Hyperset Individualisation Algorithms

Simone Boscaratto[1,*,†], Francesco Nascimben[1,*,†] and Alberto Policriti[1,*,†]

[1]Dipartimento di Matematica, Informatica e Fisica, Università di Udine, Via delle Scienze, 206, Udine, 33100, Italy

Abstract

In this work, we propose a novel framework for graph canonisation called *Hyperset Individualisation*, using bisimulation on a set-theoretic framework in an effort to tackle the Graph Isomorphism problem on simple graphs. Building on this idea, we define algorithm HID, which we prove to be strictly more expressive than colour refinement. Moreover, we define two versions of a k-dimensional HID, which we prove to have different expressive power.

Keywords

Graph isomorphism, graph canonisation, hypersets, bisimulation, individualisation

1. Introduction

The long-standing Graph Isomorphism complexity problem, i.e. the problem of determining the complexity of establishing whether two graphs are isomorphic or not, remains open to this day. Particularly relevant to it is the thoroughly studied Weisfeiler-Leman algorithm. Originally introduced in its 2-dimensional version [1] and later generalised by Babai and Mathon [2], the k-dimensional Weisfeiler-Leman algorithm WL_k produces a stable colouring of the k-tuples of nodes $\vec{v} \in V^k$ of an input graph $G = \langle V, E \rangle$. Basically, it produces a *one-way error* multiset $\mathsf{WL}_k(G)$ containing the stable colours of all tuples: two isomorphic graphs always result in the same multisets, while the converse does not hold true in general. Hence, it provides an incomplete, although statistically very accurate [3], test for graph isomorphism: larger values of k correspond to larger classes of correctly identified graphs [4], at the cost of increasing time complexity.

The first aim of this paper is to introduce and study a *Hyperset Individualisation* algorithm HI, in an attempt to give an alternative approach to tackle the Graph Isomorphism problem with respect to WL_1, to this day the most used algorithm to practically identify graphs up to isomorphism [5]. In this novel framework, *node individualisation* and *bisimulation reduction* techniques are combined in a set-theoretic framework to produce a certificate for each given undirected simple graph: this is achieved in time $\mathcal{O}(nm \log n)$, where n is the number of nodes and m the number of edges of the original graph. While node individualisation is a well-known concept in the field of graph canonisation (see, e.g., [6, 7]), and so is bisimulation reduction in several other fields (see, e.g., [8, 9]), to the best of our knowledge the proposed algorithm is the first attempt at using the two of them together. As HI is shown to be incomparable to WL_1, it is enriched by encoding nodes' degree, thus obtaining HID: this algorithm is then proved to be strictly more powerful than Weisfeiler-Leman's. Next, we define and preliminary explore two k-dimensional generalisations of it, namely HIDE_k and HIDA_k.

This paper is organised as follows: in Section 2, we introduce the adopted notation and the tackled problem, recalling some essential notions. In Section 3 we define the (basic) Hyperset Individualisation algorithm, showing its accomplishments and failures. On these grounds, Section 4 shows the improvements that can be put forward by focusing on HID. In Section 5, we define HIDE_k and HIDA_k, describe their currently known properties and compare their effectiveness, while in Section 6 we provide a

ICTCS 2025: Italian Conference on Theoretical Computer Science, September 10–12, 2025, Pescara, Italy
*Corresponding author.
†These authors contributed equally.
simoneboscaratto@outlook.com (S. Boscaratto); francesco.nascimben@uniud.it (F. Nascimben); alberto.policriti@uniud.it (A. Policriti)
0009-0008-8192-6898 (S. Boscaratto); 0009-0009-2863-165X (F. Nascimben); 0000-0001-8502-5896 (A. Policriti)

preliminary analysis of their computational complexity. Finally, in Section 7 we draw conclusions and state some open problems, mainly concerning the expressiveness of the aforementioned methods.

2. Basics

In this paper, standard graph-theoretic and set-theoretic notations will be adopted. In particular, a tuple will be delimited by angled parentheses $\langle\ \rangle$ and a multiset by the parentheses $\{\!\{\ \}\!\}$; unordered pairs will be represented as $(\cdot\,,\cdot)$, as they often do in existing literature. $\uplus$ will denote the multiset *sum* operation, which sums the multiplicities of each element, common or not, of the addend multisets. Connections between two nodes will be referred to as *edges* in the case of undirected graphs and as *arcs* in the directed case; $G = \langle V_G, E_G\rangle$ (resp., $\vec{G} = \langle V_{\vec{G}}, \vec{E}_{\vec{G}}\rangle$) will denote an undirected (resp., directed) graph, while subscripts will be omitted whenever clear from the context and unless otherwise specified. In this case, by n and m we will denote, respectively, the number of nodes and edges (or arcs) of a graph.

We will mainly treat the case of finite undirected graphs without self-loops and weighted or multiple edges, also referred to as *simple* graphs. At due time, we will also require these graphs to be connected.

Consider then a pair of simple graphs $G = \langle V_G, E_G\rangle$ and $H = \langle V_H, E_H\rangle$.

Definition 2.1. An $isomorphism$ between two simple graphs G and H is a bijection $\phi\colon V_G \to V_H$ which preserves both *adjacencies* and *non-adjacencies*, i.e. $(u, w) \in E_G \Leftrightarrow (\phi(u), \phi(w)) \in E_H$ for all nodes u, w in V_G. If such ϕ exists, G is said to be *isomorphic* to H, denoted by $G \cong H$.

Given two graphs G and H, the *Graph Isomorphism problem* (from now on, also abbreviated as *GI*) consists in establishing whether $G \cong H$ or not.

2.1. Weisfeiler-Leman Algorithm

The 1-dimensional Weisfeiler-Leman (WL_1, for short), also known as *colour refinement*, is the simplest algorithm of the WL family. Given a graph $G = \langle V, E\rangle$, WL_1 produces a stable colouring of its nodes. Let $\mathcal{C}^1_i$ be the colouring produced by WL_1 after its i-th iteration: since we only consider uncoloured graphs, we assume the initial colouring $\mathcal{C}^1_0$ to be uniform for all nodes. At step $i \geq 1$ and for each node $v \in V$, WL_1 collects the colours of v's neighbours into a multiset $M_i(v)$, called the *aggregation map* of v at step i: a new colour $\mathcal{C}^1_i(v)$ is then computed from (and uniquely associated to) v's previous colour and its current aggregation map, by means of a perfect hash function HASH. M_0 is not defined.

$$M_i(v) = \left\{\!\!\left\{\mathcal{C}^1_{i-1}(w) : w \in \mathcal{N}(v)\right\}\!\!\right\} \qquad \mathcal{C}_i(v) = \textsc{hash}\Big(\mathcal{C}^1_{i-1}(v), M_i(v)\Big)$$

Two nodes share the same colour at step i only if they shared the same colour at step $i-1$ *and* their aggregation maps match. This refinement procedure repeats until a stable colouring $\mathcal{C}^1_\infty$ is reached: termination is guaranteed by the finiteness of V. We define $\mathsf{WL}_1(G) = \{\!\{\mathcal{C}^1_\infty(v) : v \in V\}\!\}$.

