AISB/IACAP World Congress 2012

Birmingham, UK, 2-6 July 2012

1st Symposium on Nature Inspired Computation and Applications (NICA)

Dr Ed Keedwell (Editor)

Part of

Published by
The Society for the Study of
Artificial Intelligence and
Simulation of Behaviour

http://www.aisb.org.uk

ISBN 978-1-908187-25-3

Foreword from the Congress Chairs

For the Turing year 2012, AISB (The Society for the Study of Artificial Intelligence and Simulation of Behaviour) and IACAP (The International Association for Computing and Philosophy) merged their annual symposia/conferences to form the AISB/IACAP World Congress. The congress took place 2–6 July 2012 at the University of Birmingham, UK.

The Congress was inspired by a desire to honour Alan Turing, and by the broad and deep significance of Turing's work to AI, the philosophical ramifications of computing, and philosophy and computing more generally. The Congress was one of the events forming the Alan Turing Year.

The Congress consisted mainly of a number of collocated Symposia on specific research areas, together with six invited Plenary Talks. All papers other than the Plenaries were given within Symposia. This format is perfect for encouraging new dialogue and collaboration both within and between research areas.

This volume forms the proceedings of one of the component symposia. We are most grateful to the organizers of the Symposium for their hard work in creating it, attracting papers, doing the necessary reviewing, defining an exciting programme for the symposium, and compiling this volume. We also thank them for their flexibility and patience concerning the complex matter of fitting all the symposia and other events into the Congress week.

John Barnden (Computer Science, University of Birmingham) Programme Co-Chair and AISB Vice-Chair

Anthony Beavers (University of Evansville, Indiana, USA)
Programme Co-Chair and IACAP President

Manfred Kerber (Computer Science, University of Birmingham)
Local Arrangements Chair

1st Symposium on Nature Inspired Computation and Applications (NICA)

Preface

The symposium on Nature Inspired Computation and Applications (NICA) is the first of its kind at the AISB convention as it incorporates the whole field of naturally inspired computation. However, it does take inspiration from previous successful symposia that focussed on particular algorithms, including evolutionary algorithms and swarm intelligence. The papers in this symposium demonstrate the sheer breadth of research in this topic in AI, with a wide variety of algorithms covered (Bayesian algorithms, evolutionary algorithms, neural networks and herd behaviour) and applications proposed. It is clear from this breadth of research that Nature-Inspired Computation has the potential to impact many areas of everyday life for the foreseeable future.

Ed Keedwell

Table of Contents

SpendInsight: some remarks on deploying an intelligent spend-analysis system

Richard W. Barraclough
@UK PLC, 5 Jupiter House, Calleva Park
Aldermaston, Reading, Berkshire, RG7 8NN.
richard.barraclough@ukplc.net

J. Mark Bishop & Sebastian Danicic
Dept. Computing, Goldsmiths, University of London,
New Cross, London SE14 6NW, UK.
m.bishop, s.danicic@gold.ac.uk

Slawomir J. Nasuto & Richard J. Mitchell
Cybernetics Research Group, School of Systems Engineering
University of Reading, Reading, Berks, UK.
s.j.nasuto, r.j.mitchell@reading.ac.uk

May 20, 2012

Abstract

A three way collaboration between industry and two UK universities led to the development of an intelligent spend analysis system. In this paper we outline how a novel combination of the 'Decision Tree' and 'Bayesian Classifier' algorithms[1], working with 'big data' on a real world e-procurement problem, led to the development of sophisticated, A.I. driven, intelligent spend analysis software, subsequently commercially marketed throughout the UK as the 'SpendInsight' system; a system recently deployed by the UK-National Audit Office (NAO) to highlight potential savings to UK-National Health Service (NHS) procurement of £500 million per annum [14].

We subsequently investigate how deep-rooted 'institutional inertia' can often work to inhibit the full realisation of the potential economic benefits to an organisation that should accrue from the deployment of intelligent spend analysis. Finally we demonstrate how, by linking the institutional roll-out of intelligent spend analysis to an overarching green policy agenda, such barriers may be overcome. We conclude the paper by showing (perhaps counter-intuitively) that at the societal level a strong green policy agenda may realise significant benefit for both the environment and the economy.

[1]Two algorithms identified in the CFP to be of specific relevance to the 2012 NICA (Nature Inspired Computing Applications) symposium.

1 Introduction

Three linked Knowledge Transfer Partnerships (KTP) between the University of Reading, Goldsmiths College and @UK PLC — a leading cloud-based electronic marketplace provider — have produced a system named SpendInsight. This system uses various Artificial Intelligence (AI) techniques to enable e-procurers to analyse their purchases and identify potentially significant savings. As an added benefit, it is also possible to estimate the carbon footprint of products and so develop an envi-

ronmentally friendly procurement policy.

The level of automation in the spend analysis system makes it fundamentally different from competing providers, and this translates into a number of unique selling points:

- Firstly, the fast speed of the system shortens the time from analysis to report from months to days, and allows analyses to be performed more frequently. The customer is therefore better equipped to react to changing market conditions, monitor purchasing behaviour, and assess the effectiveness of procurement policies.
- Secondly, the high level of automation allows the system to produce reports in unprecedented detail. This allows the customer to drill from high level management reports right down to the original purchase order and invoice data used to generate the report. This visibility of original data provides the accountability necessary to identify genuine savings opportunities, quantify them accurately, and substantiate conclusions reached from the analysis.
- Thirdly, detailed reports empower procurement professionals to draw their own conclusions about their own data, removing any need for expensive external consultancy.

2 Background

During the three year period of development of the SpendInsight project, as a result of data processing carried out in development, opportunities arose which allowed data to be obtained about procurements in the NHS. The application for the project subsequently focused towards analysis of spend for e-procurement for the NHS.

One component of the project focused on matching companies and products identified in the spend analysis, a second component, classification, had to work on the product data so returned [11], and a third component, ranking, focused on automatic detection of attribute data in textual descriptions of products [1].

All three components worked together to form an integrated system, which has been named SpendInsight. Key to the system is the ability to classify vast numbers of different products from a variety of suppliers and hence determine equivalent products from different suppliers. Given this, it is then possible to assess the economic cost of each product and hence choose the cheapest.

The core system first went live in 2007 when the company created a repository of company and product information and, importantly, a system for identifying duplicate companies and products. Commercial opportunities for the de-duplication technology were subsequently developed which, in turn, meant that the de-duplication system became an important technology which needed to scale with the overall system, whilst maintaining traceability from input-data to output.

The scalable de-duplication technology enabled the deployment of a large-scale spend analysis solution across NHS trusts in London; this work highlighted potentially large scale savings in NHS purchasing. This result was subsequently independently affirmed in the National Audit Office report of February 2011 [14], which concludes that, rolled out across all NHS trusts in England, annual savings of £500 million pounds could be made (over 10% of NHS spending on consumables). In addition, using related 'GreenInsight' technology, the 'environmental' cost of each classified product can also be allocated, *i.e.,* GreenInsight enables e-procurers to assess the environmental cost of their purchases.

3 De-duplication

The core approach to de-duplication is to use two staging databases for the input data, at two levels of granularity. At the most detailed level are purchase orders and invoices, and each purchase order line and invoice line can be traced back to lines in clients' data files — which are typically received in CSV format.

The first challenge, then, is data integration. First, data must be extracted from diverse client systems. Although normally delivered in CSV files, the number of files, the columns in the files, and relations between the files are typically peculiar to each client. (It is the first author's experience that no two installations of the Oracle 'iProcurement' system are the same).

Once received, data must be stored in a single unified schema to allow it to be queried. However, this is insufficient to be able to generate useful reports; at this point

the duplication problem becomes apparent. Even within the finance system of a single organisation a single supplier may appear more than once — each occurrence with a subtly different name, *e.g.,* 'Limited' vs. 'Ltd.' Of course the problem is exaggerated when comparing across organisations. Furthermore, one must first successfully identify multiple instances of the same supplier before proceeding to the harder problem of identifying the unique products they sell.

3.1 Hierarchy of abstraction

When data files are received they are first loaded into a 'raw data' database. Each file is scanned and an SQL table definition statement is created for the file. The table is created and the file is loaded into it. This allows data types to be determined for each column, and to check any referential integrity constraints between the files. For example, it may turn out that a purchase order in one file gives an ID for a supplier but that ID does not exist in the supplier file.

The second step is to transform the raw data into the standard format. This can be as simple as specifying a map between the columns in the client's file and the columns supported in the system. However, more elaborate queries may need to be developed — particularly when relations between data in the input files must be used. In very rare cases it is necessary to pre-process the clients' data files — for example when data rows are interleaved with 'sub-total' rows.

The first level of abstraction models companies and the products they supply. In this model many purchase order lines may 'point' at the same product, and in turn a product 'points' at a supplier. This abstraction is key to achieving scalability. From this model the system builds a 'cleansed view' of the database which is used for driving reporting for clients. In the cleansed view the suppliers have been de-duplicated and, in turn, so have the products they supply.

The cleansed view maintains an 'audit trail' of the matching performed and the evidence upon which matching was based. This is important if clients query results because the results can always be traced back to the original data. In the final reporting, clients can see how suppliers and products have been matched, and can supply feedback to the system by identifying false-positive matches and additional matches. Because the 'cleansed view' is separate from the data itself, there is a complete 'audit trail.'

3.2 Rule engine

The cleansed view is built by a rule engine. All of the rules are applied, iteratively, until the system stabilises. Each rule may use information in both of the staging databases and in the partially built cleansed view (a kind of feedback loop), and the rule may make use of additional custom indices built on these data.

The staging databases have increased in size tenfold over the last three years, but the processing time to build the cleansed view has not increased significantly. Typically, using current technology the cleansed view can be re-built from scratch in under one week.

4 Automatic classification

The automatic classification is the other substantial component of the system. This was developed in parallel to the de-duplication technology. The core technology allows procurers to identify and cost 'equivalent' products, with an extension offering 'carbon analysis' of purchasing decisions enabling procurers to analyse both the economic and environmental cost of purchases.

Text classification is the task of predicting the class of a previously-unseen document based upon its words. The relationship between words and class is learnt from a labelled training set. Since the 1960s, many methods have been proposed, including decision rule classifiers [4], decision trees [15], k-nearest neighbour [16], Naïve Bayes [8], neural networks [13], regression models [17], Rocchio [5], the support vector machine (SVM) [7] and winno [2]. For this work the classification task is to assign each product in the cleansed view into one of about 2,000 different classes. The main data source upon which the classification task draws is the free-text descriptions on purchase order lines. In the cleansed view there may be hundreds of different descriptions for the same product.

Naïve Bayes is a probabilistic classifier that has been used since the early 1960s [10]. It has advantages over other classifiers in its simplicity, learning speed, classification speed, and storage space requirements [3]. In

the multi-variate Bernoulli event model, a document is a binary vector over the space of words [10] *i.e.,* each dimension of the space corresponds to a word. The words are assumed to be dependent only on the class to which their document belongs — an assumption which clearly is false. This is the naïve step. Nevertheless, Naïve Bayes has empirically outperformed many other algorithms [6, 9, 10, 3].

In our work the descriptions for each product are reduced to a *bag of words* and a set of manually classified *training data* is used to calculate the conditional probability of a word belonging to (a product in) each class. An application of Bayes Theorem gives the conditional probability of a class given a word; thus words and, in turn, products can be classified.

While in principle the Naïve Bayes classifier can be applied to the whole data set, in practice the applicability is limited by the training data set. If there is little or no training data for a particular class then the probability of the class being chosen is small. In one example we found 'bone granules' classified as food because the only 'granulated' product in the training data was gravy.

