Encoding the Hamiltonian Cycle Problem into SAT Based on Vertex Elimination

Neng-Fa Zhou
CUNY Brooklyn College and the Graduate Center, NY, USA

Abstract

This paper presents a SAT encoding, called vertex elimination encoding (VEE), for the Hamiltonian Cycle Problem (HCP). The encoding maps a Hamiltonian cycle in the reduced graph after vertex elimination to a Hamiltonian cycle in the original graph. While VEE is not competitive for large dense graphs due to its large encoding sizes, it can be utilized to reduce graphs when they are sparse. This paper compares VEE with the distance encoding, and shows that the hybridization of these two encodings is effective for the benchmarks. For the knight's tour problem, in particular, the hybrid encoding solves some middle-sized instances that were beyond the reach for previous eager SAT encodings.

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning; Computing methodologies → Planning and scheduling; Hardware → Theorem proving and SAT solving

Keywords and phrases Graph constraints, the Hamiltonian cycle problem, SAT encoding, Vertex elimination, Graph synthesis

Digital Object Identifier 10.4230/LIPIcs.CP.2024.40

Category Short Paper

Acknowledgements The author would like to thank Ruiwei Wang, Roland Yap Hock Chuan, and the anonymous reviewers for helpful comments.

1 Introduction

The Hamiltonian Cycle Problem (HCP), a classic problem in graph theory, seeks to find a cycle in a given directed graph that includes each and every vertex exactly once. With the availability of fast Satisfiability (SAT) solvers, various studies have been conducted to encode the HCP into SAT to leverage the solving power[2, 3, 6, 9, 10, 11]. The HCP can be encoded with degree and no-sub-cycle constraints. Most of the reported SAT encodings focus on how to translate no-sub-cycle constraints into SAT. Despite some successes, current SAT-based solvers are not yet competitive on many problems with other solvers, such as Constraint Programming (CP) and Answer Set Programming (ASP) solvers.

Vertex elimination [8] is a problem-solving technique used in various graph problems. The idea of the technique is to simplify complex graphs by removing certain vertices from a graph while preserving important properties. Recently, an encoding based on vertex elimination has been proposed for encoding acyclicity of directed graphs into SAT [7]. While this encoding is effective for sparse graphs, it is infeasible for large dense graphs due to its explosive encoding sizes. An improved encoding, which combines the vertex elimination encoding and the leaf elimination encoding, has been found to outperform both encodings on various types of graphs [13].

The vertex elimination technique can be applied to the HCP based on the observation that, if there exists a Hamiltonian cycle in a graph, then there must also exist a Hamiltonian cycle in a smaller graph obtained by removing a vertex. This paper proposes an encoding based on vertex elimination, called VEE, for the HCP. Let G be a directed graph, v be a vertex of G, and G' be the graph obtained after v is eliminated from G. While the existence

of a Hamiltonian cycle in G guarantees the existence of a Hamiltonian cycle in G', it is not inversely true, as vertex elimination may introduce cycles into G'. VEE generates constraints to ensure that the existence of a Hamiltonian cycle in G' guarantees the existence of a Hamiltonian path in G between two adjacent vertices of the eliminated vertex v.

The encoding size of VEE is prohibitively huge for a large dense graph. In the worst case, it requires $O(n^3)$ variables and generates $O(n^4)$ clauses. Following the idea of the hybrid encoding for acyclicity of graphs [13], this study also compares a hybrid encoding, which combines VEE and the distance encoding [11]. The hybrid encoding iteratively applies VEE as long as the graph meets a certain sparsity threshold, and switches to distance encoding when the smallest degree exceeds the threshold. The experimental results show that the hybrid encoding has the best overall performance, and solves some problems that were beyond the reach for previous eager SAT encodings.

2 The HCP and Distance Encoding

Given a base directed graph $G = (V, E)$, where V is a set of vertices, and E is a set of directed edges, called *arcs*, the HCP seeks to find a subgraph $H_G = (H_V, H_E)$, where $H_V \subseteq V$, $H_E \subseteq E$, and the arcs in H_E form one cycle that connects all the vertices in H_V. To represent H_G, this paper uses a binary variable, called a *characteristic variable*, for each vertex v in V, denoted as b_v, and each arc (u, v) in E, denoted as b_{uv}. A vertex v is said to be an *in-vertex* if $b_v = 1$. Similarly, an arc (u, v) is said to be an *in-arc* if $b_{uv} = 1$.