The generalised k-dimensional Weisfeiler-Leman (WL_k, for short) produces a stable colouring of the k-tuples in V^k. Each $\vec{v} \in V^k$ is initially coloured with its *atomic type* $\mathrm{atp}(\vec{v})$, which describes the (ordered) subgraph induced on G by this tuple. Formally, two tuples $\vec{u} = \langle u_1, .., u_k\rangle$ and $\vec{w} = \langle w_1, .., w_k\rangle$ have the same atomic type if and only if the mapping $u_i \mapsto w_i$ is an isomorphism from the G-subgraph induced by $\vec{u}$ to the G-subgraph induced by $\vec{w}$. We denote the initial colour of each tuple $\vec{v}$ by $\mathcal{C}^k_0(\vec{v}) = \mathrm{atp}(\vec{v})$. As for the 1-dimensional version, WL_k also proceeds by repeated aggregation of neighbouring colours: given $\vec{v} = \langle v_1, v_2, .., v_k\rangle$ and a node w, the tuple obtained by replacing exactly one of its nodes v_i with w is the i-th w-neighbour of $\vec{v}$, which we denote by $\vec{v}_{i,w}$. The colour-update schema is similar to the one described for WL_1, the sole difference being that the aggregation map of a tuple $\vec{v}$ is now a multiset of k-*tuples* of colours, one for each node w in G.

$$M^k_i(\vec{v}) = \left\{\!\!\left\{\langle\mathcal{C}^k_{i-1}(\vec{v}_{1,w}), \ldots, \mathcal{C}^k_{i-1}(\vec{v}_{k,w})\rangle : w \in V\right\}\!\!\right\} \qquad \mathcal{C}^k_i(\vec{v}) = \textsc{hash}\Big(\mathcal{C}^k_{i-1}(\vec{v}), M^k_i(\vec{v})\Big)$$

Again, the refinement procedure iterates until a stable colouring $\mathcal{C}^k_\infty$ is reached. We define $\mathsf{WL}_k(G) = \{\!\{\mathcal{C}^k_\infty(\vec{v}) : \vec{v} \in V^k\}\!\}$. WL_k can be implemented to run in time $\mathcal{O}(n^{k+1}\log n)$ on n-vertex graphs [10].

2.2. Hypersets and Bisimulation

The set-theoretic framework for our algorithm is here introduced.

A *well-founded* set x is such that every descending chain in the membership relation starting from it is finite and acyclic, meaning that $x \ni x_1 \ni \cdots \ni x_n$ eventually halts for a finite n—in a *pure* set theory, x_n is always the empty set $\emptyset$—and $x \neq x_i \neq x_j$ for every pair $i \neq j$, $i, j \in \{1, \ldots, n\}$. To compare two well-founded sets, the *extensionality* criterion is applied: two sets are equal if and only if they have the same elements.[1]

On the contrary, a *non-well-founded* set, or *hyperset*, admits loops in the membership relation: with such a move we grant, for example, the existence of “extra” sets x such that $x \in x$, or $x \in x_1 \in \cdots \in x_n \in x$. By overcoming the limits imposed by a well-founded definition of $\in$, Forti and Honsell [12], and then Aczel [13], created a richer universe in which (hyper-)sets like $\Omega = \{\Omega\}$ exist. However, by allowing loops, we also allow multiple alternative representations of the same hyperset (e.g., Ω can be represented as the hyperset solving the set-theoretic equation $\zeta = \{\zeta\}$, or $\zeta = \{\{\zeta\}\}$, and so on). This can be seen more easily by translating the set-theoretic representations of a hyperset into their graph-theoretic equivalents, on which we will rely upon to define a proper equality criterion for non-well-founded sets.

We refer to [14] for the following definitions, based on the aforementioned [13]. Consider a directed graph $\vec{G} = \langle V, \vec{E} \rangle$ with finitely many nodes and no labelled or multiple edges between them; loops and self-loops are admitted. If there exists a node $p \in V$, dubbed as the *point* of such a graph, from which every other node is reachable, we will say that $\vec{G} = \langle V, \vec{E}, p \rangle$ is an *accessible pointed graph*, *apg* for short. By interpreting each node as a (hyper-)set and each arc as the inverse membership relation—so that the existence of an arc $\langle u, w \rangle \in \vec{E}$ means that the set associated to w belongs to the set associated to u, or $w \in u$ by abuse of notation—$\vec{G}$ can be *decorated* by suitably labelling each node with its set-theoretic equivalent. For example, if $\vec{G} = \langle \{p\}, \emptyset, p \rangle$ is a single node with no edges, p shall be interpreted as the empty set $\emptyset$, as this is the only set without elements; on the contrary, if $\vec{G} = \langle \{p\}, \{\langle p, p \rangle\}, p \rangle$ is made just by the node p but with a self-loop, this shall be labelled as the aforementioned Ω, as it is the only hyperset to contain just itself as an element (at any nesting depth). It is easy to see that every well-founded set has a unique representation as an apg; by reversing this idea, the *anti-foundation axiom* (AFA) by Aczel states that each apg admits a unique decoration.

Define the *transitive closure* $\mathrm{trCl}(x)$ of a (hyper-)set x as the set containing its elements, and their elements, and so on. More formally:

$$\mathrm{trCl}(x) = x \cup \bigcup_{y \in x} \mathrm{trCl}(y).$$

If an apg can be decorated by associating to each node either the set corresponding to its point, or an element of its transitive closure, it will be referred to as the *pointed membership graph* of the (hyper-)set labelling its point. If a hyperset h has a finite transitive closure, then we can represent it by a finite pointed membership graph: in this case we will say that h is a *hereditarily finite rational hyperset*.[2]

Observe that, as the hyperset Ω can be represented in several set-theoretic ways, it has also multiple graphical representations. To handle this, the following definition is needed to outline an equality criterion compatible with hypersets.

Definition 2.2 (Bisimulation, bisimilarity). Let $\vec{G} = \langle V, \vec{E} \rangle$ be a directed graph. A binary relation $\flat$ among the nodes of V is said to be a *bisimulation* on $\vec{G}$ if $u \flat w$ with $u, w \in V$ always implies that:

- for every child u' of u there exists a child w' of w such that $u' \flat w'$, and
- for every child w' of w there exists a child u' of u such that $u' \flat w'$.

[1] This is very common to many standard set theories, in particular Zermelo-Fraenkel’s; see, e.g., [11].

[2] To see that every hyperset can have a graphical representation with infinitely many nodes, it is sufficient to “unwrap” its loops (e.g., Ω can be represented by an infinitely descending chain, see [13]); to match this definition, though, it is sufficient that there exist finite ones. Set-theoretically, a hereditarily finite rational hyperset is defined as the solution to a finite system of finite set equations, see e.g. [15].

The union of all bisimulations on $\vec{G}$ is still a bisimulation, it is called *bisimilarity*, and it is the coarsest bisimulation on $\vec{G}$: bisimilarity defines an equivalence relation on the nodes, denoted by $\equiv_{\vec{G}}$.

The notion of bisimilarity constitutes an equality criterion for hypersets: stating that two pointed membership graphs (i.e., their points) are bisimilar means that they both represent the same hyperset. Intuitively, one can think of bisimilarity as the coarsest partition of the nodes set grouping functionally equivalent ones: assuming $u \equiv_{\vec{G}} w$, any "move" performed starting from u can be mirrored starting from w. Notice that the existence of an isomorphism ϕ between two oriented graphs implies that every vertex w of the first graph is bisimilar to its image $\phi(w)$. The converse does not hold true in general.

2.3. Node Individualisation

Individualisation of single nodes, for the purpose of symmetry breaking, is a well-known technique in the field of graph canonisation. Most state-of-the-art practical tools, such as `nauty` and `Traces` [7], rely on the so called *individualisation-refinement* paradigm, which alternately applies WL_1 to the graph and assigns a unique colour to one node from a non-singleton class (chosen according to some heuristic) until a discrete node partition. Although this approach generates a (potentially) exponentially large tree of colourings, appropriate heuristics and exploitation of discovered automorphisms allow such tools to prune significant parts of the tree, leading to fast performances in most cases.