The training data set has been expanded considerably since the start of the project, and wherever possible assimilates clients' classified data sets. However, here we have the additional problem of deciding whether we believe that clients' data has been accurately classified. We found useful heuristics for this decision problem to include the proportion of products classified to non-existent classes, and the internal consistency of the classification, *i.e.,* to how many different classes has the same product been assigned.

Even when clients' data is not suitable for use as training data, it can still be used to provide 'additional guidance.' Two kinds of extra guidance data are used by the SpendInsight system. Both make use of the three-level hierarchy of the 2,000 classes. The first is product type: the system may deduce that a certain product should be classified into a specific level 1 or 2 class. Such deductions can be made where a product has previously been classified inconsistently between different clients or sources. This extra information is used to limit the classes into which the Naïve Bayes algorithm may assign the product. The second type of extra guidance is at supplier level: the system may either deduce [or be told] that a specific supplier either only supplies products in or does not supply products in certain level 1 classes. Such information is typically used to limit the classifications permitted by the Naïve Bayes algorithm.

The second major *NICA* algorithm employed for classification is the 'Decision Tree' which is a divide and conquer approach. The node at the root of the tree divides the training set into two subsets. In text classification, one set usually contains those training documents with a certain word, and the other contains those without. Both of these subsets are further split at the root's children, further split at the root's grandchildren, and so forth down the tree. A leaf is usually formed where the set of training documents are all of the same class, and that class is assigned to the leaf as a label.

One of the most popular decision tree algorithms is Quinlan's C4.5 [15] which uses an error-based pruning algorithm.

In our work the Decision Trees are taught from the training data by the symbolic rule induction system (SRIS). Decision Trees have been built for only 19 classes, all of which have been carefully selected (by hand) to have good supporting training data and be among the classes that are more highly visible to clients. Building and testing a decision tree can take many weeks, which limits the rate at which they can be added to the system.

Finally, for each product the classification system must decide between the various candidate classes suggested by each of the algorithms. Essentially, there is a trade-off between the accuracy of a method and the proportion of the dataset to which it can be applied. For example Naïve Bayes is the least accurate method but can be applied to the whole of the data set, whereas a Decision Trees can be much more accurate but applied only to relatively few classes. The total time taken to apply classification is dominated by the Naïve Bayes algorithm, which can classify about 100 descriptions per second.

5 Intelligent spend-analysis — barriers to full impact: changing purchasing behaviour

At the launch of the software industry standard for green data interchange (RSA London, 15/11/11) Ronald Duncan of @UK PLC reported how AI technology in classifi-

cation and matching was exploited in a new 'spend analysis' system co- developed with the University of Reading and Goldsmiths College.

By highlighting how equivalent products can be bought at the cheapest available price, the SpendInsight software was successfully deployed by the UK NAO[2] to generate the evidence base for its recent report [14] assaying huge potentially savings (of around £500 million per annum) to UK NHS procurement[3]. The fall-out from this research prompting widespread coverage in the UK news media[4] and a subsequent exposé by the BBC radio 'File on Four' programme detailing inefficiencies in NHS procurement[5], with the impact of the work subsequently meriting discussion in the UK parliament[6]. And yet, perhaps surprisingly given the current economic climate, data in the NAO report details only 61 of UK Health Trusts currently deploying the SpendInsight system.

At first sight the huge potential savings highlight by intelligent spend-analysis software might imply organisations would rush to embrace intelligent spend analysis technology, however @UK PLC experience in deploying the SpendInsight system with the NHS procurement suggest that this is not always the case; factors other than raw cost are often important in institutionalised purchasing environments.

Thus, results from a recent survey[7] of attitudes to individual and organisation change, amongst 229 civil servants involved in the purchasing process, clearly showed that whilst individual buying behaviour is strongly correlated with cost, only 57% of the respondents reported cost criteria alone were enough to 'probably or definitely' change organisational purchasing decisions. Conversely, the results suggested that by linking economic savings with improved sustainability (*e.g.,* a lower carbon footprint), there was a significant increase, with 84% of respondents assessing that their organisation would 'probably or definitely' change purchasing choices[8].

6 Conclusions

The survey of purchasing behaviour reported in this paper unambiguously demonstrated that sustainability issues are a strong motivating factor in changing buying behaviour; linking economic and green spend analysis may speed up and unblock process change within an organisation. Clearly, given the large potential savings identified by *intelligent spend analysis* compared with the relatively small incremental cost (per product) of carbon off-setting, serious consideration of green issues can easily result in substantial economic benefit to an organisation. Thus, at the societal level, a strong green policy agenda may realise significant benefit for both the environment and the economy.

References

[1] Brown, M. J. Automatic production of property structure from natural language, PhD Thesis, University of Reading 2011.

[2] Dagan, I., Karov, Y. and Roth, D. Mistake-driven learning in text categorization. In Claire Cardie and Ralph Weischedel, editors, Proceedings of

[2] The NAO report states, "@UK PLC *uses its in-house artificial intelligence system to classify every purchase order line raised by the trust in a twelve month period to a unique product code. This system also extracts information on supplier, cost, date and quantity ordered.*"

[3] "*...accounting for £4.6 billion in expenditure, using a conservative estimate of 10 per cent, savings of around £500 million could be made.*"

[4] BBC News:
<http://www.bbc.co.uk/news/health-14971984>
< http://www.bbc.co.uk/news/health-12338984>.

[5] BBC Radio File on 4:
<http://downloads.bbc.co.uk/podcasts/radio4/fileon4/fileon4_20110927-2045a.mp3>.

[6] House of Commons Committee of Public Accounts, Formal Minutes Session 2010–12, "TUESDAY 15 March 2011. Members present: Mrs Margaret Hodge, in the Chair, Mr Richard Bacon, Stephen Barclay, Matthew Hancock, Jo Johnson, Mrs Anne McGuire, Austin Mitchell, Nick Smith, Ian Swales, James Wharton. Agenda item (1) NHS Trust Procurement. Amyas Morse, Comptroller and Auditor General, Gabrielle Cohen, Assistant Auditor General and Mark Davies, Director, National Audit Office were in attendance. The Comptroller and Auditor General's Report NHS Trust Procurement was considered. Sir David Nicholson KCB CBE, Chief Executive, NHS, David Flory CBE, Deputy Chief, Executive, NHS, Peter Coates CBE, Commercial Director, and Howard Rolfe, Procurement Director, gave oral evidence (HC 875-i). [Adjourned till Wednesday 16 March at 15.00pm]."

[7] Survey carried out for @UK PLC at the 'Civil Service Live' conference, Olympia 7-9 July, 2011.

[8] Further support for this finding is that the result aligns well with the experience of @UK PLC in the rollout of the 'GreenInsight' [environmental analysis] software, as part of the NHS 'Carbon Footprint project'; at an organisational purchasing level, the GreenInsight software provides a fully automated environmental analysis.

EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing, pages 55 -63, Providence, US, 1997. Association for Computational Linguistics, Morristown, US.

[3] Domingos, P. and Pazzani, M. J. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, 29(2–3):103–130, 1997.

[4] Furnkranz, J. and Widmer, G. Incremental reduced error pruning. In Proceedings of ICML-94, the 11th International Conference on Machine Learning, pages 70–77, New Brunswick, NJ, 1994.

[5] Ittner, D. J., Lewis, D. D. and Ahn, D. Text categorization of low quality images. In Proceedings of SDAIR-95, the 4th Annual Symposium on Document Analysis and Information Retrieval, pages 301–315, Las Vegas, US, 1995.

[6] Joachims, T. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Douglas H. Fisher, editor, Proceedings of ICML-97, the 14th International Conference on Machine Learning, pages 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[7] Joachims, T. Learning to Classify Text Using Support Vector Machines. Kluwer Academic Publishers, 2001.

[8] Lewis, D. D. Naïve (Bayes) at forty: The independence assumption in information retrieval. In Claire Nedellec and Celine Rouveirol, editors, Proceedings of ECML-98, 10th European Conference on Machine Learning, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[9] Lewis, D. D. and Ringuette, M. A comparison of two learning algorithms for text categorization. In Proceedings of SDAIR-94, the 3rd Annual Symposium on Document Analysis and Information Retrieval, pages 81–93, Las Vegas, US, 1994.

[10] McCallum A. and Nigam K. A comparison of event models for naive Bayes text classification. In Proceedings of AAAI-98, the 15th National Conference on Artificial Intelligence, 1998.

[11] Roberts, P. J. Automatic Product Classification. PhD thesis, University of Reading, 2011.

[12] Roberts, P. J. et al.,Identifying problematic classes in text classification, Proceedings of 9th IEEE International Conference on Cybernetic Intelligent Systems (CIS), 2010.

[13] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6):386–408, 1958.

[14] Report by the Comptroller and Auditor General, National Audit Office: the procurement of consumables by NHS acute and Foundation trusts, HC 705 Session 2010–2011, 2 February 2011, Department of Health, UK.

[15] Quinlan, J.R. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[16] Yang, Y. Expert network: effective and efficient learning from human decisions in text categorization and retrieval In Proceedings of SIGIR-94, the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 13–22, 1994.

[17] Yang, Y. and Chute, C.G. A linear least squares fit mapping method for information retrieval from natural language texts. In Proceedings of COLING-92, the 14th Conference on Computational Linguistics, pages 447–453, 1992.

Evolution of unknotting strategies for knots and braids

Nicholas Jackson[1] and **Colin G. Johnson**[2]

Abstract. This paper explores the problem of *unknotting* closed braids and classical knots in mathematical knot theory. We apply evolutionary computation methods to learn sequences of moves that simplify knot diagrams, and show that this can be effective both when the evolution is carried out for individual knots and when a generic sequence of moves is evolved for a set of knots.

1 INTRODUCTION

1.1 Knots and links

Knot theory is currently one of the richest and most vibrant areas of pure mathematics, having connections not only with other topics in algebraic and geometric topology, but also with many other branches of mathematics, as well as mathematical physics [20] and biochemistry [18].

A full introduction to the study of knots and links is beyond the scope of this article, but a readable introduction may be found in, for example, the book by Cromwell [8], and a more comprehensive but still accessible survey in Rolfsen's classic text [17].

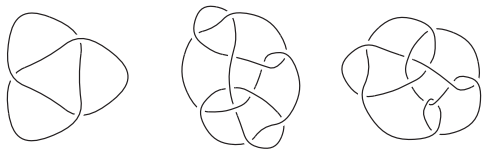


Figure 1. Examples of knots: the trefoil (3_1), Conway's knot ($11n34$) and the Kinoshita–Terasaka knot ($11n42$)

We define a *knot* to be an isotopy class of embeddings $K\colon S^1 \hookrightarrow \mathbb{R}^3$, where $S^1 = \{e^{i\theta} \in \mathbb{C} : 0 \leqslant \theta < 2\pi\}$ denotes the standard unit circle; informally, this is a set of placements of a closed circle in space. A *link* is a knot with more than one circular component, that is, an (isotopy class of an) embedding $L\colon S^1 \sqcup \ldots \sqcup S^1 \hookrightarrow \mathbb{R}^3$. (Alternatively, a knot may be regarded as a link with a single component.)

We generally represent knots and links with diagrams in the plane: projections of the embedded circle(s) where each double intersection point is equipped with crossing information, and we disallow cusps, tangencies or triple-points. Examples may be seen in Figure 1. Identifiers such as 3_1 and 8_{17} refer to the table in Rolfsen's book [17], while identifiers of the form $11a367$ and $11n34$ refer to, respectively, alternating and non-alternating knots in the census of Hoste, Thistlethwaite and Weeks [10].

