July 2016

DIGITISED HUMAN LIFE FORM ISOMORPHIC ALGORITHM

T. MOHANAN

Department of computer science and engineering, Angel College of Engineering and Technology Under Anna university, Chennai, mohan.stanky@gmail.com

V. MOHANATHAN

Department of computer science and engineering, Angel College of Engineering and Technology Under Anna university, Chennai, vmohanathan@gmail.com

D. JEEVANANDHAM

Department of computer science and engineering, Angel College of Engineering and Technology Under Anna university, Chennai, djeevanandham@gmail.com

I. SARAVANAN

Department of computer science and engineering, Angel College of Engineering and Technology Under Anna university, Chennai, isaravanan@gmail.com

Follow this and additional works at: <https://www.interscience.in/ijcct>

Recommended Citation

MOHANAN, T.; MOHANATHAN, V.; JEEVANANDHAM, D.; and SARAVANAN, I. (2016) "DIGITISED HUMAN LIFE FORM ISOMORPHIC ALGORITHM," *International Journal of Computer and Communication Technology*: Vol. 7 : Iss. 3 , Article 3.

DOI: 10.47893/IJCCT.2016.1359

Available at: <https://www.interscience.in/ijcct/vol7/iss3/3>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

DIGITISED HUMAN LIFE FORM ISOMORPHIC ALGORITHM

T.MOHANAN¹, V.MOHANATHAN², D.JEEVANANDHAM³ & I.SARAVANAN⁴

^{1,2,3&4}Department of computer science and engineering, Angel College of Engineering and Technology
Under Anna university, Chennai
E-mail id : mohan.stanky@gmail.com

Abstract-ISOs are "isomorphic algorithms", which are life forms that emerged-- unplanned--from the artificial environment of the grid. Isomorphic Algorithms (better known as ISOs) are a race of programs that spontaneously evolved on the Grid, as opposed to being created by users. ISOs differ from standard programs with distinctions in their appearance and capabilities, but where they are truly unique is in their code base. While regular programs conform to the rigid structure defined by their users, ISOs have evolved, complete with a genetic code of sorts.. This inner structure of their code has allowed ISOs to develop beyond the capabilities of regular programs.. These miraculous algorithms had the capacity to evolve and change and grow at tremendous rates utilizing genetic algorithms, whereas normal programs that were intentionally written by users could only change slowly in anticipated fashions. What's REALLY important about the isomorphic algorithms id, it is revealed that indeed programs can escape the grid into the real world, essentially raising questions of what is life, sentience, the soul, etc. that humanity is no longer confined to humans, but has essentially arisen out of our digital dust and the real and digital world can become interchangeable This kind of human life form is made possible for ISOs because of digital DNA and object recognition which is made possible in isomorphic algorithms.. In this paper we are going to describe how an algorithm can be emerged into human life form.

Requestors:

- A Graph Isomorphism Algorithm for Object Recognition and creation
- DNA pattern that synchs with algorithm
- Memory and processing engine for each algorithm

Isomorphism Algorithm for Object Recognition and creation: Why do we go for Isomorphic Algorithms: In Graph Theory:

Results from experiments on a wide variety and sizes of graphs are reported. Results are also reported for experiments on recognizing graphs that represent protein molecules. The algorithm consists of three phases: preprocessing, link construction, and ambiguity resolution. Object creation is done by Pattern Recognition, Graph Matching, Labeled Graphs, Random Graphs, Local Consistency.. as graph matching in general, is a difficult task. The subclass of graph matching that we will be interested in is called graph isomorphism¹. The graph isomorphism (GI) problem is defined as follows: given two graphs, determine if a one-to-one mapping exists from the nodes of the first graph to the nodes of the second such that if two nodes in the first graph are connected by an edge, then their corresponding mapped nodes in the second graph must also be connected by an edge.- Solution:

The design of our algorithm is inspired by the Dynamic Link Architecture (DLA) [21, 22]. Given an input labeled graph and a database of model labeled graphs, the DLA constructs links from nodes in the input graph to all nodes in

the database that have the same label, regardless of which model graph they belong to. In later stages, if a similar model graph to the input graph exists in the database, the algorithm attempts to prune spurious links with the goal of leaving only the correct links connected to the recognized model graph. The compatibility of nodes is based on just their labels. Since the DLA uses just labels for determining compatibility of nodes, it can lead to erroneous results since it does not explicitly include neighborhood information. As we show below, we have enhanced this concept of local compatibility by including the neighborhood structure of the nodes to create new labels for the nodes. The algorithm then creates links according to these new labels. When a node in the first graph is compatible with (is linked to) more than one node in the second graph, we have ambiguities. The proposed algorithm then attempts to prune the incorrect links by disambiguating these ambiguous links. We have translated the local compatibility and disambiguating process into a solution to the GI problem as follows. Any two nodes, u in the first graph and v in the second one, are linked to each other if there is a similarity between their neighborhood structures of at least one level deep, i.e. there is an isomorphism between

the following subgraphs: the first consisting of node u and its children, the second consisting of node v and its children. This idea is applied recursively and is shown to be sufficient to determine the existence of any isomorphism between the two graphs being compared (except for the highly ambiguous cases to be discussed later). Since we are dealing in this paper with labeled graphs (there are labels attached to the nodes of the graphs), we refer to this problem as Labeled Graph Isomorphism (LGI)..ALG:

The Algorithm for Human structure modeling:
In these graph phase, the algorithm works with each strand of DNA pattern to construct the human structure..

Phase 0: [Preprocessing] In this phase, we compute the degrees of nodes and other related information to be included in the new labels that will be used in the following two phases. New labels are created for each node in both of the graphs. In addition to the original labels, the new labels contain the degree of the node itself and degrees and labels of each child (or immediate neighbor) of this node. To eliminate any confusion, we

use \labels" to refer to the original labels that are given with the graphs and \new labels" to denote the labels generated by the preprocessing phase.

Phase 1: [Link Construction] Links are constructed across the two graphs between all pairs of nodes that have identical new labels. An. In this figure, where original labels are assumed to be \black" or \white", node u in graph G is linked to node v in graph H . The link is constructed because the new label of u matches the new label of v , i.e. label and degree of node u match those of node v and each child of u , namely i , j , and k , has a corresponding child of node v that matches its label and degree. Here, the children i , j , and k in G correspond to the children x , y , and z in H respectively. Clearly, the graphs G and H in Figure 1 are not isomorphic.

Phase2:[Ambiguity Resolution] The resolution phase is the computationally dominant portion of the algorithm. This phase attempts to resolve ambiguities that may have resulted from the link construction phase. The goal of this phase is to try and decide whether a one-to-one mapping from nodes in the first graph to nodes in the Figure 1: Node $u \in G$ is locally compatible with node $v \in H$. second is possible.

Figure 2 shows the phases of the algorithm pictorially. After the pre-processing stage, the nodes are represented by stylized \inflated" nodes to represent the extra information contained in the new labels. Each inflated node will contain the original label, the degree of the node, and the extra information about the neighborhood around it constituting the new label. Also notice that in this example, all the nodes in G have the same new label and all the nodes in H have the same new label. In addition, the new labels are the same in both graphs because the original labels are the same and the graphs are regular graphs. Before explaining the details of the algorithm, we will define needed terminology.

ASSUMPTIONS AND DEFINITIONS

Even though our approach works for disconnected graphs, we assume that all graphs used in this paper are connected. The number of nodes in each graph is n unless mentioned otherwise. The first graph is denoted by $G = (V1; E1)$, and the second graph is denoted by $H = (V2; E2)$. Children of node u or equivalently neighbors of node u denote the set of nodes connected to u by edges. Note that most algorithms would find it hard to compare these kinds of graphs while our algorithm handles them well. Figure 2: Given graphs G and H , the three phases of the algorithm are shown. In this example, each node in G is linked to every node in H by the link construction phase, and later all the links are

disconnected after the ambiguity resolution phase since the graphs are not isomorphic. 8Definition 1 (Graph Isomorphism) Given two graphs $G = (V1; E1)$ and $H = (V2; E2)$ where $|V1| = |V2|$, is G isomorphic to H ? In other words, is there a one-to-one

mapping $f: V_1 \rightarrow V_2$ such that $(u; v) \in E_1 \iff (f(u); f(v)) \in E_2$? It has been a convention in the literature to use unlabeled and unweighted graphs with GI. We would like to expand the graphs used in GI to any graph, specifically node-labeled graphs. Therefore, we need to define the Labeled Graph