The graph H_G is completely determined by the characteristic variables as follows:

$$H_V = \{v \mid v \in V,\ b_v = 1\}$$
$$H_E = \{(u, v) \mid (u, v) \in E,\ b_{uv} = 1\}$$

If an arc is in, then both of its incident vertices must also be in: for each arc $(u, v) \in E$, $b_{uv} \to b_u \wedge b_v$.

The modeling of HCP with characteristic variables on vertices and arcs is generic, and the `circuit(`L`)` and `subcircuit(`L`)` constraints available in CP systems [1], where L is a list of domain variables representing the base graph, can be converted to constraints on characteristic variables. Let the length of L be n. In `circuit(`L`)`, all the vertices are assumed to be included in the resulting Hamiltonian cycle, therefore, $b_v = 1$ for each vertex in $1..n$. In `subcircuit(`L`)`, the vth variable in L is bound to v iff $b_v = 0$ for $v \in 1..n$.

The HCP can be decomposed into *degree* constraints and *no-sub-cycle* constraints. Let k be the cardinality of H_V: $k = \sum_{v \in V} b_v$. The degree and no-sub-cycle constraints are only enforced when $k > 1$. Various encodings are possible based on the framework. The following gives an adapted distance encoding [11], which can be traced back to the standard decomposer used in MiniZinc [5] and the integer programming formulation [4].

The degree constraints require every vertex to be in a cycle if $k > 1$:

For each $v \in V$:

$$k > 1 \wedge b_v \to \sum_{(u,v) \in E} b_{uv} = 1 \tag{D-1}$$

$$k > 1 \wedge b_v \to \sum_{(v,w) \in E} b_{vw} = 1 \tag{D-2}$$

For each vertex in H_V, constraint (D-1) forces it to have exactly one incoming arc, and constraint (D-2) forces it to have exactly one outgoing arc.

With constraints (D-1) and (D-2), the graph represented by the characteristic variables may contain sub-cycles. One well-known technique used in MIP and SAT encodings for HCP to prevent sub-cycles is to map vertices to different positions. The distance encoding chooses a vertex as the starting vertex, and treats each vertex's position as the distance from the starting vertex. For each vertex v, the distance encoding uses a binary variable s_v to indicate if v is the starting vertex, and an integer-domain variable d_v ($0 \leq d_v \leq n-1$) to indicate v's distance from the starting vertex, where n is the number of vertices in the base graph G. The following constraints are imposed on the variables:

$$k > 1 \rightarrow \sum_{v \in V} s_v = 1 \tag{D-3}$$

For each $v \in V$:

$$s_v \rightarrow b_v \tag{D-4}$$

$$s_v \rightarrow d_v = 0 \tag{D-5}$$

Constraint (D-3) ensures that there is a unique starting vertex if $k > 1$. Constraint (D-4) states that the starting vertex is an in-vertex, and constraint (D-5) forces the starting vertex's distance to be 0.

In addition to the above constraints, constraint D-6, given below, ensures that vertices are positioned successively:

$$\text{For each } (u,v) \in E\text{: } b_{uv} \wedge \neg s_v \rightarrow d_v = d_u + 1 \tag{D-6}$$

For each in-arc (u, v), if v is not the starting vertex, then v is the successor of u. There are several different ways to encode the successor constraint $d_v = d_u + 1$ [11]. The binary adder encoding is used in this study.

Constraints (D-1) through (D-6) guarantee that the last-positioned vertex is connected back to the starting vertex, as the last vertex must have distance $k-1$ and the degree constraints force it to be connected back to the starting vertex.