[1] Mathematics Institute, University of Warwick, Coventry CV4 7AL, UK. Email: Nicholas.Jackson@warwick.ac.uk

[2] School of Computing, University of Kent, Canterbury CT2 7NF, UK. Email: C.G.Johnson@kent.ac.uk

Isotopy of embeddings descends to certain allowable local moves on diagrams which were first studied by Reidemeister [16] and by Alexander and Briggs [2]. These *Reidemeister moves* are depicted in Figure 2. Two knots or links are isotopic if and only if their diagrams are connected by a finite sequence of Reidemeister moves and continuous deformations of the ambient projection plane.

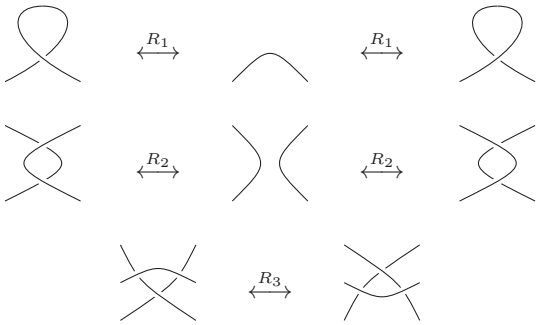


Figure 2. Reidemeister moves

There are a number of different measures of the complexity of a given knot or link K, the best known of which is the *crossing number*: the minimal number of crossings over all possible diagrams for K. Related to this is the *unknotting number* $u(K)$: the minimal number, over all possible diagrams for a knot, of crossings which must be changed in order to obtain a trivial knot. The trefoil in Figure 1 has unknotting number $u(3_1) = 1$: it may be seen to be nontrivially knotted (that is, not isotopic to an unknotted circle) but changing any single crossing results in a diagram which may be transformed (by means of an R_2 move followed by an R_1 move) into an unknotted circle. Less obviously, the other two knots in Figure 1 are also known to have unknotting number 1.

The unknotting number $u(K)$ is a conceptually simple measure of the complexity of a given knot K (broadly speaking, the higher the unknotting number, the more knotted the knot in question) but one which is often not straightforward to calculate. According to the *KnotInfo* database [12], the unknotting number is currently unknown for nine of the 165 prime knots with ten crossings (and also for many knots of higher crossing number); for these nine knots, the unknotting number is is known to be either 2 or 3, due at least in part to work on Heegaard Floer homology by Oszváth and Szabó [15] which ruled out unknotting number 1. Recent work by Borodzik and Friedl [6] has introduced a new invariant which provides a lower bound of 3 for the unknotting number of twenty-five otherwise difficult cases of knots with up to twelve crossings.

In this paper, our main goal is not necessarily to find optimal

bounds on the unknotting numbers of currently unresolved cases, but to explore the possibilities afforded by applying evolutionary computing techniques to pure mathematical problems in group theory and geometric topology, and to try to obtain some qualitative understanding of the search landscape for these problems. The unknotting problem is relatively straightforward to describe and implement, and thus provides a good candidate for a preliminary investigation of this type.

1.2 Braids

The theory of braids was first seriously investigated by Artin [3], and again a full treatment is far beyond the scope of this article, so we will discuss only those aspects essential for what follows. A more comprehensive discussion may be found in either the book by Hansen [9] or the classic monograph by Birman [5].

We define a geometric *braid* on n strings to be a system of n disjoint, embedded arcs $A = \{A_1, \ldots, A_n\}$ in $\mathbb{R}^2 \times [0,1]$, such that the arc A_i joins the point $P_i = (i, 0, 1)$ to the point $Q_{\tau(i)} = (\tau(i), 0, 0)$, where τ denotes some permutation of the numbers $\{1, \ldots, n\}$, such that each arc A_i intersects the intermediate plane $\mathbb{R}^2 \times \{z\}$ exactly once, for all $0 < z < 1$. Figure 3 shows an example of a 4–string braid.

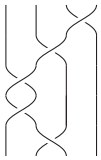


Figure 3. A 4–string braid

The *elementary braid* σ_i, for $1 \leqslant i \leqslant n-1$, is the n–string braid in which the $(i+1)$st string crosses over the ith string, and no other interactions take place; its inverse σ_i^{-1} is the braid in which the ith string crosses over the $(i+1)$st string (see Figure 4). Any n–string braid β may be represented (although not, in general, uniquely) as a concatenated sequence of elementary braids.

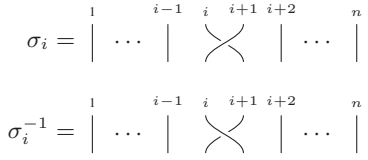


Figure 4. The elementary braids σ_i and σ_i^{-1}

We consider two n–braids β_1 and β_2 to be equivalent if they are related by an isotopy which keeps their endpoints fixed. In the language of elementary braids, this translates to the following identities:

$$\sigma_i \sigma_j = \sigma_j \sigma_i \qquad \text{for } |i-j| > 1 \tag{1}$$

$$\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1} \qquad \text{for } 1 \leqslant i \leqslant n-1 \tag{2}$$

Geometrically, the first of these corresponds to moving two non-interacting elementary braids past each other, and the second is essentially the third Reidemeister move R_3.

We may define the *braid group* B_n by the following presentation:

$$\left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i \sigma_j = \sigma_j \sigma_i & |i-j| > 1 \\ \sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1} & 1 \leqslant i \leqslant n-1 \end{array} \right\rangle \tag{3}$$

It can be shown that the group defined by this presentation is isomorphic to the group obtained by imposing the obvious concatenation operation on the (in general, infinite) set of all n–string geometric braids. In this latter group, the identity element is the trivial n–braid (the one with no crossings) and for any braid β, the inverse β^{-1} may be obtained by reflecting β in the horizontal plane $\mathbb{R}^2 \times \left\{\frac{1}{2}\right\}$.

There are other, equivalent constructions of the n–string braid group, including one in terms of the fundamental group of a particular configuration space factored by an action of the symmetric group S_n but these will not concern us here.

Given a braid $\beta \in B_n$, we can obtain a link $\widehat{\beta}$ by the *closure* operation depicted in Figure 5, that is, we join each point $P_i = (i, 0, 1)$ to the point $Q_i = (i, 0, 0)$. In fact, Alexander's Theorem [1] (see also Birman [5, Theorem 2.1]) states that any knot or link can be obtained in this way; an explicit algorithm may be found in the paper by Vogel [19]. Note that a closed-braid presentation need not be minimal with respect to the crossing number of the knot. That is, an n–crossing knot might not have a closed-braid presentation with n crossings. Table 1 lists several examples of non-minimal presentations.

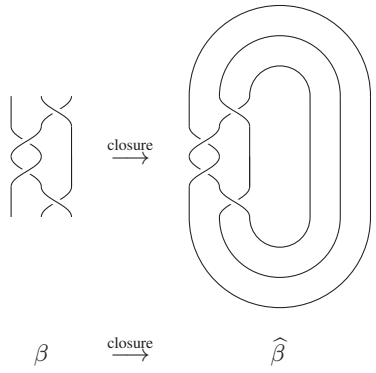


Figure 5. The closure operation on braids

The following theorem gives explicit conditions for when two different braids yield isotopic knots or links. This result was due originally to Markov, although a full proof was only published some years later by Birman [5, Theorem 2.3] (see also the paper by Rourke and Lambropoulou [11]).

Theorem 1 (Markov [14])**.** *Two braids $\beta_1 \in B_m$ and $\beta_2 \in B_n$ yield closures $\widehat{\beta_1}$ and $\widehat{\beta_2}$ which are isotopic as links if and only if β_1 and β_2 are connected by a finite sequence of moves of type M_1 (conjugation) and M_2 (stabilisation), as depicted in Figure 6.*

2 UNKNOTTING

We now attempt to use evolutionary techniques to devise optimal unknotting strategies for knots and links. The theorems of Alexander and Markov enable us to represent knots and links as words in the standard generators σ_i of the braid group B_n for some n. The

K	Braid word	Strands	Crossings
3_1	σ_1^3	2	3
4_1	$\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}$	3	4
5_1	σ_1^5	2	5
5_2	$\sigma_1^3\sigma_2\sigma_1^{-1}\sigma_2$	3	6
6_1	$\sigma_1^2\sigma_2\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_3^{-1}$	4	7
6_2	$\sigma_1^3\sigma_2^{-1}\sigma_1\sigma_2^{-1}$	3	6
6_3	$\sigma_1^2\sigma_2^{-1}\sigma_1\sigma_2^{-2}$	3	6
7_1	σ_1^7	2	7
7_2	$\sigma_1^3\sigma_2\sigma_1^{-1}\sigma_2\sigma_3\sigma_2^{-1}\sigma_3$	4	9
7_3	$\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1\sigma_2\sigma_1^{-1}\sigma_2$	3	8
7_4	$\sigma_1^2\sigma_2\sigma_1^{-1}\sigma_2^2\sigma_3\sigma_2^{-1}\sigma_3$	4	9
7_5	$\sigma_1^4\sigma_2\sigma_1^{-1}\sigma_2^2$	3	8
7_6	$\sigma_1^2\sigma_2^{-1}\sigma_1\sigma_3\sigma_2^{-1}\sigma_3$	4	7
7_7	$\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_3\sigma_2^{-1}\sigma_3$	4	7
8_1	$\sigma_1^2\sigma_2\sigma_1^{-1}\sigma_2\sigma_3\sigma_2^{-1}\sigma_4^{-1}\sigma_3\sigma_4^{-1}$	5	10
8_2	$\sigma_1^5\sigma_2^{-1}\sigma_1\sigma_2^{-1}$	3	8
8_3	$\sigma_1^2\sigma_2\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_3^{-1}\sigma_4^{-1}\sigma_3\sigma_4^{-1}$	5	10
8_4	$\sigma_1^3\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_3^{-1}\sigma_2\sigma_3^{-1}$	4	9
8_5	$\sigma_1^3\sigma_2^{-1}\sigma_1^3\sigma_2^{-1}$	3	8
8_6	$\sigma_1^4\sigma_2\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_3^{-1}$	4	9
8_7	$\sigma_1^4\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_2^{-1}$	3	8
8_8	$\sigma_1^3\sigma_2\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_3^{-2}$	4	9
8_9	$\sigma_1^3\sigma_2^{-1}\sigma_1\sigma_2^{-3}$	3	8
8_{10}	$\sigma_1^3\sigma_2^{-1}\sigma_1^2\sigma_2^{-2}$	3	8
8_{11}	$\sigma_1^2\sigma_2\sigma_1^{-1}\sigma_2^2\sigma_3^{-1}\sigma_2\sigma_3^{-1}$	4	9
8_{12}	$\sigma_1\sigma_2^{-1}\sigma_1\sigma_3\sigma_2^{-1}\sigma_4^{-1}\sigma_3\sigma_4^{-1}$	5	8
8_{13}	$\sigma_1^2\sigma_2^{-1}\sigma_1\sigma_2^{-2}\sigma_3^{-1}\sigma_2\sigma_3^{-1}$	4	9
8_{14}	$\sigma_1^3\sigma_2\sigma_1^{-1}\sigma_2\sigma_3^{-1}\sigma_2\sigma_3^{-1}$	4	9
8_{15}	$\sigma_1^2\sigma_2^{-1}\sigma_1\sigma_3\sigma_2^3\sigma_3$	4	9
8_{16}	$\sigma_1^2\sigma_2^{-1}\sigma_1^2\sigma_2^{-1}\sigma_1\sigma_2^{-1}$	3	8
8_{17}	$\sigma_1^2\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-2}$	3	8
8_{18}	$\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}$	3	8
8_{19}	$\sigma_1^3\sigma_2\sigma_1^3\sigma_2$	3	8
8_{20}	$\sigma_1^3\sigma_2^{-1}\sigma_1^{-3}\sigma_2^{-1}$	3	8
8_{21}	$\sigma_1^3\sigma_2\sigma_1^{-2}\sigma_2^2$	3	8

Table 1. Braid words for knots with up to eight crossings [12]

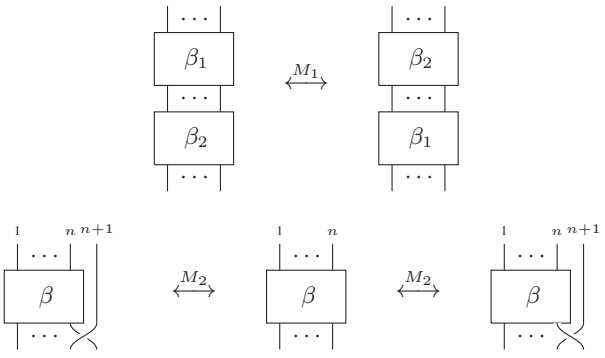


Figure 6. Markov moves of type M_1 (conjugation) and M_2 (stabilisation)

crossing-change operation is then simply a matter of taking such a word and then replacing a given σ_i with its inverse σ_i^{-1} or vice-versa.