Individualisation finds applications in important theoretic results. t-CR bounded graphs, for which a discrete colouring can be obtained by repeatedly applying WL_1 and assigning a unique colour to each node of a non-singleton class of size at most t, play a key role in Grohe et al.'s test for isomorphism running in time $n^{\text{polylog}(h)}$ for n-vertex graphs excluding some h-vertex graph as a minor [16].

In both previous examples, this simmetry-breaking technique is iteratively paired with WL_1 in order to reach a discrete colouring. On the other hand, in the HI framework introduced below, individualisation is applied only once for each node v in the graph, before launching a bisimulation algorithm which produces a hyperset h_v associated to v.

3. The Hyperset Individualisation Algorithm

The first definition of the *Hyperset Individualisation* algorithm HI is aimed at giving a set-theoretic cut to the graph canonisation problem by assigning a *multiset of hypersets* to each undirected simple graph. As for WL_1, the result obtained after performing HI is a multiset encoding pieces of information about the chosen graph; however, while this is based on nodes' degrees for the former, the latter computes bisimilarity contractions by interpreting the graph as the pointed membership graph of some hyperset.

Due to the following lemma, we can, and will, restrict our analysis to *connected* simple graphs.

Lemma 3.1. *Let $G = \langle V_G, E_G \rangle$ and $H = \langle V_H, E_H \rangle$ be undirected, possibly non-connected simple graphs. Let $G' = \langle V_G \cup \{s_G\}, E_G \cup \{(s_G, v) : v \in V_G\} \rangle$ and $H' = \langle V_H \cup \{s_H\}, E_G \cup \{(s_H, v) : v \in V_H\} \rangle$ be the graphs obtained by adding to both a source node reaching each node of the original graphs. Then, $G \cong H$ if and only if $G' \cong H'$.*

HI pseudocode is reported as Algorithm 1. After replacing each (undirected) edge with a pair of opposite arcs, a new arc from a node v to a new node $v_\emptyset$—which, from a set-theoretic perspective, represents the empty set—is added. Given the so-modified pointed graph $\vec{G}_v$, with v itself as the point, a tool such as the one defined in [8] can be applied to get its bisimulation contraction, resulting in the apg of a hyperset h_v. Then, h_v is added to a multiset and the operation is repeated for every node of the original graph: the resulting multiset is the certificate given by the algorithm to the input graph.

Remark 1. Consider the definition of *rank* of a node u in an individualised graph $\vec{G}_v$ as its distance from the empty set $\emptyset$ following a simple path, i.e. without cycles.[3] Clearly, for any $\vec{G}_v$, the only node of rank 1 is v itself; furthermore, by definition of bisimulation, two nodes of different rank will not collapse within each other in h_v.

[3]This definition is consistent with the one of rank for hereditarily finite well-founded sets.

Algorithm 1 Hyperset individualisation HI

Require: $G = \langle V, E \rangle$ ▷ Undirected, simple, connected

Ensure: $\mathsf{HI}(G)$

$\vec{E} \leftarrow \emptyset$

for $(u, w) \in E$ **do**

  $\vec{E} \leftarrow \vec{E} \cup \{\langle u, w \rangle, \langle w, u \rangle\}$ ▷ Replace an edge with a pair of arcs

end for

$\mathsf{HI}(G) \leftarrow \{\!\!\{\,\}\!\!\}$ ▷ Initialise $\mathsf{HI}(G)$ as the empty multiset

for $v \in V$ **do**

  $\vec{G}_v \leftarrow \langle V \cup \{v_\emptyset\}, \vec{E} \cup \{\langle v, v_\emptyset \rangle\}, v \rangle$ ▷ Append a node labelled as the empty set $\emptyset$ to v

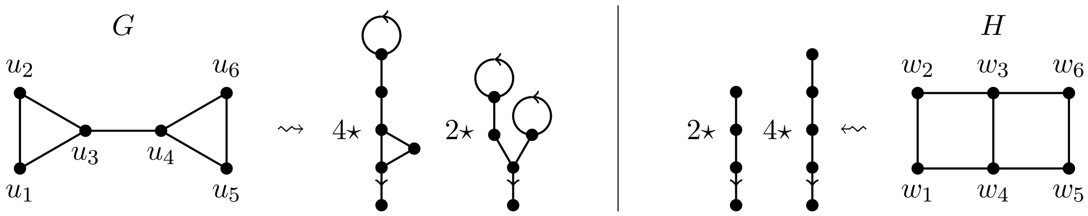
  $h_v \leftarrow \mathsf{DPP}(\vec{G}_v)$ ▷ Run the bisimulation algorithm defined in [8] on $\vec{G}_v$

  $\mathsf{HI}(G) \leftarrow \mathsf{HI}(G) \uplus \{\!\!\{h_v\}\!\!\}$ ▷ Add the resulting hyperset to $\mathsf{HI}(G)$

end for

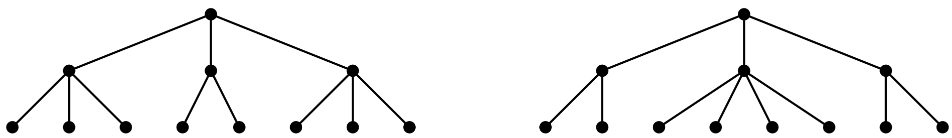
The hyperset individualisation algorithm provides a one-way error test for graph isomorphism: given a pair (G, H) of undirected graphs, $\mathsf{HI}(G) \neq \mathsf{HI}(H)$ implies that G and H are not isomorphic, while the converse does not necessarily hold.

Example 3.1. HI can distinguish non-isomorphic graph pairs which WL_1 cannot. Indeed, by performing HI on the graphs G and H below, we obtain the multisets containing the depicted hypersets with the reported multiplicities (recall that each undirected edge represents a pair of opposed arcs in hypersets).



Observe that in $\mathsf{HI}(G)$ (resp., $\mathsf{HI}(H)$) there are four copies of the hyperset obtained by individualising u_i (resp., w_i) with $i \in \{1, 2, 5, 6\}$ and two copies of the one obtained by individualising u_i (resp., w_i) with $i \in \{3, 4\}$. As they differ between the two graphs, HI distinguishes G and H; on the contrary, WL_1 produces the same colours for each pair (u_i, w_i). Yet, this graph pair is distinguished also by WL_2.

Example 3.2. Despite WL_1 can identify each non-isomorphic tree, the following pair is not distinguished by HI (for both graphs, nodes at the same depth yield the same hyperset after individualisation).



From the previous examples, it follows that the expressive powers of WL_1 and HI are incomparable.

4. Refinements

As Example 3.2 shows, HI (in fact, bisimulation) does not take into account the size of a node's neighbourhood: indeed, it proves itself more powerful than WL_1 exactly when this property is irrelevant, but it could fail otherwise. A natural way to address this limitation, in an attempt to make HI strictly stronger than WL_1, is to endow each node with an initial label encoding its degree.

In order to maintain a purely set-theoretic view, the nodes' degrees can be conveniently represented by adding edges towards a node in a directed, descending chain (*gadget graph*) of nodes which is external with respect to the original graph. Basically, this chain represents the (Zermelo) ordinals, or *super-singletons* of the empty set up to the maximum degree of a node in the graph; the fact that $v \in G$ has degree d can be represented by an arc $\langle v, v_{\{\emptyset\}^d} \rangle$, where, iteratively, $\{\emptyset\}^0 := \emptyset$ and $\{\emptyset\}^i := \{\{\emptyset\}^{i-1}\}$.