3 Vertex Elimination Encoding for HCP

Let $G = (V, E)$ be a directed graph with no self-loops and v be a vertex in V, the vertex elimination operation on v produces a directed graph $G' = (V', E')$:

$$\begin{aligned} V' &= V \setminus \{v\} \\ E' &= E \setminus \{(u,v) \mid (u,v) \in E\} \\ &\quad \setminus \{(v,w) \mid (v,w) \in E\} \\ &\quad \cup \{(u,w) \mid u \in nbs^-(v),\ w \in nbs^+(v), u \neq w\} \end{aligned}$$

where

$$\begin{aligned} nbs^-(v) &= \{u \mid (u,v) \in E\} \\ nbs^+(v) &= \{w \mid (v,w) \in E\}. \end{aligned}$$

The operation eliminates v's incident arcs, and adds the arc (u, w) into E' for each u in $nbs^-(v)$ and each w in $nbs^+(v)$ ($u \neq w$) if the arc is not contained in E. An arc in E' is said to be *new* if it is not in E and is newly added by vertex elimination.

As the resulting graph G' preserves the acyclicity of G [7], vertex elimination has the following properties:

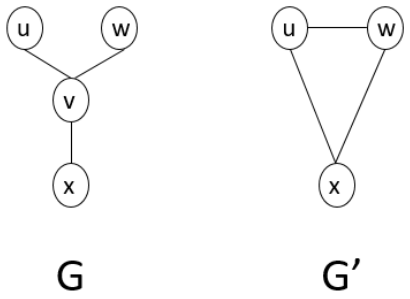

Figure 1 Vertex elimination introduces a new cycle.

- If there exists a Hamiltonian cycle in the original graph G, then there must also exist a Hamiltonian cycle in the resulting graph G' after vertex elimination.
- The existence of a Hamiltonian cycle in the resulting graph G' does not guarantee the existence of a Hamiltonian cycle in the original graph G as vertex elimination may introduce cycles. Figure 1 gives an example, where graph G has no Hamiltonian cycles, but the resulting graph G' has Hamiltonian cycles after v is eliminated from G.[1]

Recall that a Hamiltonian cycle H_G of a graph G is represented by characteristic variables associated with the vertices and the arcs. Let $G' = (V', E')$ be the graph obtained after vertex v is eliminated from graph $G = (V, E)$, and let the characteristic variables for G' be b'_z, where z is a vertex or an arc. The characteristic variables for vertices are unchanged after a vertex is eliminated, so for each vertex $v \in V', b'_v = b_v$. For each arc $(u, w) \in E' \cap E$, if $(u, v) \notin E$ or $(v, w) \notin E$, then $b_{uw} = b'_{uw}$. This means that for arcs that are not incident to the eliminated vertex v, G' inherits the characteristic variables from G.

Let k be the cardinality of H_G and k' be the cardinality of $H_{G'}$. The following constraint must hold: $k = k' + b_v$. A Hamiltonian cycle H'_G can be mapped to a Hamiltonian cycle H_G if and only iff $H_{G'}$ corresponds to a Hamiltonian path in G between a neighbor w in $nbs^+(v)$ and a neighbor u in $nbs^-(v)$ of the eliminated vertex v $(u \neq w)$. The vertex elimination encoding (VEE) ensures the mapping from $H_{G'}$ to H_G with constraints.

In H_G, there must be exactly one incoming arc to the eliminated vertex v and exactly one outgoing arc from v if $k > 1$ and v is in H_V. This is ensured by the degree constraints (VE-1) and (VE-2):

$$k > 1 \wedge b_v \rightarrow \sum_{(u,v)\in E} b_{uv} = 1 \tag{VE-1}$$

$$k > 1 \wedge b_v \rightarrow \sum_{(v,w)\in E} b_{vw} = 1 \tag{VE-2}$$

Also, there cannot exist cycles of size 2 involving v if $k' > 1$.

For each $(u, v) \in E$, if $(v, u) \in E$:

$$k' > 1 \rightarrow \neg b_{uv} \vee \neg b_{vu} \tag{VE-3}$$

Constraint (VE-3) ensures that the existence of $H_{G'}$ entails the existence of a Hamiltonian path in G between two distinct adjacent vertices of v.

[1] The edges are assumed to be doubly directed.