Our goal is, given a knot K represented as the closure of a braid word $w \in B_n$, to evolve a sequence of certain moves which trivialises the knot with the smallest number of crossing-changes, thus obtaining an upper bound on the unknotting number $u(K)$ of K. The allowable moves are those which either leave the isotopy class of the corresponding knot unchanged, or which change the sign of a single crossing, and may be seen in Table 2.

$R_2^{\pm}$	$\sigma_i^{\pm 1}\sigma_i^{\mp 1}$	$\longmapsto$	1
$\overline{R}_2^{\pm}$	1	$\longmapsto$	$\sigma_i^{\pm 1}\sigma_i^{\mp 1}$
$R_3^{\pm}$	$\sigma_i^{\pm 1}\sigma_{i+1}^{\pm 1}\sigma_i^{\pm 1}$	$\longmapsto$	$\sigma_{i+1}^{\pm 1}\sigma_i^{\pm 1}\sigma_{i+1}^{\pm 1}$
$\overline{R}_3^{\pm}$	$\sigma_{i+1}^{\pm 1}\sigma_i^{\pm 1}\sigma_{i+1}^{\pm 1}$	$\longmapsto$	$\sigma_i^{\pm 1}\sigma_{i+1}^{\pm 1}\sigma_i^{\pm 1}$
$M_1^{\pm}$	$\sigma_i^{\pm 1}\alpha$	$\longmapsto$	$\alpha\sigma_i^{\pm 1}$
$\overline{M}_1^{\pm}$	$\alpha\sigma_i^{\pm 1}$	$\longmapsto$	$\sigma_i^{\pm 1}\alpha$
$M_2^{\pm}$	$\alpha\sigma_n^{\pm 1}$	$\longmapsto$	α
$\overline{M}_2^{\pm}$	α	$\longmapsto$	$\alpha\sigma_n^{\pm 1}$
$S^{\pm\pm}$	$\sigma_i^{\pm 1}\sigma_j^{\pm 1}$	$\longmapsto$	$\sigma_j^{\pm 1}\sigma_i^{\pm 1}$ (if $\lvert i-j\rvert > 1$)
$\overline{S}^{\pm\pm}$	$\sigma_j^{\pm 1}\sigma_i^{\pm 1}$	$\longmapsto$	$\sigma_i^{\pm 1}\sigma_j^{\pm 1}$ (if $\lvert i-j\rvert > 1$)
$U^+ = \overline{U}^-$	σ_i	$\longmapsto$	σ_i^{-1}
$U^- = \overline{U}^+$	σ_i^{-1}	$\longmapsto$	σ_i

Table 2. Allowable moves on braid words

In more detail, a move of type R_2^+ cancels a substring of the form $\sigma_i\sigma_i^{-1}$, a move of type R_2^- cancels a substring of the form $\sigma_i^{-1}\sigma_i$, and moves of type $\overline{R}^{\pm}$ introduce corresponding substrings. These are the Reidemeister moves of type 2, translated into the context of braid words. Similarly, moves of type R_3 perform Reidemeister moves of type 3, moves of type M_1 and M_2 are Markov moves, moves of type S represent non-interacting crossings sliding past each other, and moves of type U change the sign of a single crossing.

3 METHODS

In this paper, we use evolutionary techniques to find sequences of unknotting primitives which are optimal with respect to two subtly different but related problems:

Problem 1. *Given an arbitrary knot K, described as the closure $\widehat{\beta}$ of some braid word $\beta \in B_n$, is there a sequence of moves which reduces β to the trivial word $1 \in B_1$? If so, what is the minimal such sequence with respect to the number of crossing-change operations $U^{\pm}$, and in what cases does this yield a sharp upper bound for the unknotting number $u(K)$ of K when compared to known values such as those listed in the* KnotInfo *database* [12].

Problem 2. *Given a finite set S of knots, each described as the closure $\widehat{\beta}$ of some braid word $\beta \in B_n$, is there a single, universal sequence of moves which, possibly with repeated applications, trivialises each knot in S?*

The second of these problems is, broadly speaking, the generalisation of the first to more than one reference knot; equivalently, the first problem is the special case of the second where we consider a single knot.

As usual in an evolutionary computation approach, we need to define seven things: how population members are represented; the

parameters; how the population is initialised; how the mutation and crossover operators are defined; and how fitness evaluation and selection happen.

3.1 Representation

Each member of the population consists of a list of moves drawn from those in Table 2. This list can be of any length.

3.2 Parameters

For Problem 1 the population size was 500 and the number of iterations was $4 \times \text{length}(K)^2$, where K is the knot in question and length is the number of crossings. For Problem 2 the population size was 200 and the number of iterations was $4 \times \max(\text{length}(S))^2$, where S is the set of knots. These parameters were determined by informal experimentation. The program was run three times for each knot or set of knots, and the best result is reported.

3.3 Initialisation

Each member of the population is initialised by first choosing a number uniformly at random in the range [1,15] which is the length of the list, and then each slot in the list is filled in uniformly at random from the moves given in Table 2. The moves are selected with replacement; that is, a sequence may include more than one move of a given type.

3.4 Mutation

There are three different mutation operators, which are selected uniformly at random and applied to individuals with the overall mutation probability being 10%. These are:

1. Select a random move from the list and replace it with a move drawn randomly from the moves in Table 2.
2. Choose a random move from the list and delete it (as long as the list of moves contains at least one move).
3. Choose a random position in the list and insert a randomly chosen move from the moves in Table 2.

3.5 Crossover

One-point crossover is applied to all individuals as follows: the two strings are aligned, a position less than or equal to the length of the shortest string is selected at random, and the strings crossed over at that point.

3.6 Fitness evaluation

The members of the population are evaluated by attempting to unknot each of a set of knots, which in the case of Problem 1 will be just a single example, and in the case of Problem 2 will consist of more than one knot. The execution is carried out as follows. Let $M = m_1, m_2, \ldots, m_n$ be the n moves in the list. Let $K = K_0$ be the original knot, and $K_1, K_2, \ldots$ be the sequence of knots generated.

The knot K_0 is analysed for the preconditions for m_1 to be carried out, if they are satisfied then the move is applied, so that $K_1 := m_1(K_0)$; if the preconditions are not satisfied then the knot is unchanged ($K_1 := K_0$). The next step is to attempt to apply m_2 to K_1 by seeing if its preconditions are satisfied, and so on. When the end of M is reached, the list is begun again from the beginning.

Each time the knot is changed as a result of applying the move, the knot is checked to see if there are any crossings remaining. If so, the algorithm terminates, and a positive result returned.

For Problem 1 we apply the sequence M once only, but for Problem 2 we perform repeated applications of M. If the knot hasn't been trivialised by 50 applications of M, we assume that it has become stuck in a repeating loop (which, for the 1 701 936 knots with sixteen or fewer crossings [10], is a valid assumption), terminate the process with a negative result, and move on to the next knot (if any) on the list.

By inspection of Table 2 we see that certain operations (namely, those of type R_2 and M_2) reduce the length of the braid word under investigation, some (types R_3, M_1 and U) don't, and the rest (types $\overline{R}_2$ and $\overline{M}_2$) increase the length of the braid word.

It is known (see, for example, the paper by Coward [7]) that there exist diagrams for the unknot which can only be reduced to the standard (zero-crossing) diagram of the unknot by means of at least one move of type $\overline{R}_2$. That is, at some point during the reduction process, we have to temporarily increase the complexity. In this article however, for simplicity, we will restrict ourselves to sequences of moves which do not increase the length of the braid word. We intend to explore the more general case in later work.

With regard to Problem 1, we are trying to find a sequence of operations which unknot a specific knot with the smallest number of crossing changes. Slightly less importantly, we want to find the simplest possible such unknotting sequence.

For Problem 2, we are trying to find a sequence of operations which (perhaps with repeated applications) unknots as many knots as possible, as efficiently as possible.

Let S denote the set of reference braids, on which each sequence is being tested. (For Problem 1 this will consist of a single braid word.) Let $r_S(M)$ denote the number of braids in S which are fully reduced by (one or more application of) the sequence M. (In practice, we specify an upper threshold of 50 iterations, as described above.) Let $\min_S(M)$ and $\max_S(M)$ denote, respectively, the minimum and maximum number, over all braids in S, of iterations of M required to reduce a (reducible) braid. Let $l(M)$ denote the length of the sequence M. By $c(M)$ we denote the number of crossing-change (type U) operations in M, and by $c_S(M)$ we denote the total number, over all braids in S, of successful crossing-change (type U) operations. That is, $c_S(M)$ gives a measure of the total amount of unknotting actually performed by the sequence M.

In the case of Problem 1, some of the operations in the string M may have no effect on the braid under examination. For example, applying an R_2 move to a braid which at that stage has no $\sigma_i^{\pm 1}\sigma_i^{\mp 1}$ substrings will leave the braid unchanged, and may thus be safely elided from the sequence, resulting in a shorter sequence. Given a sequence M, applied to a braid $\beta \in B_n$, we denote by $l_{\text{opt}}(M)$ the length of the sequence obtained by optimising M in this way with respect to β.

The fitness function should, ideally, seek to minimise the number of crossing changes, maximise (at least when working on Problem 2) the number of knots which can be reduced by a given sequence, minimise the length of the sequence, and minimise the number of repeated applications of the sequence required to reduce those braids in S which are reducible by the operations under consideration.

With those criteria in mind, we define the fitness function for Problem 1 to be

$$f_1(M) = 1 + \frac{10000 r_S(M)}{l_{\text{opt}}(M) + c_S(M)^3 + 1}$$

and that for Problem 2 to be

$$f_2(M) = 1 + \frac{r_S(M)^2}{1 + \max_S(M) + l(M)}.$$

Since both $f_1(M)$ and $f_2(M)$ depend only on the set of braids under consideration, which doesn't change between generations, we can optimise the simulations by caching the fitness values for a given string M, rather than recalculating it each time.

3.7 Selection

Using an approach similar to the Stochastic Universal Sampling Algorithm [4], in each generation, we rank the candidate sequences in order of their normalised fitness $\bar{f}(M)$: the fitness $f(M)$ of the sequence M divided by the mean fitness over the whole population. The integer part of $\bar{f}(M)$ gives the number of copies contributed to the next generation, while the fractional part gives the probability of an additional copy. So, a sequence M with a normalised fitness $\bar{f}(M) = 1.72$ contributes one copy to the following generation, plus a 72% chance of a second copy.