Isomorphism (LGI) problem by adding one more requirement to the definition of GI. In addition to the edges that must be compatible, we demand that the mapped nodes must have the same labels. The following is the formal definition of the LGI problem.]

Definition2(Labeled Graph Isomorphism)

Given graphs $G = (V_1; E_1)$ and $H = (V_2; E_2)$ where $|V_1| = |V_2|$ and node labels are assigned from a set of labels F by the function $l: V_1 \text{ (or } V_2) \rightarrow F$, is there a one-to-one function $f: V_1 \rightarrow V_2$ such that $l(u) = l(f(u)) \forall u \in V_1$ and $(u; v) \in E_1 \iff (f(u); f(v)) \in E_2$? The link construction phase of our algorithm may produce a many-to-many relation. Consequently, we may have ambiguities, a case which makes LGI

difficult. If the link construction phase produces a one-to-one mapping, we can safely declare the existence of an isomorphism

between the two graphs. Next, we will describe how the algorithm works, especially how the resolution process operates. The ambiguity resolution phase attempts to discover a one-to-one mapping, if it exists..

Graph Isomorphism Detection Process

As shown in Section 2.1, the first step of the algorithm embeds the DNA STRUCTURE in the nodes as new labels. So, in addition to the original label of a node u , the preprocessing phase creates a new description of the node that contains the following information: degree of u and degrees and labels of all the children of u . We will refer to the original label as label of u or $l(u)$. The link construction phase is a straightforward process. Given the information produced in the preprocessing phase, any two nodes: u in the graph G and v in the graph H , are linked if u is locally compatible with v . All possible links are constructed in this manner. The resolution phase starts with the following check: if the links from the link construction phase result in a one-to-one

mapping, then we stop and declare the existence of an isomorphism. Otherwise we go through the disambiguating process. We find the first node, say u , where $\text{degAmbig}(u) = g > 1$ and compose the corresponding locally compatible set B and the ambiguous set A that includes u . We perform another test here to make sure that an isomorphism is still feasible. A necessary condition for an isomorphism to exist is that the sizes of the ambiguous set A and the corresponding locally compatible set B must be equal and the links from nodes in A must

be connected exclusively to the nodes in B . Therefore, our test is based on making sure that

A is of size g and that the locally compatible set corresponding to

each node in A is in fact B. If this is not the case, we stop and declare the absence of any isomorphism since the necessary condition is not satisfied. Otherwise, we try to find a one-to-one mapping between nodes in A and nodes in B by resolving the ambiguities. Three outcomes are possible after the resolution process has been applied to A and B. First, if a one-to-one mapping cannot be produced, then we stop and declare the absence of an isomorphism between the two graphs.

The second possibility is that we are able to find a one-to-one mapping unambiguously between nodes in A and nodes in B. Third, it is possible that each node in A can be mapped to any node

in B, a case of full ambiguity that may lead at the end of the algorithm in the inability to conform the existence of isomorphism between the two graphs (details are given later); a flag is set to reflect this case. If the outcome is one of the latter two, then the degree of ambiguity of all nodes in A is set to 1 and the algorithm continues. If we need to continue, we find the next node with $\text{degAmbig} > 1$ and repeat the process. Figure 4 shows an example where $\text{degAmbig}(u) = 2$ and an isomorphism is still feasible while processing the ambiguous set $A = \{u, v\}$; $B = \{x, y\}$.

Figure 4: Shows that $\text{degAmbig}(u) = 2 = \text{degAmbig}(v)$. The ambiguous set $A = \{u, v\}$ and the corresponding locally compatible set $B = \{x, y\}$.