Definition 4.1. Consider an undirected simple graph $G = \langle V_G, E_G \rangle$ and define its gadget graph as

$$\vec{D}_G = \langle V_{\vec{D}_G}, \vec{E}_{\vec{D}_G} \rangle \coloneqq \big\langle \big\{ v_{\{\emptyset\}^d} : 0 \leq d \leq M \big\}, \big\{ \langle v_{\{\emptyset\}^d}, v_{\{\emptyset\}^{d-1}} \rangle : 1 \leq d \leq M \big\} \big\rangle,$$

where $M \coloneqq \max_{v \in G}\{\deg(v)\}$. Then, the *Hyperset Individualisation algorithm with Degrees* HID is the variant of HI obtained by re-defining $\vec{G}_v$ as

$$\vec{G}_v = \big\langle V_G \cup V_{\vec{D}_G},\ \vec{E}_G \cup \vec{E}_{\vec{D}_G} \cup \{ \langle w, v_{\{\emptyset\}^{\deg(w)}} \rangle : w \in V_G \} \cup \{ \langle v, v_\emptyset \rangle \},\ v \big\rangle$$

at line 7 in Algorithm 1 ($\vec{E}_G$ is the set of arcs replacing the undirected edges of G).

Notice how this edit shall not create ambiguity with the process of individualising a node: since we are now dealing just with connected graphs, no node is without edges, so they all have a positive degree; therefore, the only arc pointing to the node dubbed as the empty set—now part of the gadget graph—is the one issuing from the intended individualised node.

Remark 2. Let $\vec{G} = \langle V_G \cup V_{\vec{D}_G}, \vec{E}_G \cup \vec{E}_{\vec{D}_G} \cup \{ \langle w, v_{\{\emptyset\}^{\deg(w)}} \rangle : w \in V_G \} \rangle$ (the same as in Definition 4.1, but without individualising any node and with no definite point; from now on, we will keep this notation whenever it is not ambiguous). Then, computing maximum bisimulation on the graph $\vec{G}$ is equivalent to computing maximum bisimulation on the graph G by degree-encoding colours to each node. Moreover, the well-founded part of $\vec{G}$ is its gadget graph $\vec{D}_G$.

An interesting result that proves the greater effectiveness of HID w.r.t. HI and WL_1 is the following.

Theorem 4.1. HID *is strictly more expressive than* WL_1.

Proof. By example 3.1, there exists a non-isomorphic graph-pair distinguished by HID, but not by WL_1. Thus, we only need to show that every graph-pair distinguished by WL_1 is also distinguished by HID. Let $\mathcal{C}_i(v)$ be the colour assigned to the node v after the i-th WL_1 iteration: we recall that $\mathcal{C}_i(v)$ identifies the *subtree structure* $T_i(v)$ of height i rooted in v.[4] We show that, for any pair of nodes (u, w):

$$(\exists i \in \mathbb{N})(\mathcal{C}_i(u) \neq \mathcal{C}_i(w)) \implies h_u \neq h_w, \tag{1}$$

where h_u (resp., h_w) is the (pointed membership graph of the) hyperset resulting after performing bisimulation contraction on $\vec{G}_u$ (resp., $\vec{G}_w$), pre-processed with nodes' degrees.

For $i = 0$, the implication is trivially true. Assume now that $i > 0$, $\mathcal{C}_i(u) \neq \mathcal{C}_i(w)$ and $\mathcal{C}_j(u) = \mathcal{C}_j(w)$ for all $j < i$. Since the subtree structures match up to height $i - 1$, but not further, there must exist two leaves $\hat{u} \in T_{i-1}(u)$, $\hat{w} \in T_{i-1}(w)$ such that $\deg(\hat{u}) \neq \deg(\hat{w})$ and the degree sequences from the roots to their parents are equal. Besides, at least one between $\hat{u}$ and $\hat{w}$ must be of rank exactly k:[5] otherwise, the WL_1 colouring for u and w would have diverged in an earlier iteration. It follows that, in the degree-partitioned graph, there is a coloured path of length $i - 1$ starting from u which cannot be replicated starting from w: thus, u is not bisimilar to w or, equivalently, $h_u \neq h_w$, proving (1).

Therefore, by considering every node-pair (u, w), we obtain that if the multisets produced for two graphs (G, H) by WL_1 are different, so are the multisets produced by HID. □

As HID is so proved to be strictly more expressive than WL_1, we are interested in studying its limits. As an upper bound, we observe that HID is not as expressive as WL_3.

Theorem 4.2. *There exists a graph pair distinguished by* WL_3*, but not by* WL_2 *or* HID.

Proof. Consider the 4x4 Rook's and the Shrikhande graphs in Fig. 4, both strongly regular graphs with parameters $\langle n = 16, d = 6, \mu = 2, \tau = 2 \rangle$, where n is the number of nodes, d the degree of each node, μ and τ the number of common neighbours for adjacent and non-adjacent nodes, respectively. This pair is distinguished by WL_3, but not by WL_2 (more generally, WL_2 cannot tell apart SRGs with identical parameters [18]). HID is also unable to distinguish them, since, on both graphs, the resulting multiset contains 16 copies of the same hyperset. □

[4] A subtree-structure differs from a subtree, as the former contains the same node multiple times; see [17, p. 2542].

[5] Equivalently, it must be a previously unseen node in its respective subtree structure.

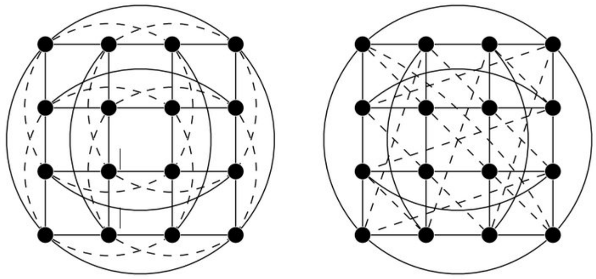


Figure 1: The 4x4 Rook's graph and the Shrikhande graphs as pictured in [19]. The graphs are not isomorphic, since the neighbours of a node form two separate 3-cycles in the Rook's graph, while they form a single 6-cycle in Shrikhande graph, despite having the same strongly regular graphs' parameters.

We point out two scenarios where HID allows to check for isomorphism in polynomial time.

Lemma 4.1. *Let the simple connected graphs* $G = \langle V_G, E_G \rangle$, $H = \langle V_H, E_H \rangle$ *be such that* $|V_G| = |V_H| = n$ *and* $\exists h \in \mathsf{HID}(G) \cap \mathsf{HID}(H) : |\mathrm{trCl}(h) = n + M + 1|$. *Then* $G \cong H$.

Proof. Notice that the h of the statement has maximum transitive closure: there are no bisimulation collapses among the n nodes originally in G or H, nor (trivially) among the $M + 1$ nodes of their gadget graphs. Any two graphs $\vec{G}_u$, $\vec{H}_w$ associated to h in $\mathsf{HID}(G)$ and $\mathsf{HID}(H)$ are, except in the connection to $\emptyset$, isomorphic to G and H themselves, respectively. The corresponding discrete partition induced by h on both G and H naturally yields an isomorphism between the two. □

Definition 4.2. Let $G = \langle V, E \rangle$ be a graph, let 2^V be the powerset of its nodes. The (node) *orbit partition* of G is the coarsest partition $O \subset 2^V$ of V such that all nodes belonging to the same class can be mapped into each other by an automorphism: such classes are called the (node) *orbits* of G. If $|O| = n$, we say that G has a *discrete* orbit partition, as it only admits the identity on V as an automorphism.