3.8 Implementation

The source code for the implementation is available from the authors on request.

4 RESULTS

4.1 The single unknotting problem

Table 1 lists braid words for all knots with up to eight crossings, and Table 3 lists unknotting sequences for those knots, generated by a Perl program implementing Problem 1. The sequences are not necessarily unique (and in many cases will not be), nor are they guaranteed to be optimal; however we observe that for 21 of the 35 knots with eight or fewer crossings, our program has correctly calculated the unknotting number.

Figure 7 shows the braid $\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}$ (whose closure is isotopic to the figure-eight knot 4_1) being reduced by the sequence $UM_1^2R_3R_2M_2^2$.

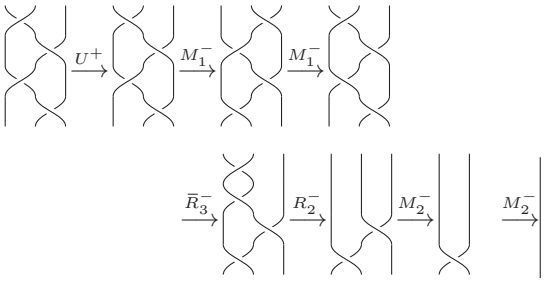


Figure 7. Reduction of the figure-eight knot

The knots 3_1, 5_1 and 7_1 are worth examining a little closer: these are the *torus knots* of type $(2, 2n+1)$, with braid presentation $\sigma_1^{2n+1} \in B_2$ and unknotting number $u(K) = n$. The unknotting sequences obtained for these knots have a very similar form, namely $(UR_2)^{n-1}M_2$. More generally, given two positive, coprime integers p and q, the *torus knot* of type (p, q) is the knot which can be drawn on the surface of a standard, unknotted torus so that the strands wind p times round the torus in the longitudinal direction, and q times in the meridional direction. The torus knot $T_{p,q}$ of type (p, q) has unknotting number $u(T_{p,q}) = \frac{1}{2}(p-1)(q-1)$. See Rolfsen [17, Section 3.C] or Cromwell [8, Section 1.5] for further details on torus knots.

K	Unknotting sequence M	$c(M)$	$u(K)$
3_1	UR_2M_2	1	1
4_1	$UM_1^2R_3R_2M_2^2$	1	1
5_1	$(UR_2)^2M_2$	2	2
5_2	$UR_2M_1^2R_3R_2M_2^2$	1	1
6_1	$UR_2UM_2M_1^2R_3R_2M_2^2$	2	1
6_2	$UR_2UM_1^2R_3R_2M_2^2$	2	1
6_3	$UR_2UM_2R_2M_2$	2	1
7_1	$(UR_2)^3M_2$	3	3
7_2	$UR_2M_1(M_1U)^2(M_1R_3R_2M_2)^2M_2$	3	1
7_3	$(UR_2M_1)^2R_3R_2M_2^2$	2	2
7_4	$UR_2M_2UR_2M_1^2R_3R_2M_2^2$	2	2
7_5	$(UR_2)^2M_2UR_2M_2$	3	2
7_6	$UR_2M_2UM_1^2R_3R_2M_2^2$	2	1
7_7	$M_1UM_1^4(R_3R_2M_2)^2M_2$	1	1
8_1	$(UR_2M_2)^2UM_1^2R_3R_2M_2^2$	3	1
8_2	$UR_2M_1^4UM_1^2R_3M_1R_2^2M_2^2$	2	2
8_3	$UR_2M_2M_1^2UM_1^2R_3R_2M_1M_2R_3R_2M_2^2$	2	2
8_4	$M_1UR_2M_1UM_1^3(R_3R_2M_2)^2M_2$	2	2
8_5	$UM_1UR_2M_1^2R_3R_2^2M_2^2$	2	2
8_6	$(UR_2)^2M_2UM_1^2R_3R_2M_2^2$	3	2
8_7	$M_1UR_2M_1UM_1^2R_3^2(R_2M_2)^2$	2	1
8_8	$UR_2M_1^5UR_2M_1^2R_3M_1R_2M_2^3$	2	2
8_9	$M_1^3UM_1^2R_3^3M_2R_3^2M_2$	1	1
8_{10}	$UR_2UM_1^3(R_3R_2)^2M_2^2$	2	2
8_{11}	$UR_2M_2M_1^2UM_1^2R_3M_1R_2^2M_2^2$	2	1
8_{12}	$M_1UM_1^5UM_1R_3M_1UM_2R_2M_2R_3R_2M_2^2$	3	2
8_{13}	$UR_2M_2UR_2M_1^2R_3R_2M_2^2$	2	1
8_{14}	$UR_2M_1UM_1^2UM_1^2(R_3R_2M_2)^2M_2$	3	1
8_{15}	$UR_2M_2M_1^3UR_2M_1R_3R_2M_2^2$	2	2
8_{16}	$UR_2UM_1U(R_2M_2)^2$	3	2
8_{17}	$M_1^2UM_1^3R_3M_1R_2(R_3R_2)^2M_2^2$	1	1
8_{18}	$UM_1^2R_3R_2M_1U(R_2M_2)^2$	2	2
8_{19}	$UR_2M_1R_3UM_1U(R_2M_2)^2$	3	3
8_{20}	$M_1^5UR_2M_1^2R_3R_2^2M_2^2$	1	1
8_{21}	$UR_2M_1^3(R_3R_2)^2M_2^2$	1	1

Table 3. Unknotting sequences M for single knots K, using the braid words from Table 1, comparing the number $c(M)$ of crossing changes performed by the sequence M, and the crossing number $u(K)$ of the knot K

4.2 The multiple unknotting problem

A simulation of Problem 2, again implemented in Perl, obtains universal or near-universal unknotting sequences, some examples of which are listed in Table 4. Some of the more complex braids (those corresponding to knots with minimal crossing number 9 or higher) were unreducible by any scheme found by our program, because we restricted ourselves to operations which don't increase the length of the braid word. As noted earlier (see the paper by Coward [7] for details), sometimes we need to perform a move of type $\bar{R}_2^{\pm}$ to introduce two additional crossings during the reduction scheme.

S	M	$\max_S(M)$	$r_S(M)$	$\lvert S\rvert$
3_1–4_1	$UM_1^2R_3R_2M_2^2$	1	2	2
3_1–4_1	$UM_1^2R_3R_2M_2$	2	2	2
3_1–4_1	$UR_2R_3M_2M_1^2$	3	2	2
3_1–5_2	$M_1UR_3R_2M_1M_2^2U$	2	4	4
3_1–5_2	$UR_2M_1^2R_3M_2R_2M_2$	2	4	4
3_1–5_2	$UR_3R_2M_1^3R_3M_2^2$	2	4	4
3_1–6_3	$M_1R_2UM_1^2R_3R_2M_2$	3	7	7
3_1–6_3	$M_1UM_1^2R_3R_2M_2$	3	7	7
3_1–7_7	$M_2UR_2M_2M_1UR_3M_1^2M_2$	4	14	14
3_1–7_7	$M_1R_3USM_1R_2M_2M_1$	5	14	14
3_1–7_7	$UR_3R_2SM_2M_1^2M_2$	6	14	14
3_1–8_{21}	$M_1UR_3M_1^2R_3M_2SR_3R_2M_2^3$	5	35	35
3_1–8_{21}	$M_1UM_1R_3R_2M_2^2R_3M_1$	7	35	35
3_1–9_{49}	$UR_2R_3M_1^2UM_2R_3M_1M_2M_1R_3$	6	74	84
3_1–9_{49}	$R_3R_2UM_1^2M_2^2$	10	73	84

Table 4. Universal or near-universal unknotting sequences for multiple knots

5 CONCLUSIONS AND FUTURE WORK

In this paper we have seen how an unknotting algorithm can be evolved based on a number of primitive moves. There are a number of future directions for research in the area of applying evolutionary algorithms and other machine learning techniques to mathematical problems in knot theory and related areas.

Initially, there are some basic extensions to the work described in this paper. For example, rather than focusing on unknotting we could use similar techniques to address the related problem of knot equivalence. Furthermore, there are similar problems in other areas of mathematics (for example, graph theory and group theory) so this could be extended to those.

More interestingly, there is the question of counterexample search. There are a number of conjectures in this area where there are some measures that could be used to ascertain how close a particular example is to being a counterexample to the conjecture. Using these measures, experiments could be done on exploring the space of knots to find a counterexample; some preliminary work along these lines has been done by Mahrwald [13].

Another approach would be to take a data mining approach to certain mathematical problems. For example, we could generate a large database of knots and then apply classification techniques to distinguish different classes of knot, or applying clustering techniques to group knots according to some metric.

An examination of the results from this classification might give new insights into the underlying structure of the space of knots. A related topic is using genetic programming to evolve *invariants*, that is, functions that distinguish between different knots by processing their diagrams.

REFERENCES

[1] J. W. Alexander, 'A lemma on systems of knotted curves', *Proc. Nat. Acad. Sci. USA*, **9**, 93–95, (1923).

[2] J. W. Alexander and G. B. Briggs, 'On types of knotted curves', *Ann. of Math. (2)*, **28**(1-4), 562–586, (1926/27).

[3] Emil Artin, 'Theorie der Zöpfe', *Abh. Math. Sem. Univ. Hamburg*, **4**, 47–72, (1925).

[4] James E. Baker, 'Reducing bias and inefficiency in the selection algorithm', in *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, ed., John J. Grefenstette, pp. 14–21, Hillsdale, New Jersey, (1987). L. Erlbaum Associates.

[5] Joan S. Birman, *Braids, links, and mapping class groups*, volume 82 of *Annals of Mathematics Studies*, Princeton University Press, Princeton, N.J., 1974.

[6] Maciej Borodzik and Stefan Friedl. The unknotting number and classical invariants I. arXiv preprint 1203:3225, 2012.

[7] Alexander Coward, 'Ordering the Reidemeister moves of a classical knot', *Algebr. Geom. Topol.*, **6**, 659–671, (2006).

[8] Peter R. Cromwell, *Knots and links*, Cambridge University Press, Cambridge, 2004.

[9] Vagn Lundsgaard Hansen, *Braids and coverings: selected topics*, volume 18 of *London Mathematical Society Student Texts*, Cambridge University Press, Cambridge, 1989. With appendices by Lars Gæde and Hugh R. Morton.

[10] Jim Hoste, Morwen Thistlethwaite, and Jeff Weeks, 'The first 1,701,936 knots', *Math. Intelligencer*, **20**(4), 33–48, (1998).

[11] Sofia Lambropoulou and Colin P. Rourke, 'Markov's theorem in 3–manifolds', *Topology Appl.*, **78**(1-2), 95–122, (1997). Special issue on braid groups and related topics (Jerusalem, 1995).

[12] Charles Livingston and Jae Choon Cha. KnotInfo. `http://www.indiana.edu/~knotinfo/`.

[13] Valentin Mahrwald, *Search-based Approaches to Knot Equivalence*, MMath dissertation, Department of Computer Science, University of York, May 2008.

[14] A. A. Markov, 'Über die freie Aquivalenz geschlossener Zöpfe', *Rec. Math. Mosc.*, **1**, 73–78, (1935).

[15] Peter Ozsváth and Zoltán Szabó. Knots with unknotting number one and Heegaard Floer homology. arXiv preprint math/0401426, 2004.

[16] Kurt Reidemeister, 'Elementare begründung der knotentheorie', *Abh. Math. Sem. Univ. Hamburg*, **5**, 24–32, (1926).