A new arc (u, w) that is included in E' but not in E indicates a path from u to w via the eliminated vertex v. The following constraints are imposed on the newly added arcs.

$$\text{For each } (u, w) \in (E' \setminus E)\text{: } b'_{uw} \to b_{uv} \wedge b_{vw} \tag{VE-4}$$

$$\sum_{(u,w)\in E'-E} b'_{uw} \leq 1 \tag{VE-5}$$

Constraint (VE-4) ensures that, if an arc (u, w) in $E' - E$ is included in $H_{G'}$, then both (u, v) and (v, w) must be included in H_G. Constraint (VE-5) bans multiple such paths, making the case in Figure 1 impossible.

If there is an incoming arc (u, v) to the eliminated vertex v and an outgoing arc (v, w) from v in H_G, then the arc (u, w) must occur in $H_{G'}$ but not in H_G.

$$\text{For each } (u, v) \in E,\ (v, w) \in E,\ u \neq w\text{:}$$

$$b_{uv} \wedge b_{vw} \to b'_{uw} \tag{VE-6}$$

$$b_{uv} \wedge b_{vw} \to \neg b_{uw} \tag{VE-7}$$

Constraints (VE-6) and (VE-7) ensure that a Hamiltonian cycle $H_{G'}$ can be mapped to a Hamiltonian cycle H_G by removing the arc (u, w) from $H_{G'}$ and adding the arcs (u, v) and (v, w).

Recall that G' inherits the characteristic variables from G for the arcs that are not incident to the eliminated vertex v. The following constraint constrains the characteristic variables of the arcs that are incident to v:

$$\text{For each } (u, v) \in E,\ (v, w) \in E,\ u \neq w\text{:}$$

$$\neg b_{uv} \vee \neg b_{vw} \to b_{uw} = b'_{uw} \tag{VE-8}$$

Constraint (VE-8) ensures the correspondence of $H_{G'}$ to a Hamiltonian path in G.

The correctness of VEE is guaranteed by the fact that a Hamiltonian cycle in G' corresponds to a Hamiltonian path from a neighbor w in $nbs^+(v)$ to a neighbor u in $nbs^-(v)$ of the eliminated vertex v ($u \neq w$), and the path can be extended to a cycle by adding the arcs (u, v) and (v, w).

The encoding size, which is dominated by constraint (VE-8), depends on the sparsity of the graph. In the worst case, which happens when the graph is complete, VEE generates $O(n^2)$ new variables for the characteristic variables of E'[2] and adds $O(n^3)$ clauses in each step, where n is the size of V. Overall, VEE requires $O(n^3)$ variables and $O(n^4)$ clauses.

4 Hybrid Encoding

Due to the formidable encoding sizes of VEE for large dense graphs, VEE is generally not a feasible encoding for HCP. Nevertheless, VEE can be utilized to reduce a graph when the graph is sparse, and a compact encoding can be employed to encode the resulting dense graph. This idea follows the hybrid encoding for acyclicity of graphs [13].

Any encoding for HCP can be hybridized with VEE. This paper uses the distance encoding (DIST). The hybrid encoding iteratively applies VEE until a sparsity condition becomes false, and after then, it switches to DIST.

[2] As every vertex is incident to the eliminated vertex, E' inherits no variables from E.

The sparsity condition can be defined in many different ways. Let n be the number of vertices in the original base graph, d be the smallest out-degree of the vertices in the current graph,[3] and σ be the total number of eliminated vertices so far. The sparsity condition used in the experiment is: $d \times \sigma \leq n$. For instance, when $d = 1$, the condition is always true, and VEE is used; when $d = 2$, the condition becomes false when more than half of the vertices have been eliminated.

5 Experimental Results

All the encodings presented in the paper have been implemented in Picat[4] (version 3.6#3), which employs Kissat[5] as the underlying SAT solver. This experiment uses the same elimination ordering as the one used in [7, 13], namely, choosing a vertex with the smallest degree. The SAT encodings of the basic constraints can be found in [11, 12].