Lemma 4.2. *Let the simple connected graphs* $G = \langle V_G, E_G \rangle$, $H = \langle V_H, E_H \rangle$ *be such that* $|V_G| = |V_H| = n$ *and* $\mathsf{HID}(G) = \mathsf{HID}(H)$ *contains* n *distinct hypersets. Then, both graphs have discrete orbit partitions and the bijection* $\phi \colon V_G \to V_H$ *such that* $\phi(u) = w$ *if and only if* $h_u = h_w$ *is the only possible isomorphism between* G *and* H.

Proof. Two nodes producing different hypersets cannot belong to the same orbit (this is trivial: if they are mapped into each other by an automorphism, then they are bisimilar, thus they produce the same hyperset once individualised). Therefore, only bijections mapping nodes whose individualisation produces the same hyperset are candidate isomorphisms. □

If the conditions of the previous lemma hold, $G \cong H$ can be verified in linear time.

5. Moving to a Higher Plane: k-Dimensional HID_k

In this section, we move to the following, natural question: can the HI(D) idea be lifted to achieve higher expressive power, in a similar fashion to the k-dimensional WL? Specifically, we investigate different approaches to compute a multiset of hypersets associated to sets of $k > 1$ nodes at a time, in order to define an algorithmic family HID_k akin to WL_k.

Before doing so, observe that a 0-dimensional Hyperset Individualisation algorithm HI_0 is definable regardless of the precise definition given to any higher-dimensional version, as it subsumes the bisimulation contraction without individualising any node. Since there is no privileged starting point for the bisimulation algorithm, we will focus once again on the connected case only, so as to avoid that a single bisimulation reduction results in more than one hyperset.[6] For the same reason, in this case we will also avoid to identify the point of the graph and so of the resulting hyperset.[7]

Remark 3. HI_0—without degrees pre-processing—as applied to a simple connected graph G either results in the empty set $\emptyset$ or in the hyperset Ω. Thus, it can only distinguish the graph having just one node (and no edges) from any other simple connected graph.

Differently from the previous case, if the degrees' pre-processing is performed—thus obtaining the HID_0 algorithm—the graph results to be already partitioned before computing bisimulation. The following lemma holds true.

Lemma 5.1. HID_0 *is equivalent to* WL_1.

Proof. At each step and for each node v, WL_1 aggregates information about v's own degree and, iteratively, those of all nodes that can be reached from it. Therefore, two nodes will get the same colour in the final stable partitioning of the graph if and only if they have the same degree, and their neighbours have pairwise the same degree, and so on. So, computing maximum bisimulation on that graph is equivalent to collapsing all the nodes belonging to the same WL_1-class. ☐

5.1. Individualising by the Empty Set: HIDE_k

We define algorithm HIDE_k (*Hyperset Individualisation with Degrees and the Empty set*) on a connected simple graph G. Instead of individualising just a node at a time, we will take k-many, with $k \leq n$, and run maximum bisimulation contraction on the resulting graph, much as HID does. For the sake of completeness, a new node will be linked to the newly-individualised nodes to serve as the point of the resulting hyperset.

Definition 5.1. Let $G = \langle V, E\rangle$ be a connected simple graph, $n = |V|$ and $k \in \{2, \ldots, n\}$; let $\vec{G}$ be the directed version of G, encoding degrees through the gadget graph, as described in Remark 2. Then, the *Hyperset Individualisation algorithm with Degrees and the Empty set* of order k HIDE_k is the generalisation of HID obtained by replacing $v \in V$ with $S \subseteq V : |S| = k$ at every occurrence, and $\vec{G}_v$ with

$$\vec{G}^E_S \coloneqq \langle V_{\vec{G}} \cup \{v_S\},\ \vec{E}_{\vec{G}} \cup \{\langle v_S, w_i\rangle, \langle w_i, v_\emptyset\rangle : w_i \in S\},\ v_S\rangle,$$

at line 7 in Algorithm 1, thus obtaining $\mathsf{HIDE}_k(G) \coloneqq \{\!\!\{h^E_S \coloneqq \mathsf{DPP}(\vec{G}^E_S) : S \subseteq V, |S| = k\}\!\!\}$.

Observe that, since each hyperset is associated to a unique k-subset of V, $\mathsf{HIDE}_k(G)$ contains $\binom{n}{k}$-many hypersets; for clarity, such hypersets will be denoted as h^E_S for each subset S fulfilling the previous definition. It is not trivial to see how much the expressive power of HIDE_k varies by changing k. However, as a first result, we prove that the expressive power reached by HIDE_k on graphs with n-many nodes is the same as the one reached by HIDE_{n-k}.

Lemma 5.2. *Let* $G = \langle V_G, E_G\rangle$, $H = \langle V_H, E_H\rangle$ *be connected simple graphs. Then,* $\mathsf{HIDE}_k(G) = \mathsf{HIDE}_k(H)$ *if and only if* $\mathsf{HIDE}_{n-k}(G) = \mathsf{HIDE}_{n-k}(H)$.

Proof. Assume $\mathsf{HIDE}_k(G) = \mathsf{HIDE}_k(H)$: therefore, there is a one-to-one correspondence associating each $S \subseteq V_G$ to a $T \subseteq V_H$, both of cardinality k, in such a way that $h^E_S = h^E_T$. As they are both contracted by maximum bisimulation, for each node $u \in h^E_S$ there exists exactly one node $w \in h^E_T$ bisimilar to it: notice that u and w are the bisimulation contractions of $\{u' \in V_{\vec{G}^E_S} : u' \equiv_{\vec{G}^E_S} u\}$ and $\{w' \in V_{\vec{H}^E_T} : w' \equiv_{\vec{H}^E_T} w\}$ respectively, and that every $u' \in S$ if and only if $w' \in T$. Taking u' and w' as representatives of those bimilarity classes, and assuming $u' \in S$ and $w' \in T$ (resp. $u' \in V_G \setminus S$ and $w' \in V_G \setminus T$), by removing (resp. adding) the links from the sources v_S, v_T and to the empty set $v_\emptyset$ from (to) the both of them, they will still be bisimilar. Therefore, if u and w are bisimilar

[6]As Lemma 3.1 points out, we can reduce to this case without any loss of generality anyway.

[7]This comes both because it is useless in DPP algorithm and because any node can be coherently dubbed as point (as they all have the same transitive closure): any chosen point then should be ignored when the hypersets are compared.

w.r.t. h^E_S and h^E_T, then they will be bisimilar also w.r.t. the hypersets $h^E_{V_G \setminus S}$ and $h^E_{V_H \setminus T}$ obtained by individualising the previously non-individualised nodes, thus proving that $h^E_{V_G \setminus S} = h^E_{V_H \setminus T}$ and finally $\mathsf{HIDE}_{n-k}(G) = \mathsf{HIDE}_{n-k}(H)$. The opposite implication is proven in the same way. □

This result proves that HIDE_k reaches its maximum expressiveness for k at most $\lceil n/2 \rceil$. Since we have not obtained more precise results about it so far, we can just conjecture that such maximum is indeed reached at that exact point. To support our claim, the following lemma shows that HIDE_2 could be strictly more powerful than $\mathsf{HIDE}_1 = \mathsf{HID}$ on graphs with sufficiently many nodes.