ACKNOWLEDGMENT:

This is where the DNA pattern is given a human structure..

DNA PATTERN:

1. DNA to construct a biological human race with artificial intelligence only results in creation of artificial, designed nanostructures out of nucleic acids, such as this DNA tetrahedron. Each edge of the tetrahedron is a 20 base pair DNA double helix, and each vertex is a three-arm junction. The 4 DNA strands that form the 4 tetrahedral faces are color coded. 2. Each strand consists of millions of nodes... 3. The nodes from the dimensional strands are made up of reflector semi partialised parts and trillions of instructions to process a function.

PROCESSOR:

On developing an e Hierarchical Processor and Memory Architecture: A Petaflop Point Design Study The objective of this project was to investigate the feasibility of a new architecture concept that can yield very high performance, in the range of 100 Tflops, at a reasonable cost. The proposed Hierarchical Processor And Memory architecture (HPAM) consists of a heterogeneous collection of processors organized as a multilevel hierarchy. Each level of the hierarchy has a different implementation and is intended to efficiently execute appropriate portions of the application. The higher levels can efficiently execute serial and low parallelism sections of the application. The low levels of the hierarchy have a large number of slower processors and can efficiently execute portions of the code with high degree of parallelism. This work entailed developing the initial infrastructure of a simulator (HPAM-Sim) and conducting in-depth studies of complete industrial applications as well as benchmark applications.

SOLUTION TO THE PROCESSOR:

The Silicon Graphics Fuel visual workstation maximizes the performance of your desktop applications while offering you unprecedented price/performance value. A premium blend of industry-leading technology from SGI, Silicon Graphics Fuel features the latest MIPS® R16000™ processor and the unparalleled VPro™ 3D graphics system for IRIX® in a new high-bandwidth architecture. Drive your creativity and productivity to a new level with Silicon Graphics Fuel. Tezro is powered by up to four MIPS® processors in an advanced high-bandwidth architecture leveraged from the SGI® 3000 family of high-performance supercomputers and delivers industry-leading visualization, digital media, and I/O connectivity on the desktop. Tezro is designed to help topflight individuals and teams deliver cutting-edge results in ever-shorter production cycles--

whether the algorithm is a innovative designer, scientist, engineer, defense specialist, film producer, or geophysicist as same as in real world..

MEMORY ARCHITECHTURE:

Memory in digitized human life form is always represented as memory cycles where each cycle should be the least time due by comparison with their processor cycles...

SOLUTION:

Memory architecture is merged with processor base configuration.. Thus memory cycles are synchronized with processor throughput cycle and this gives an edge where processor would not have to work hardly on retrieving and storing on memory rather it would concentrate on expanding DNAs instructions..

STRUCTURE:

- Memory is totally inter related with processors working.. so, it is planned to obtain a structure with nano particles for storage..
- Nanoparticle Design:

APPLICATIONS:

Processor and memory cycle with female interface..

Preformed human architecture without graphical interface.

Super massive system building structure..

Core Building formation

REFERENCES:

- [1]. ABADI, D. J., MADDEN, S. R., AND FERREIRA, M. C. Integrating compression and execution in column-oriented database systems. Proc. of SIGMOD (2006).
- [2]. FRENCH, C. D. One size fits all database architectures do not work for DSS. In Proc. Of SIGMOD (May 1995), pp.449.450.
- [3]. <http://books.couchdb.org/relax/intro/eventual-consistency>
- [4]. <http://www.rackspacecloud.com/blog/2009/11/09/NOSQL-ecosystem/> Jonathan Ellis.
- [5]. <http://blog.evanweaver.com/articles/2009/07/06/up-and-running-with-cassandra>
- [6]. DEAN, J., AND GHEMAWAT, S. MapReduce: Simplified data processing on large clusters. In Proc. of the 6th OSDI (Dec. 2004), pp. 137.150.
- [7]. DEWITT, D., KATZ, R., OLKEN, F., SHAPIRO, L., STONEBRAKER, M., AND WOOD, D. Implementation techniques for main memory database systems. In Proc. of SIGMOD (June 1984), pp. 1.8.