This study has compared VEE, DIST, the hybrid encoding (HYBRID) on the benchmark suite used in [11], which consists of several instances of the knight's tour problem and several HCP instances taken from the Flinders challenge set[6] with numbers of vertices ranging from 338 to 1584. For these benchmarks, k, which indicates the cardinality of the resulting Hamiltonian cycle, is set to be n, the number of vertices in the base graph. All the CPU times reported below were measured on Linux Ubuntu with an Intel i7 3.30GHz CPU and 32G RAM. The time limit used was 20 minutes per instance.

Table 1 compares the encodings on CPU time, which includes both the translation and solving times (the entry TO indicates timeout). The column **instance** shows the instances, where ktxx are knight's tour instances, and graphxxx are the instances taken from the Flinders challenge set. The other three columns give the times taken by the three encodings for the instances.

It can be seen that, among the three encodings, **HYBRID** performs the best on the knight's tour instances, while **DIST** performs the best on all of the Flinders instances, except for graph254 and graph48. While **VEE** fails to solve 8 of the instances (6 of the kt instances and 2 of the Flinders instances) and **DIST** fails to solve 4 of the instances (3 of the kt instances and 1 of the Flinders instances), **HYBRID** solves every instance within the time limit.

VEE is not competitive on the knight's tour instances because of the large encoding sizes. For example, for kt30, the 30×30 instance, the generated CNF code by **VEE** contains 23,299,909 clauses with 7,648,642 variables. For the Flinders instances, on the other hand, **VEE** is quite competitive, winning on two of the instances, because the graphs are much more sparse than the kt graphs. One of the reasons why **HYBRID** performs better on the knight's tour graphs than the Flinders graphs could be the structures of the graphs. While the knight's tour graphs are quite dense when the encoder switches from **VEE** to **DIST**, the Flinders graphs remain quite sparse at the switching times. For instance, for kt30, the smallest degree is 8 at the switching time after 120 out of the 900 vertices are eliminated; for graph237, the smallest degree is 3 at the switching time after 493 out of the 1476 vertices are eliminated.

[3] As the graphs in all the benchmarks are undirected, a vertex's degree is defined as its out-degree.
[4] `http://picat-lang.org`
[5] `https://github.com/arminbiere/kissat`
[6] `http://fhcp.edu.au/fhcpcs`

Table 1 A comparison on CPU times(seconds).

Instance	VEE	DIST	HYBRID
kt12	28.75	7.11	**0.32**
kt14	135.80	5.77	**1.23**
kt16	614.23	118.45	**2.72**
kt18	1050.80	16.55	**3.65**
kt20	TO	20.70	**6.16**
kt22	TO	19.60	**19.21**
kt24	TO	76.31	**46.03**
kt26	TO	TO	**116.14**
kt28	TO	TO	**192.73**
kt30	TO	TO	**200.98**
graph162	TO	**33.89**	39.47
graph171	45.38	**5.35**	50.29
graph197	78.64	**13.16**	488.38
graph223	TO	**80.05**	200.71
graph237	125.66	**12.27**	237.51
graph249	62.48	**1.89**	61.04
graph252	182.27	**18.57**	468.85
graph254	**84.55**	TO	338.34
graph255	245.61	**31.30**	66.49
graph48	**0.75**	217.88	64.96

HYBRID is able to solve some middle-sized instances of the knight's tour problem that were beyond the reach for eager SAT encodings.[7] For example, it solves the 40×40 instance in 2711 seconds. In contrast, **DIST** fails to solve the 40×40 instance in 24 hours, and **VEE** fails to translate the instance to CNF. While **HYBRID** clearly advances the state of the art for the knight's tour problem, it is still not comparable with CP and ASP solvers, which can solve much larger instances than 40×40, thanks to their reachability-checking capabilities during search.

6 Discussion and Conclusion

This paper presents a SAT encoding, called vertex elimination encoding (VEE), for the HCP. The encoding maps a Hamiltonian cycle in the reduced graph after vertex elimination to a Hamiltonian cycle in the original graph. While VEE, in its current form, is not competitive for large dense graphs due to its large encoding sizes, it can be utilized to reduce graphs when they are sparse. This paper compares VEE with the distance encoding, and shows that the hybridization of these two encodings is effective for the benchmarks. For the knight's tour problem, in particular, the hybrid encoding solves some middle-sized instances that were beyond the reach for previous eager SAT encodings.