Lemma 5.3. *There exists a graph pair distinguished by* HIDE_2*, but not by* WL_2 *or* HID.

Proof. Consider again the 4x4 Rook's and the Shrikhande graphs in Fig. 4. On the Rook's graph, HIDE_2 produces two distinct hypersets, obtained by individualising adjacent and non-adjacent node pairs, respectively. On the Shrikhande graph, three distinct hypersets are produced, obtained by individualising adjacent pairs, non-adjacent pairs with adjacent common neighbours, and non-adjacent pairs with non-adjacent common neighbours. □

Currently, we do not know if the expressive power of HIDE is bounded (i.e. if there exists a non-isomorphic graph pair which cannot be distinguished by HIDE_k, for any possible k), or if it can reach complete identification up to isomorphism.

5.2. Individualising by Atoms: HIDA_k

Next, we define an alternative generalised algorithm HIDA_k (*Hyperset Individualisation with Degrees and Atoms*). Let $\vec{G}$ be as before and add k nodes $A = \{a_1, \ldots, a_k\}$, called *atoms*, each belonging to a unique class, so that they are pairwise non-bisimilar by definition. Thus, the method consists in linking one node of $\vec{G}$ to one node of A at a time, then collapsing the resulting graph by maximum bisimulation. By introducing different atoms, we guarantee that each pair of the so-individualised nodes will not be bisimilar, which is not necessarily the case for HIDE_k; on the other hand, as the ordering of those atoms—and, consequently, of the individualised nodes—is irrelevant for that purpose, we need to introduce an equivalence relation in such a way that different orderings do not affect the reliability of the test.

Definition 5.2. Consider k atoms $A = \{a_1, \ldots, a_k\}$, and two hypersets with atoms $h^A_{\{u_1,\ldots,u_k\}}$, $h^A_{\{w_1,\ldots,w_k\}}$ such that the i-th node in the subscript is the only one connected by an arc to the atom a_i, for any $i \in \{1, \ldots, k\}$. We write $h^A_{\{u_1,\ldots,u_k\}} \sim_k h^A_{\{w_1,\ldots,w_k\}}$ if and only if $\exists \sigma \in S_k : h^A_{\{u_1,\ldots,u_k\}} = h^A_{\{w_{\sigma(1)},\ldots,w_{\sigma(k)}\}}$, where S_k is the symmetric group over the discrete interval $[1, k]$.

From this point onwards, with a slight abuse of notation in order to improve readability, we shall use h^A_S to denote the (permutation invariant) equivalence class $[h^A_S]_{\sim_k}$ to which the (permutation dependent) hyperset h^A_S belongs. We are now ready to define HIDA algorithmically.

Definition 5.3. Let $G = \langle V, E \rangle$ be a connected simple graph, $n = |V|$, $k \in \{2, \ldots, n\}$ and define a set of k-many atoms $A = \{a_1, \ldots, a_n\}$; let $\vec{G}$ be the directed version of G, encoding degrees through the gadget graph, as described in Remark 2. Then, the *Hyperset Individualisation algorithm with Degrees and Atoms* of order k HIDA_k is the generalisation of HID obtained by replacing $v \in V_G$ with $S \subseteq V_G : |S| = k$ at every occurrence and $\vec{G}_v$ with

$$\vec{G}^A_S := \big\langle V_{\vec{G}} \cup A \cup \{v_S\},\ \vec{E}_{\vec{G}} \cup \{\langle v_S, w_i \rangle, \langle w_i, a_i \rangle : w_i \in S\},\ v_S \big\rangle,$$

at line 7 in Algorithm 1, thus obtaining $\mathsf{HIDA}_k(G) := \{\!\{ h^A_S := \mathsf{DPP}(\vec{G}^A_S) : S \subseteq V, |S| = k \}\!\}$.

In order to distinguish a hyperset produced by HIDA_k w.r.t. other already defined algorithms, we will write it as h^A_S for a subset S of k nodes. Given the previous definition, when comparing the multisets produced by HIDA_k on a pair of graphs G and H, we will check if the hypersets they contain are equal *up to any permutation of atoms.* In this way, we do not have to consider all the possible permutations of atoms while performing the algorithm (leading to factorial space complexity), limiting the number of hypersets in $\mathsf{HIDA}_k(G)$ to $\binom{n}{k}$ on a graph G with $n \geq k$ nodes, as HIDE_k does. Instead, this complexity is eventually transferred to the research space, as, computationally speaking, we do not have a trivial way to address sets' comparison without imposing upon them a specific, although arbitrary, ordering.

The following lemma, along with its immediate corollary, allows us to conclude that the HIDA family defines a hierarchy of increasingly stronger algorithms, akin to WL_k.

Lemma 5.4. *Given a simple graph $G = \langle V, E\rangle$ such that $|V| = n$ and $k < n$, the multiset $\mathsf{HIDA}_{k+1}(G)$ uniquely determines $\mathsf{HIDA}_k(G)$, up to any permutation of the atoms.*

Proof. $\mathsf{HIDA}_k(G)$ is obtained from $\mathsf{HIDA}_{k+1}(G)$ by removing one atom a_i from each hyperset h^A_S and then checking for possible bisimulation collapse between the de-individualised node v_i and any other non-individualised node in h^A_S. In this way we get $k + 1$ new hypersets $h^{A\setminus\{a_i\}}_{S\setminus\{v_i\}}$ for $i \in \{1, \dots, k + 1\}$ from each S; however, as the same subset of k nodes can be obtained from $(n - k)$-many subsets of cardinality $k + 1$, from $|\mathsf{HIDA}_{k+1}(G)| = \binom{n}{k+1}$ we get

$$\binom{n}{k+1} \cdot \frac{k+1}{n-k} = \frac{n!}{(k+1)!(n-k-1)!} \cdot \frac{k+1}{n-k} = \binom{n}{k} = |\mathsf{HIDA}_k(G)|,$$

thus confirming that the cardinality of the produced multiset coincides with the one of $\mathsf{HIDA}_k(G)$. □

Corollary 5.1. *For all $k > 1$, HIDA_{k+1} induces a finer or equal partition on the universe of simple graphs than HIDA_k does.*

As the final point of this preliminary analysis, we prove that the highest expressiveness of HIDA_k—which is equivalent to explicitly checking for isomorphism—is reached for $k = n - 1$ over graphs with n nodes. It is currently unknown whether such expressiveness can also be achieved with a smaller k.

Lemma 5.5. *Let G and H be connected simple graphs with n nodes. Then, the following are equivalent.*

1. *$G \cong H$;*
2. *$\mathsf{HIDA}_n(G) = \mathsf{HIDA}_n(H)$;*
3. *$\mathsf{HIDA}_{n-1}(G) = \mathsf{HIDA}_{n-1}(H)$.*

Proof. Trivially, claim 1 implies claims 2 and 3, as the existence of an isomorphism between G and H implies $\mathsf{HIDA}_k(G) = \mathsf{HIDA}_k(H)$ for every $k \leq n$, in particular for $k = n$ and $k = n - 1$.

Assume claim 2 holds true. Then, any bijection $\phi\colon V_G \to V_H$ linking nodes connected to the same atom in two equal hypersets from $\mathsf{HIDA}_k(G)$ and $\mathsf{HIDA}_k(H)$ is an isomorphism, thus proving claim 1.