[17] Dale Rolfsen, *Knots and links*, volume 7 of *Mathematics Lecture Series*, Publish or Perish Inc., Houston, TX, 1990. Corrected reprint of the 1976 original.

[18] De Witt Sumners, 'Lifting the curtain: using topology to probe the hidden action of enzymes', *Notices Amer. Math. Soc.*, **42**(5), 528–537, (1995).

[19] Pierre Vogel, 'Representation of links by braids: a new algorithm', *Comment. Math. Helv.*, **65**(1), 104–113, (1990).

[20] Edward Witten, 'Quantum field theory and the Jones polynomial', *Comm. Math. Phys.*, **121**(3), 351–399, (1989).

A Scalable Genome Representation for Neural-Symbolic Networks

Joe Townsend, Antony Galton and Ed Keedwell
College of Engineering, Mathematics and Physical Sciences, Harrison Building, University of Exeter, North Park Road. Exeter, EX4 4QF
jt231@ex.ac.uk, A.P.Galton@ex.ac.uk , E.C.Keedwell@ex.ac.uk

Abstract. Neural networks that are capable of representing symbolic information such as logic programs are said to be neural-symbolic. Because the human mind is composed of interconnected neurons and is capable of storing and processing symbolic information, neural-symbolic networks contribute towards a model of human cognition. Given that natural evolution and development are capable of producing biological networks that are able to process logic, it may be possible to produce their artificial counterparts through evolutionary algorithms that have developmental properties. The first step towards this goal is to design a genome representation of a neural-symbolic network. This paper presents a genome that directs the growth of neural-symbolic networks constructed according to a model known as SHRUTI. The genome is successful in producing SHRUTI networks that learn to represent relations between logical predicates based on observations of sequences of predicate instances. A practical advantage of the genome is that its length is independent of the size of the network it encodes, because rather than explicitly encoding a network topology, it encodes a set of developmental rules. This approach to encoding structure in a genome also has biological grounding.

1 INTRODUCTION

Neural-Symbolic Integration [1, 9] is a field in which symbolic and sub-symbolic approaches to artificial intelligence are united by representing logic programs as neural networks or by developing methods of extracting knowledge from trained networks. The motivation behind this work is either the construction of effective reasoning systems, the understanding of knowledge encoded in neural networks, a model of human cognition, or a combination of these.

It may be possible to find powerful neural-symbolic networks through an evolutionary search. However, as the size of a logic program increases, so does the size of the network used to represent it. An evolutionary search for larger networks would take longer than it would for smaller networks as the search space would be larger, unless networks can be represented in a scalable way. *Artificial development* is a sub-field of evolutionary computing in which genomes encode rules for the gradual development of phenotypes rather than encoding their structures explicitly [3]. The genomes are scalable because genomes of equal length can produce solutions of different sizes. Among other applications, this encoding method can be applied to the representation of neural networks. This method of encoding networks is referred to as *indirect encoding*.

In addition to producing powerful reasoning systems, representing neural-symbolic networks in this way is more biologically plausible than encoding topologies directly. Because neural-symbolic networks claim to be a step towards a model of human cognition, it seems reasonable to develop them in a way which is also biologically plausible. If human cognition can be produced through evolution and development, then perhaps an artificial model of cognition can be produced through artificial models of evolution and development.

No attempt has yet been made to evolve neural-symbolic networks using artificial development. This paper introduces a scalable genome representation of neural-symbolic networks which adhere to a model known as SHRUTI [22, 23]. The genome was successful in its ability to construct four SHRUTI networks that were able to learn a set of relations between logical predicates. The intention is to eventually produce these genomes using an evolutionary algorithm, but this algorithm has yet to be implemented. Nonetheless, if SHRUTI networks can be produced through artificial development, it opens the possibility that other neural-symbolic models can be too. Section 2 provides an overview of SHRUTI and artificial development models used for the evolution of standard neural networks. Section 3 describes the experiments performed, the genome model used in these experiments and the target networks. Section 4 presents and discusses the results and section 5 concludes.

2 BACKGROUND

2.1 SHRUTI

SHRUTI is a neural-symbolic model in which predicates are represented as clusters of neurons and the relations between them as connections between those neurons [22, 23]. Predicate arguments are bound to entities filling the roles of those arguments by the synchronous firing of the neurons representing them. There is therefore no need to create a connection for every argument-entity combination. The SHRUTI authors claim that their approach, known as *temporal synchrony*, has biological grounding in that it is used for signal processing in biological neurons. SHRUTI can be used for forward or backward reasoning. In forward reasoning, the system is used to predict all the consequences of the facts. In backward reasoning, the system is used to confirm or deny the truth of a predicate instance based on the facts encoded. In other words, a backward reasoner answers 'true or false' questions. This paper concerns backward reasoning only.

Figure 1 provides an example of a basic SHRUTI network. Argument and entity neurons fire signals in phases, and a predicate is instantiated by firing its argument neurons in phase

with the entities fulfilling roles in the predicate instance. The other neurons in a predicate cluster fire signals of continuous phase only upon receipt of signals of the same nature. Positive and negative collectors (labelled '+' and '-') fire when the current predicate instance is found to be true or false respectively, and enablers (labelled '?') fire when the truth value of the current predicate instance is queried. Relations between predicates are established by linking corresponding neurons such that argument bindings are propagated between predicates. Facts are represented by static bindings between entities and predicate arguments such that the firing of a corresponding *fact neuron* (represented as a triangle in figure 1) is inhibited if the current dynamic bindings of the predicate do not match the static bindings of the fact.

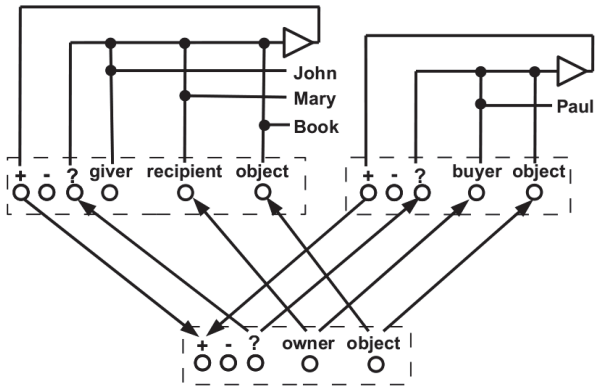


Figure 1 – A simple SHRUTI network for the relations Give(x,y,z) → Own(y,z) and Buy(x,y) → Own(x,y) and the facts Give(John, Mary, Book) and Buy(Paul,y)

The example network in figure 1 represents the relations *Give(x,y,z) → Own(y,z)* (if person x gives z to person y, then person y owns z) and *Buy(x,y) → Own(x,y)* (if person x buys y, then person x owns y). Also, two facts are represented: *give(John,Mary,Book*) (John gave Mary the book) and *buy(Paul,x)* (Paul bought something). This network is configured for backward reasoning. If one wishes to find the truth for *own(Mary, Book)* (Does Mary own the Book?), an instance of the *own* predicate must first be created by firing its *owner* and *object* neurons in the same phases as the neurons representing *Mary* and *Book* respectively. This creates a pair of dynamic bindings. The enabler (?) of *own* must also be fired to indicate that a search of *own*'s current instance is sought. The dynamic bindings are propagated along the connections to *give* and *buy* such that the neurons representing *recipient* and *buyer* are now firing in phase with *Mary* and the neurons representing *object* for *give* and *buy* are firing in phase with *Book*. *Give* and *buy* are therefore instantiated with the queries *give(x, Mary, Book)* (did somebody give Mary the book?) and *buy(Mary, Book)* (did Mary buy the book?). The dynamic bindings are then propagated to the static bindings representing facts. The static bindings of *buy(Paul,x)* do not match the dynamic bindings of *buy(Mary, Book)*, and so the static bindings inhibit the firing of the corresponding fact node which would otherwise activate the positive collector of *buy*. However, the dynamic bindings of *give(x, Mary, Book)* do match the static bindings of *give(John,Mary,Book*). The corresponding fact node is therefore not inhibited and activates the positive collector of *give* to assert that *give(x, Mary, Book)* is true. This collector in turn activates the positive collector of *own* to assert that *own(Mary, Book)* is also true, i.e. that Mary does indeed own the book.

There are many more features which may be included in a SHRUTI network. The literature also presents means of restricting dynamic bindings by entity types, conjoining predicates, enabling multiple instantiations of a predicate, and many other features. More complex models even use multiple neurons to represent one argument or entity, as the use of only one neuron to represent a concept lacks biological plausibility. One particular feature worth discussing in further detail is SHRUTI's learning mechanism [26] since it plays an important role in the developmental process discussed later in this paper.

SHRUTI's learning mechanism takes inspiration from Hebbian learning [10]. The training data is a sequence of events (predicate instances) observed over time that reflect the causal relations between the predicates. When two predicates are observed within a fixed time window, any connections representing relations between them are strengthened to increase the likelihood that the predicate observed first is a cause of the second. After a predicate is observed, any predicates that are connected to it but are not observed within the time window have those connections weakened to reflect the likelihood that they are not consequents of the first predicate. When a weight ω is strengthened, it is updated according to equation 1. When ω is weakened, it is updated according to equation 2. In both cases, the learning rate α is defined according to equation 3. This ensures that it becomes more difficult to change a relation for which evidence has been observed a large number of times.

$$(1) \quad \omega_{t+1} = \omega_t + \alpha * (1 - \omega_t)$$

$$(2) \quad \omega_{t+1} = \omega_t - \alpha * \omega_t$$

$$(3) \quad \alpha = \frac{1}{Number\ of\ Updates}$$

For example, if B(a,b) is observed shortly after A(a,b), connections will be updated to reflect A(x,y) → B(x,y). However, if A(a,b) is observed with no immediate observation of B(a,b), the same connection weights are weakened to reflect the lack of a relation between the two predicates. A new predicate can be recruited into the network once the connection weights of its neurons have gained sufficient strength.

The SHRUTI developers argue that some level of pre-organisation would be necessary for this learning model to work, and that this pre-organisation could be the product of evolutionary and developmental processes. To support the biological plausibility of pre-organisation, they point to the work of Marcus [16], who proposed ideas similar to those found in artificial development. However, a further review of literature has failed to find any attempts to produce SHRUTI networks using artificial development or similar methods. This is what motivates the ideas proposed in this paper.

2.2 Artificial Development of Neural Networks

Artificial development is a form of evolutionary computing in which the genome encodes instructions for the gradual development of the phenotype. This method is argued to be more biologically plausible than the alternative of encoding the structure of a phenotype explicitly in the genome, as it is closer

to the means by which DNA encodes biological structures. Dawkins argues that DNA is not a blueprint of biological structure but is more like a recipe for its construction [5]. Artificial development can be applied to a range of problems, and Chavoya provides a recent overview of artificial development models [3]. However, this paper is only concerned with the artificial development of neural networks.

One approach to evolving neural networks involves genomes which encode network topologies [24, 25]. For example, the genome may contain a list of neurons and another list of connections between them, or it may represent a connection matrix. Such methods of encoding are often referred to as *direct encoding*. The disadvantage of direct encoding is that the size of the genome increases in proportion to that of the network it represents. The alternative, *indirect encoding*, overcomes this problem by encoding a set of rules for the gradual development of the network. Just as a biological organism's cells all contain the same DNA, neurons within networks encoded by indirect encoding all contain or refer to a copy of the same genome, which represents a set of developmental rules. These rules provide instructions as to how the neuron should develop, for example by duplicating or deleting itself or by establishing a connection to another neuron. The developmental process often begins with only one neuron. When a neuron divides, its genome is passed on to both of its children, which is how all neurons are able to share the same genome. Which developmental operation takes place depends on the current attributes of the neuron. Therefore even though all neurons share the same genome, which developmental operations are executed at which point in time will not necessarily be the same for each neuron. Some models for the artificial development of neural networks use graph grammars inspired by Lindenmayer systems [15], whereas others use more biologically inspired ideas where neurons and connections are defined in a two or three dimensional Euclidean grid space.