The HCP is significant due to its broad applications across various disciplines, and researchers continue to explore various techniques to encode HCP into SAT in order to leverage the solving power of the cutting-edge SAT solvers. Most of the previous SAT encodings focus on how to translate no-sub-cycle constraints into SAT. VEE is novel in the sense that it is the first encoding based on vertex elimination for the HCP, and, unlike the previous encodings, it does not need to deal with no-sub-cycle constraints.

[7] Lazy encodings (e.g., [3, 9]), which incrementally generate sub-cycle elimination clauses, may be able to solve some large instances.

VEE sheds new light on future explorations in encoding the HCP into SAT. The encoding presented in the paper generates $O(n^4)$ clauses for a complete graph with n vertices. One future exploration is to improve the encoding so that it has a lower order of an encoding size. Another future exploration is to hybridize multiple encodings with VEE. Once a graph is reduced by vertex elimination, the resulting graph can be encoded with any encoding. For very dense graphs, the *bijection* encoding [2], which uses an edge constraint for each non-arc pair of vertices, could be more efficient. It is always a challenge to devise the best heuristic for switching from VEE to another encoding. While the simple heuristic used in this paper has achieved encouraging results, it is not meant to be the best. As training data can be easily generated by running the solver under different settings, it could be viable to devise a near-optimal switching heuristic using machine learning.

References

1 Nicolas Beldiceanu, Mats Carlsson, and Jean-Xavier Rampon. Global constraint catalog, 2021. URL: http://sofdem.github.io/gccat/gccat/.

2 Alexander Hertel, Philipp Hertel, and Alasdair Urquhart. Formalizing dangerous SAT encodings. In *SAT*, volume 4501, pages 159–172, 2007. doi:10.1007/978-3-540-72788-0_18.

3 Marijn J. H. Heule. Chinese remainder encoding for hamiltonian cycles. In *SAT*, pages 216–224, 2021. doi:10.1007/978-3-030-80223-3_15.

4 C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, 1960.

5 Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. MiniZinc: Towards a standard CP modelling language. In *CP*, pages 529–543, 2007. doi:10.1007/978-3-540-74970-7_38.

6 Steven David Prestwich. SAT problems with chains of dependent variables. *Discret. Appl. Math.*, 130(2):329–350, 2003. doi:10.1016/S0166-218X(02)00410-9.

7 Masood Feyzbakhsh Rankooh and Jussi Rintanen. Propositional encodings of acyclicity and reachability by using vertex elimination. In *Thirty-Sixth AAAI Conference*, pages 5861–5868. AAAI Press, 2022. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20530, doi:10.1609/AAAI.V36I5.20530.

8 Donald J. Rose, Robert Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976. doi:10.1137/0205021.

9 Takehide Soh, Daniel Le Berre, Stéphanie Roussel, Mutsunori Banbara, and Naoyuki Tamura. Incremental SAT-based method with native Boolean cardinality handling for the Hamiltonian cycle problem. In *Logics in Artificial Intelligence (JELIA)*, pages 684–693, 2014.

10 Miroslav N. Velev and Ping Gao. Efficient SAT techniques for absolute encoding of permutation problems: Application to Hamiltonian cycles. In *SARA*. AAAI, 2009. URL: http://www.aaai.org/ocs/index.php/SARA/SARA09/paper/view/837.

11 Neng-Fa Zhou. In pursuit of an efficient SAT encoding for the Hamiltonian cycle problem. In *CP*, pages 585–602, 2020. doi:10.1007/978-3-030-58475-7_34.

12 Neng-Fa Zhou and Håkan Kjellerstrand. Optimizing SAT encodings for arithmetic constraints. In *CP*, pages 671–686, 2017. doi:10.1007/978-3-319-66158-2_43.

13 Neng-Fa Zhou, Ruiwei Wang, and Roland H. C. Yap. A comparison of SAT encodings for acyclicity of directed graphs. In *26th International Conference on Theory and Applications of Satisfiability Testing, SAT*, pages 30:1–30:9, 2023. doi:10.4230/LIPICS.SAT.2023.30.