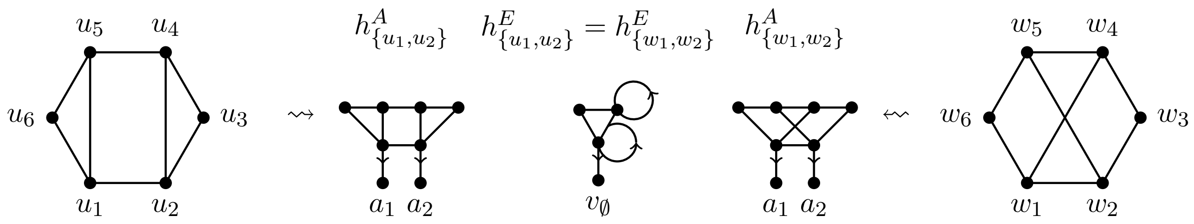
Assume claim 3 holds true: we will prove that in this case claim 2 holds, too. Since G and H are connected, each hyperset in both $\mathsf{HIDA}_{n-1}(G)$ and $\mathsf{HIDA}_{n-1}(H)$ has a unique node that has rank 2 with respect to at least one atom, i.e. it cannot collapse with any other node by computing bisimulation. Therefore, by appending a new atom to this spare node we obtain n copies of the same hyperset (up to any permutation of atoms) from both $\mathsf{HIDA}_{n-1}(G)$ and $\mathsf{HIDA}_{n-1}(H)$. As these are equal, the resulting hypersets are all $\sim_n$-equivalent, thus proving that $\mathsf{HIDA}_n(G) = \mathsf{HIDA}_n(H)$. □

5.3. Comparing Expressiveness

While the definition of the HIDE_k version comes quite naturally as a generalisation of HID, it is not immediate to see whether the HIDA_k version provides any advantage. By definition, HIDA_k is at least as strong as HIDE_k on a local level, since $h^E_S \neq h^E_T \Rightarrow h^A_S \neq h^A_T$ for any two subsets S, T of k nodes, while the opposite implication does not hold true.

Lemma 5.6. *Two sets of nodes $S = \{u_1, u_2\}, T = \{w_1, w_2\}$ can generate distinct hypersets $h^A_S \neq h^A_T$, but equal hypersets $h^E_S = h^E_T$.*

Proof. Consider the following example (gadget graphs and sources have been omitted for clarity).



As depicted above, individualisation of u_1 and u_2 in the first graph and of w_1 and w_2 in the second graph yield the same hyperset by doing so with the empty set (HIDE_2), but different ones with atoms (HIDA_2). The same can be said for pairs (u_4, u_5) w.r.t. (w_4, w_5) (trivial), and for (u_3, u_6) w.r.t. (w_3, w_6). ☐

It can be shown that by computing the whole $\mathsf{HIDE}_k/\mathsf{HIDA}_k$ certificates, the previous graphs are distinguished by both methods;[8] thus, although HIDA_k might be stronger at a local level, this does not imply that it is a stronger isomorphism test: HIDE_k might still be able to distinguish all graph pairs distinguished by HIDA_k, as long as $k \leq \lceil n/2 \rceil$. It is yet to be verified whether there exists some k^* such that HIDE_{k^*} identifies all n-vertex graphs up to isomorphism: if this holds, the maximum k required for HIDA_k would also be at most k^*, much smaller than the currently established $k \geq n - 1$ bound.

6. Implementation Sketch and Complexity Analysis

We now provide an analysis of the time/space complexity of the aforementioned methods; as a reference, we recall that WL_k has time complexity $\mathcal{O}(n^{k+1} \log n)$. Although a multiset of hypersets may seem harder to describe than the multiset of colours produced by WL_k, it must be noted that such colours are iteratively obtained by applying a hash function to a multiset of previously computed colours, i.e. for all purposes they are equivalent to nested multisets.

Multisets of hypersets can be handled, for instance, by keeping a list L of all distinct hypersets generated during a run of HI on a graph $G = \langle V, E \rangle$. Whenever a node $v \in V$ is individualised, $h_v \in L$ can be checked in polynomial time [14]: if this is the case, we simply increase by 1 the multiplicity of h_v in $\mathsf{HI}(G)$; otherwise, we add h_v to L and $\mathsf{HI}(G)$, with multiplicity 1.

A single hyperset h_v can be computed in time $\mathcal{O}(m \log n)$ by an algorithm such as [8]. Whether h_v belongs to L can be checked in time $\mathcal{O}(|L|\, m \log n)$, where $|L| < n$ is the number of unique hypersets in L, again using [8]. Algorithm 2 shows how to compare hypersets by their *accessible pointed graphs*, running bisimulation on a third apg: such procedure can then be applied to h_v and each $h \in L$.

Each $h \in L$ may also be associated to a unique integer in $[1, n]$, according to the time it first appeared in L, allowing for constant time comparisons between the hypersets of distinct nodes, if later needed.

To test a graph-pair (G, H) for isomorphism, one can simply run HI in parallel on both, taking care of checking both L_G and L_H whenever a hyperset h_v from either graph is computed, in order to keep the aforementioned hyperset enumeration coherent. In terms of space, this solution requires $\mathcal{O}(n + m)$ for each $h \in L_G/L_H$, by definition of bisimilarity: in the worst case, assuming each individualised hyperset to be distinct, HI has space complexity $\mathcal{O}(n(n + m))$. In terms of time, the overall complexity of HI is $\mathcal{O}(n^2 m \log n)$, since up to n^2 hypersets comparisons, each of cost $\mathcal{O}(m \log n)$, may be required if either $|L_G| \in \Theta(n)$ or $|L_H| \in \Theta(n)$. The same bounds hold for HID, as computing the initial degree-partition of the nodes and extending the graph takes time $\mathcal{O}(n + m)$.

Moving to the k-dimensional HIDE_k, complexity scales according to the number of computed hypersets: in the worst cases, space $\mathcal{O}(\binom{n}{k}(n + m))$ and time $\mathcal{O}(\binom{n}{k}^2 m \log n)$ may be required. HIDA_k may appear equally costly, since its final multiset also contains $\binom{n}{k}$ elements; however, it must be noted

[8]By individualising (u_1, u_4) or (u_1, u_5) w.r.t. (w_1, w_4) or (w_1, w_5), both HIDE_2 and HIDA_2 produce different hypersets.

Algorithm 2 Hypersets comparison

Require: $\vec{G}_1 = \langle V_1, \vec{E}_1, p_1 \rangle, \vec{G}_2 = \langle V_2, \vec{E}_2, p_2 \rangle$ ▷ *Apg*s of hypersets h_1, h_2 to be compared

Ensure: $h_1 \stackrel{?}{=} h_2$

```
1: V_0 ← V_1 ∪ V_2 ∪ {p_0}
2: E_0 ← E_1 ∪ E_2 ∪ {⟨p_0, p_1⟩, ⟨p_0, p_2⟩}
3: G_0 ← ⟨V_0, E_0, p_0⟩          ▷ New apg whose point is linked to the points of G_1 and G_2
4: h_0 ← DPP(G_0)
5: if p_1 ≡_{G_0} p_2 then        ▷ Equivalently, they are merged together in h_0
6:     h_1 = h_2                  ▷ h_1 and h_2 are bisimilar, and thus equal
7: else
8:     h_1 ≠ h_2
9: end if
```

that this relies on the use of equivalence relation $\sim_k$, which hides the computational cost of comparing two hypersets $h^A_{\{u_1,\dots,u_k\}}$ and $h^A_{\{w_1,\dots,w_k\}}$ up to any of the $k!$ possible permutations of the atoms. Since checking whether two hypersets belong to the same $\sim_k$ class requires $k!$ comparisons of the kind described in Algorithm 2, the overall time complexity of HIDA_k is $\mathcal{O}(k!\binom{n}{k}^2 m \log n)$. On a practical level, simple heuristics can be applied to avoid costly comparisons in all of the above algorithms: for instance, hypersets whose *apg*s differ in their number of nodes or edges will certainly be distinct.