Figure 2 presents grammar trees used by Gruau to define cell division processes [8]. Each node in the tree describes a cell (neuron) division. The children of each node describe the following division for each child neuron produced by the previous division. Separate grammar trees define sub-trees, the roots of which can be referenced by leaf nodes of the main tree. A sub-structure can therefore be encoded once but reused multiple times. As a consequence, genomes are more compact and convergence speed during evolution is reduced because the search space is smaller. Kitano used grammar encoding to develop connection matrices [12, 13]. This method could also produce repeated sub-structures, evident from repeated patterns in the connection matrices produced.

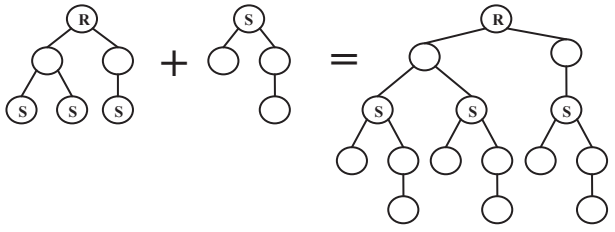


Figure 2 – Gruau's grammar trees. Each node corresponds to a cell division. The left-most tree describes the initial divisions from the root, and the second tree describes sub-trees which may grow from the leaves of the first tree. The third tree depicts the overall cell division process.

In the more biologically inspired methods, neurons and their connections (often regarded as *axons*, as with actual biological neurons) have positional attributes. A neuron's axon grows in the grid space, guided by developmental rules encoded in the genome, and form a connection when they come into contact with another neuron. The positional information of a neuron and its axons can be used to influence development. Eggenberger employed this idea using gene regulation [6, 7]. The activation of one gene, in addition to producing or deleting cells or connections, may also activate or inhibit the activation of other genes in the genome. Information can be passed between cells and the between the genes in those cells using artificial molecules. Concentration gradients of these molecules provide the positional information required to direct growth. Kitano also developed a similar model [14]. A different approach has been to use Cartesian genetic programs [18] to influence development in a grid [11]. The genome represents a set of seven interconnected programs. Three of these programs control signal processing in neurons, three control life-cycle processes such as the addition and deletion of neurons, and another controls weight updates.

Nolfi and Parisi used a means of measuring the fitness of a developing network that may prove useful in further research [19, 20]. Rather than simply measuring fitness at the end of the life-cycle of each phenotype, fitness was measured at different stages throughout development in order to observe how fitness increased over time. Such information on how the phenotype develops may be useful in the calculation of an overall fitness. For example, one might wish to measure overall fitness as the area under the fitness-time graph.

3 METHOD

A scalable genome model for the development of SHRUTI networks was produced, and the aim of the experiments conducted was to demonstrate, using an instance of this genome model, that the model could be used to develop four networks that could learn a set of logical relations between predicates based on a series of observed events. Each event was a predicate instance and each event sequence was representative of the relationships between the predicates in each logic program. Experiments were later repeated with shuffled event sequences in order to observe whether or not the genome could still develop networks which could represent the same logic programs. In additional experiments, sections of the genome were removed in order to see how the structures of the developed networks were affected. This section outlines how a SHRUTI model was implemented for these experiments, the genome model used to represent this implementation, and the target networks that were developed by the genome.

3.1 SHRUTI implementation

It seems reasonable to attempt the artificial development of a simple SHRUTI network before the development of more complex features is attempted. Therefore the basic SHRUTI model capable of learning as described in section 2.1 was implemented but more complicated features such as type restrictions and conjunction were excluded.

A minor adjustment was made to the learning mechanism in order to overcome difficulties learning certain structures. If two relations exist which share the same antecedent but with different signs for that antecedent, the system struggles to learn

both relations because the strengthening of one weakens the other unless both relations have been observed a sufficient number of times. To explain why this occurs, the learning of the relations P(x,y) → Q(x,y) and ¬P(x,y) → R(x,y) will be used as an example. The collectors of Q and R receive input from different collectors of P (+P and -P respectively). However, the enablers of Q and R both provide input to the same (and only) enabler of P (?P). Observation of P(x,y) and Q(x,y) within the time window will strengthen the connection from +P to +Q and the connection from ?Q to ?P. However, since ?P is activated and ?R is not, the connection from ?R to ?P will weaken. Likewise, if -P(x,y) and R(x,y) are observed within the time window, The connection from ?R to ?P will strengthen but the connection from ?Q to ?P will weaken.

To overcome this problem, the learning mechanism was configured by adjusting the learning rate α to update by a greater magnitude when strengthening weights than when weakening them. Therefore when weakening weights, α is defined as in equation 3, but when strengthening weights it is increased as shown in equation 4:

$$\alpha = \frac{1.25}{Number\ of\ Updates} \quad (4)$$

This makes it possible to learn these conflicting pairs of relations as long as the events that reflect them occur a sufficient number of times.

3.2 The Genome

In this first genome model, only the connections between neurons are developed, and not the neurons themselves. This approach assumes the pre-existence of neuron clusters representing facts and predicates, but there is room in future work to attempt the development of these clusters also.

Each genome describes a tree structure in which leaf nodes represent actions to be performed and all other nodes represent conditions. Each path through the tree structure from the root node to a leaf node represents a different rule. After each event has been observed and weights have been updated accordingly, the conditions encoded in a genome are tested for each neuron and each of its existing and possible inputs. If a leaf node is reached, the action it encodes is executed. The genome labels the current neuron for which input connections are being made as SELF. The neuron from which a connection is being considered is labelled as P_INPUT (possible input) if it does not yet exist and E_INPUT (existing input) if it does exist.

Figure 3 shows a set of conditions encoded by a genome for the development of a SHRUTI network and figure 4 shows them as a decision tree. Figure 5 presents an example of how an input connection is created using rule 2. To reduce execution time, conditions which affect SELF are considered first, so that evaluation of existing or potential inputs is only performed when necessary. Branching from one condition to another is therefore limited such that SELF conditions can branch to P_INPUT and E_INPUT conditions, but P_INPUT and E_INPUT conditions cannot branch to SELF conditions. The genome begins with a header containing the genome index of each type of condition and of the actions.

For each condition, the genome encodes the attribute to be tested, an operator (<,≤,=,≥,>,≠), and the value to test that attribute against. Attributes which can be tested in this model are the neuron's current level of activity, its type (role, enabler or collector), the total number of inputs, and for existing inputs, the weight and the number of updates (how many times a connection has been strengthened or weakened). The genome also specifies the next condition to test or action to perform in the event of the current condition being evaluated as true or false. Alternatively, the tree search can end when a condition is evaluated as false and no actions are performed. For each action, the genome specifies one of two types of action to be performed: the addition or deletion of a connection. If a new connection is to be created, the genome also specifies the weight of the new connection.

SELF conditions:
1. If activity > 0.5, go to 2, else go to 5
2. If type = role node, go to 8, else go to 3
3. If type = enabler, go to 9, else go to 4
4. If type = collector, go to 10, else end.

E_INPUT conditions:
5. If number of updates > 7, go to 6, else end
6. If weight < 0.5, go to 12, else end.

P_INPUT conditions:
7. If activity > 0.5, go to 11, else end.
8. If type = role node, go to 7, else end.
9. If type = enabler, go to 7, else end.
10. If type = collector, go to 7, else end.

Actions:
11. Add connection with weight 0.1
12. Delete connection

Figure 3 – Conditions represented in the genome.

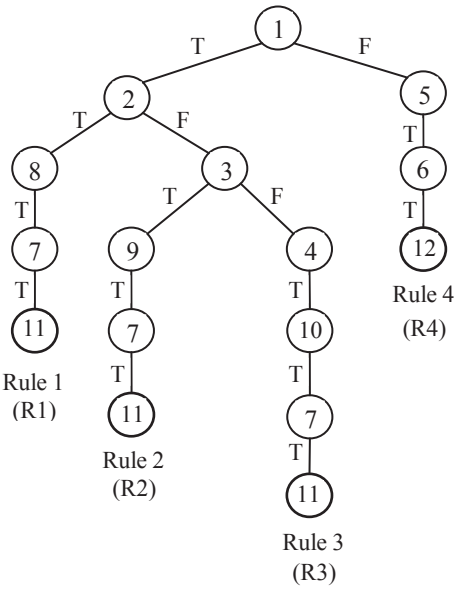


Figure 4 – A decision tree representation of the conditions given in figure 3 (T = True, F = False).

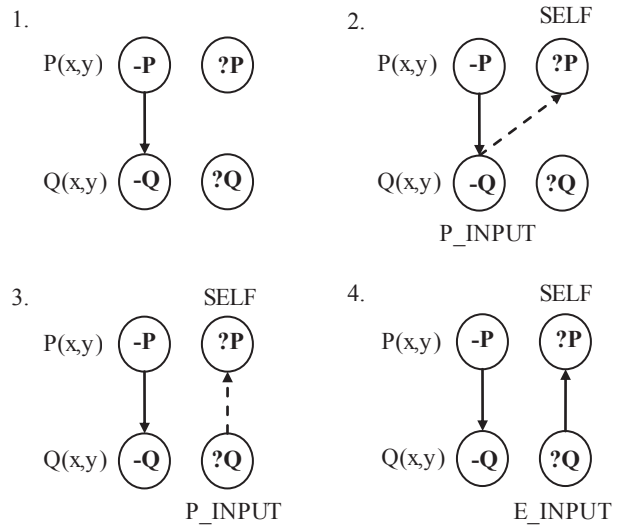


Figure 5 – How rule 2 is used to develop connections between enablers for the rule P(x,y) → Q(x,y). All nodes are active and a connection from -P to -Q already exists (1). The genome in ?P searches for input connections, starting with -Q (2). Both neurons are active and ?P is an enabler, but -Q is not, so no connection is made. The test is repeated for ?Q (3). Both neurons are enablers and both nodes are active, so an input connection is made (4).

In the genome defined in figures 3 and 4, rule 1 (R1) establishes connections between active role nodes. Rule 2 (R2) does the same for enablers and rule 3 (R3) does the same for collectors. Unwanted connections between neurons will inevitably form, but after Hebbian learning has taken place, their weights will weaken. Rule 4 (R4) prunes connections that are weak despite a large number of updates. A threshold of seven updates was chosen because this was the minimum value required to enable all test networks to learn desired connection weights without those connections being removed too early. This genome is one of a number which may be defined using this model to produce working SHRUTI networks.

3.3 Target networks

Figure 6 shows four target networks to be developed using the genome in figure 3. One of the networks is smaller than others in order to demonstrate the scalability of the genome. The two larger networks are of similar sizes but differ in structure. The logic program represented by the network with the label *SubNets* contains a relation and a predicate that are disjoint from the other relations and from each other. They are therefore each represented by separate sub-networks.

The initial state of each network is a set of neurons encoding facts connected to predicates, with the intention that connections will develop between predicate neurons over time in order to represent the relations between the predicates. For each network, a sequence of events in the form of predicate instances is defined. Each sequence reflects the relations between the predicates in the corresponding logic program. Different sub-sequences provide evidence for different sets of transitive relations. For example, observing the sub-sequence P(a,b), Q(a,b), R(a,b) supports the transitive pair of relations P(x,y)→Q(x,y), Q(x,y)→R(x,y).

