with.

For almost all algorithms, the nonHam was
harder then every instance in the 91,000 random graph

\(^4\)Numbers in this study differ a bit because we forced
1,000 graphs for every possible edge degree.
ensemble. For Vacul’s, Martello’s and Rubin’s, it was 3197 times, 1914 times and 1223 times harder than the hardest graph in the ensemble, which was non-Hamiltonian in Vacul’s and Martello’s, but Hamiltonian in Rubin’s case, surprisingly enough. A quick-and-dirty explanation might be that nonHam0 narrowly ‘dodges’ the pruning and test procedures that these algorithms have, leaving it up to a high number of recursions.

For the non-pruning non-check algorithms, results were a lot less conclusive: in both plain depth-first and Van Horn’s, nonHam0 was ‘only’ 1.34 times harder than the hardest (non-Hamiltonian) instance in the ensemble. For Cetal’s, it wasn’t even the hardest, but 0.65 times the hardness of the hardest (non-Hamiltonian) instance in the ensemble, and 1.43 times harder than the second hardest (non-Hamiltonian) instance in the ensemble. This appears largely a testament of the efficiency of the algorithms with checks and pruning, or, as Steven Skiena once said: “Clever pruning can make short work of surprisingly hard problems” (Skiena, 1998).

For the 27 graphs of Ham0 set, the results were far less resounding, but still a clear distinction could again be noticed for the three algorithms with checks and pruning. For the highly efficient algorithm by Vacul, the entire Ham0 set was 113 times harder than the hardest instance from the Rand91K-set. For Rubin’s algorithm, the 27 instances of Ham0 set no longer form a plateau. They are nonetheless a hard set, requiring between 75,894 and 37,824 recursions, still 15 times more than the hardest Rand91K-instance at 2528 recursions. For Martello’s, the 27 graphs of Ham0 do constitute a plateau, whose instances are 48 times harder than the hardest graph from Rand91K.

For the non-pruning non-check algorithms, the Ham0 was still a hard set of instances, but results were a lot messier. In Van Horn’s, no less than 60 instances (58 non-Hamiltonian and 2 Hamiltonian) from the Rand91K-set were harder than Ham0. For Cetal’s algorithm, the 81 hardest instances contained the entire Ham0-set, the nonHam0 and another 53 Rand91K-instances. For depth-first, the Ham0-set, the nonHam0-instance and 421 Rand91K-instances made up the 449 hardest instances. The quick and dirty explanation is that this algorithm underperforms on the Ham0-set, but underperforms even worse on some of the Rand91K-instances. Nonetheless the difference with the nonHam0, which is the almost exclusively hardest instance for all studied algorithms, is stark.

5 CONCLUSIONS

The current results present some evidence that the hardest Hamiltonian cycle instances for one exact backtracking algorithm, are hard(est) for other exact backtracking algorithms too. The contrast is greatest in the most efficient algorithms, but that has more to do with the inefficiency of the other backtracking algorithms. Still, we must not close our eyes to nuances in the narrative: for some inefficient algorithms, still harder instances exist, be it very few. Nonetheless, nonHam0 and the Ham0-set can be considered almost universally hard for backtracking algorithms.

The fact that these hardest instances do not appear in large random ensembles such as the Rand91K-set also begs the question what makes the Hamiltonian cycle problem NP-complete. Is it possible that efficient backtrackers solve all but a very small number of instances, which can be listed in a table? And if so, how does the table scale under increasing V? Considering their structural regularity, they might even have a short algorithmical description. We should further refine these instances. Although somewhat out of fashion, it is clear that the class of recursive backtracking algorithms has not yet revealed all of its secrets.

REFERENCES