7. Open Problems and Conclusions

In this preliminary work, we introduced the notion of Hyperset Individualisation algorithm HI, which combines a set-theoretic perspective, bisimulation, and a node individualisation technique in order to provide a novel approach to the Graph Isomorphism and Graph Canonisation problems. After proving that our 1-dimensional HID algorithm has a strictly stronger separation power than the well-known WL_1 algorithm, we defined two k-dimensional generalisations, called HIDE_k and HIDA_k, whose properties do not perfectly overlap. On n-vertex graphs, HIDE_k is proved to be exactly as expressive as HIDE_{n-k}, so that its peak must be reached at some $k \leq \lceil n/2 \rceil$; on the other hand, HIDA_k is at least as expressive as HIDA_{k-1} for any k, becoming a complete isomorphism test for $k \geq n-1$.

A number of open problems arise. HIDA_k is known to be at least as expressive as HIDE_k at a local level, but the exact relationship between the two families should be further investigated. It is not clear whether HIDE_k ever reaches the level of a complete isomorphism test for some $k \leq \lceil n/2 \rceil$: in this case, HIDA_k for the same (or lower) k would, too—thus for a k much lower than the already established bound of $k = n-1$.

How the $\mathsf{HIDE}_k/\mathsf{HIDA}_k$ and WL_k hierarchies intersect, besides the preliminary result on HID being strictly more expressive than WL_1, is another point of interest. Studying the behaviour of our algorithms on non-isomorphic graph pairs generated through the CFI construction [4] seems the most natural way to gain insight on this matter. If either HIDE_k or HIDA_k turned out not to line up with the WL_k hierarchy (up to some additive constant on their dimensionality), looking for a suitable logic capturing their expressiveness would be the next step.

From a practical standpoint, when checking for isomorphism between two graphs (G, H), one could think of iteratively applying $\mathsf{HIDE}_1, \mathsf{HIDE}_2, \dots, \mathsf{HIDE}_{k\leq n}$ until either a mismatch is found or a threshold (e.g. a bound on k) is met. In such a context, we would be interested in determining whether (and how much) we could restrict the choice of the k-sets to be individualised when running HIDE_k on G and H, depending on the previously computed multiset $\mathsf{HIDE}_{k-1}(G) = \mathsf{HIDE}_{k-1}(H)$, in order to optimise such a sequential application. For the same purpose, given two sets S and T of size k, being able to efficiently distinguish their hypersets h^E_S and h^E_T a priori, based on the hypersets for S and T's $(k-1)$-subsets, could greatly reduce the number of required comparisons. Entirely similar considerations apply to HIDA_k.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] B. Weisfeiler, A. A. Lehman, A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction, Nauchno-Technicheskaya Informatsia Ser. 2 (1968) 12–16.

[2] L. Babai, R. Mathon, Talk at the south-east conference on combinatorics and graph theory, 1980.

[3] L. Babai, P. Erdös, S. Selkow, Random graph isomorphism, SIAM J. Comput. 9 (1980) 628–635. doi:10.1137/0209047.

[4] J.-Y. Cai, M. Furer, N. Immerman, An optimal lower bound on the number of variables for graph identification, in: 30th Annual Symposium on Foundations of Computer Science, 1989, pp. 612–617. doi:10.1109/SFCS.1989.63543.

[5] M. Grohe, D. Neuen, Recent advances on the graph isomorphism problem, London Mathematical Society Lecture Note Series, Cambridge University Press, 2021, p. 187–234.

[6] F. Fuhlbrück, J. Köbler, I. Ponomarenko, O. Verbitsky, The Weisfeiler-Leman algorithm and recognition of graph properties, in: T. Calamoneri, F. Corò (Eds.), Algorithms and Complexity, Springer International Publishing, Cham, 2021, pp. 245–257.

[7] B. D. McKay, A. Piperno, Practical graph isomorphism, ii, Journal of Symbolic Computation 60 (2014) 94–112. URL: https://www.sciencedirect.com/science/article/pii/S0747717113001193. doi:https://doi.org/10.1016/j.jsc.2013.09.003.

[8] A. Dovier, C. Piazza, A. Policriti, An efficient algorithm for computing bisimulation equivalence, Theoretical Computer Science 311 (2004) 221–256. URL: https://www.sciencedirect.com/science/article/pii/S030439750300361X. doi:https://doi.org/10.1016/S0304-3975(03)00361-X.

[9] E. G. Omodeo, Bisimilarity, hypersets, and stable partitioning: a survey, Rend. Istit. Mat. Univ. Trieste Volume 42 (2010) 211–234.

[10] N. Immerman, E. Lander, Describing Graphs: A First-Order Approach to Graph Canonization, Springer New York, New York, NY, 1990, pp. 59–81. URL: https://doi.org/10.1007/978-1-4612-4478-3_5. doi:10.1007/978-1-4612-4478-3_5.

[11] T. Jech, Set Theory: The Third Millennium Edition, revised and expanded, Springer Monographs in Mathematics, 3 ed., Springer Berlin Heidelberg, 2003. URL: https://books.google.it/books?id=CZb-CAAAQBAJ.

[12] M. Forti, F. Honsell, Set theory with free construction principles, Annali della Scuola Normale Superiore di Pisa - Classe di Scienze 10 (1983) 493–522. URL: http://eudml.org/doc/83914.

[13] P. Aczel, Non-Well-Founded Sets, Csli Lecture Notes, Palo Alto, CA, USA, 1988.

[14] E. G. Omodeo, A. Policriti, A. I. Tomescu, On Sets and Graphs: Perspectives on Logic and Combinatorics, Springer, 2017. URL: https://link.springer.com/book/10.1007/978-3-319-54981-1. doi:10.1007/978-3-319-54981-1.

[15] S. Boscaratto, E. G. Omodeo, A. Policriti, On generalised ackermann encodings - the basis issue, in: E. D. Angelis, M. Proietti (Eds.), Proceedings of the 39th Italian Conference on Computational Logic, Rome, Italy, June 26-28, 2024, volume 3733 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3733/paper3.pdf.

[16] M. Grohe, D. Neuen, D. Wiebking, Isomorphism testing for graphs excluding small minors, SIAM Journal on Computing 52 (2023) 238–272. URL: https://doi.org/10.1137/21M1401930. doi:10.1137/21M1401930. arXiv:https://doi.org/10.1137/21M1401930.

[17] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, K. M. Borgwardt, Weisfeiler-Lehman graph kernels, J. Mach. Learn. Res. 12 (2011) 2539–2561.

[18] M. Fürer, On the combinatorial power of the Weisfeiler-Lehman algorithm, in: D. Fotakis, A. Pagourtzis, V. T. Paschos (Eds.), Algorithms and Complexity, Springer International Publishing, Cham, 2017, pp. 260–271.

[19] V. Arvind, F. Fuhlbrück, J. Köbler, O. Verbitsky, On Weisfeiler-Leman invariance: Subgraph counts and related graph properties, Journal of Computer and System Sciences 113 (2020) 42–59. URL: https://www.sciencedirect.com/science/article/pii/S0022000020300386. doi:https://doi.org/10.1016/j.jcss.2020.04.003.