Small

Expected Relations	Facts
P(x,y) → Q(x,y)	P(a,b)
¬P(x,y) → R(x,y)	¬P(c,d)

Large1

Expected Relations	Facts
P(x,y) → Q(x,y)	P(a,b)
Q(x,y) → ¬R(x,y)	¬Q(c,d)
¬Q(x,y) → S(x,y)	Q(e,f)
¬R(x,y) → ¬T(x,y)	S(g,h)
¬R(x,y) → ¬U(x,y)	
S(x,y) → V(x,y)	

Large2

Expected Relations	Facts
P(x,y) → ¬Q(x,y)	P(a,b)
¬P(x,y) → R(x,y)	¬Q(c,d)
¬Q(x,y) → ¬S(x,y)	¬P(e,f)
R(x,y) → T(x,y)	R(g,h)
¬R(x,y) → ¬U(x,y)	¬R(i,j)

SubNets

Expected Relations	Facts
P(x,y) → Q(x,y)	P(a,b)
R(x,y) → S(x,y)	R(c,d)
¬R(x,y) → T(x,y)	¬R(e,f)
¬T(x,y) → ¬U(x,y)	¬T(g,h)
	V(a,b)

Figure 6 – Target networks. Each table shows the relations which were expected to develop and the hard coded facts which make up the background knowledge. In the logic program represented by SubNets, the relation P(x,y) → Q(x,y) and the predicate V(a,b) are disjoint from the other relations and from each other.

Any number of events may occur at each time *t*, even zero. At each *t*, neurons that represent an observed predicate are fired and Hebbian learning is used to update the weights of the connections between the neurons that fire within a fixed time window of each other in order to build relations between predicates. Developed networks were tested by inputting 'true or false' questions and fitness was based on the number of questions answered correctly. The reader is reminded that each predicate includes two collectors: one positive and one negative, to assert the truth and falsity of the predicate respectively. Activation of one of these collectors shall be denoted 1, and deactivation 0. The truth of a predicate instance is therefore denoted by (1, 0), falsity by (0, 1), uncertainty by (0, 0) and contradiction by (1, 1). Fitness is measured as the number of correct collector activations. For example, consider a question with expected answer (1, 0). Answering (1, 0) would add 2 to the fitness, answering (0, 0) or (1, 1) would add 1, and (0, 1) would add 0.

For each network, the following statistics were recorded: the number of connection additions and deletions, the final number of connections and the final number of live connections. A connection is live when its weight is above the threshold of the neuron for which it is an input. Connections that are not live

make no contribution to inference in the network. However, this does not necessarily mean that all live connections do.

4 RESULTS

All test networks developed such that they could answer all of their test questions correctly. The same genome was successfully applied to the development of large and small networks, demonstrating that the genome is scalable in that its size is independent of the size of the phenotype. Further experiments observed how different components of the genome affected the network structure and what affected the change in fitness over time.

4.1 Network structure

Table 1 shows the statistics for each network. In each case, the total number of connections developed was not much greater than the number of live connections, meaning that only a few superfluous connections were developed.

Network	Connections	Live	Additions	Deletions
Small	10	10	30	20
Large1	49	36	138	89
Large2	43	35	86	43
SubNets	45	29	74	29

Table 1 – Statistics of fully developed networks: the total number of connections, the number of live connections (connections for which weight is greater than or equal to 0.5), and the number of connection additions and deletions.

Table 2 shows the results of removing different components of the genome when testing on the network *Large1*. In each case, maximum fitness was achieved with the same number of live connections. However, the total number of connections was greater because removing rules and conditions removed constraints on network size. The genome was constructed not only to develop networks capable of answering all questions correctly, but to do so with the minimal number of connections.

Excluded	Connections	Live	Additions	Deletions
None	49	36	138	89
Rule 4	98	36	98	0
Condition 7	171	36	491	320
Rule 4 and Condition 7	328	36	328	0

Table 2 – The effects of excluding rules and conditions from the genome when developing *Large1*.

Removing rule 4, which prunes superfluous connections, caused the total number of connections to double. However the number of additions decreased, implying that when rule 4 is included some of the connections it removes redevelop. Condition 7 limits connections to inputs from active neurons. Bypassing this caused an even greater increase in the number of connections. This, coupled with the tendency of deleted connections to redevelop, suggests that it is more beneficial to prevent the growth of superfluous connections than it is to delete them once created. Removing conditions 2 to 4 and 8 to 10, which limit connections to neurons of the same type, resulted in the network being unable to answer all questions correctly. Removing these conditions caused connections to form between enablers and collectors such that activation of a predicate's enabler triggered the immediate activation of one or both of its collectors, depending on which collectors were activated during training. All questions were therefore answered true (1,0), false (0,1) or both (1,1), but never unknown (0,0), and so questions for which unknown was the correct answer were answered incorrectly.

4.2 Fitness

Figure 7 shows the change in fitness as the network *Large1* develops, and figure 8 shows the change in fitness after shuffling the event sequence. Note that the initial fitness in both cases is not zero. This is due to the fact that fitness is based on the number of correct collector firings. An undeveloped network will answer all questions as 'unknown' (0, 0). For some test questions, this will in fact be the correct answer, so an undeveloped network automatically answers them correctly. For other questions, target answers are either (1, 0) (true) or (0, 1) (false), meaning that an answer of (0, 0) will be half correct for each of these questions. In summary, the initial fitness is due to the inability of an undeveloped network to fire any collectors and the large number of zeros (instances of collectors failing to fire) in the test data.

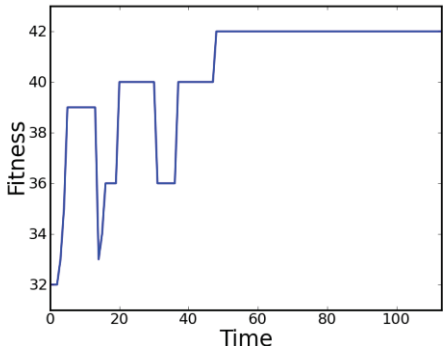


Figure 7 – The change in fitness over time for the network *Large 1*

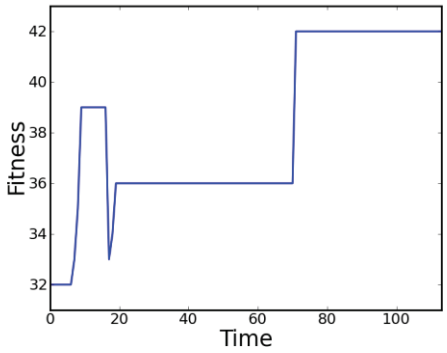


Figure 8 – The development of *Large1* after shuffling the event sequence.

In figures 7 and 8, maximum fitness is eventually achieved, but the fitness decreases and increases again before it reaches the maximum. This behaviour was caused by the weakening of some relations upon the strengthening of others, as described in section 3.1. In order to confirm that it was these conflicting relations that caused the trend of oscillating fitness, the learning experiment was repeated on a simple network which did not contain conflicting relations. The network represented a linear chain of predicates in which each predicate (with the exception of those at the beginning and end of the chain) was the consequent and antecedent of only one other predicate ($P(x,y) \rightarrow Q(x,y)$, $Q(x,y) \rightarrow R(x,y)$ …. $T(x,y) \rightarrow U(x,y)$). Figure 9 shows that in this case, the fitness only increased and never decreased, as no learned relations were disturbed by the learning of others.

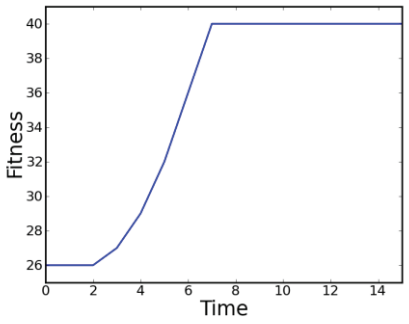
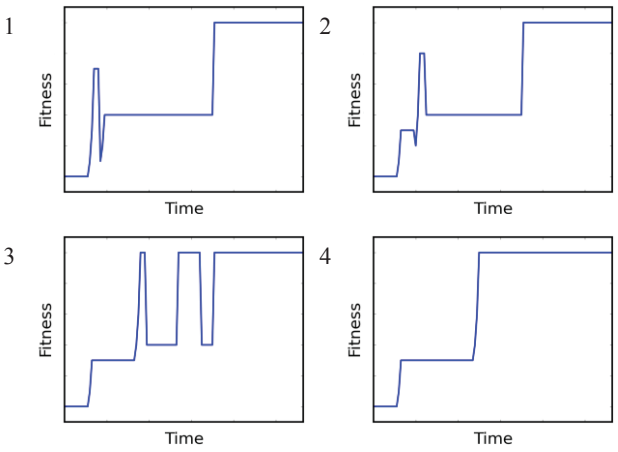


Figure 9 – The development of a network representing a linear chain of predicates

In the process of learning conflicting relationships which affect each other in this way, relations are learned and unlearned until both have been observed enough times (usually about 3) to support evidence for both, at which point both relations are successfully represented. This learning and unlearning of relations affects the truth values of predicate instances that depend on them, meaning that the correct assertion of these predicate instances is also periodic until the network settles. As a consequence, questions are periodically answered correctly and incorrectly before they can be consistently answered correctly, which explains the peaks and troughs in the graph. How soon a network settles into a state whereby this behaviour stops depends on the number of event sub-sequences supporting each relation and on the order in which they occur. This is due to the fact that the magnitude of change depends on the value of α, which is defined slightly differently for the weakening and strengthening of weights (as in equations 3 and 4 respectively) but is inversely proportional to the number of updates in both cases. In other words, how long the network takes to settle depends on how many times connections have been strengthened and how many times they have been weakened.

For example, consider the shuffled event sequence used for learning *Large1* as shown in figure 8. The set of relations dependent on $Q(x,y)$ is $[Q(x,y) \rightarrow \neg R(x,y), \neg R(x,y) \rightarrow \neg T(x,y), \neg R(x,y) \rightarrow \neg U(x,y)]$. The set of sub-sequences that supports this set will be referred to as X. The set of relations dependent on $\neg Q(x,y)$ is $[\neg Q(x,y) \rightarrow S(x,y), S(x,y) \rightarrow V(x,y)]$. The set of event sub-sequences supporting evidence for this set will be referred to as Y. The initial peak in fitness occurs when a member of X, X_1, completes at *t=7*, and drops again when Y_1 completes at *t=17*. Y_2 and Y_3 then complete at *t=25* and *t=39*. X_2 completes at *t=54*, but has no effect on fitness because the number of instances of X isn't enough to balance the connection weights. Y_4 completes at *t=66*. X_3 completes at *t=72* and instances of both X and Y have now occurred enough times that the relationships they each support are strong enough to maintain maximum fitness without the observation of one disturbing the relationship supported by the other. After X_3, further instances of X and Y occur interchangeably but fitness does not drop now that the relationships are balanced.

This hypothesis as to why the peaks and troughs occur was tested by moving X_1 further along the timeline of events in order to move the initial peak in fitness seen in figure 8 along the graph. This is demonstrated in figure 10. In the first image, X_1 is moved to occur just before Y_1, causing the peak to become narrower. In the second image, X_1 is moved to occur after Y_1 but just before Y_2, creating another narrow peak as X_1 improves fitness but Y_2 reduces it again. In the third image, X_1 is moved to occur just before Y_3 to temporarily increase fitness to the maximum. After Y_3 causes fitness to drop soon after, X_2 is able to increase it again, for a bit longer, before Y_4 causes a drop in fitness once more. After X_3, the network is balanced. In the final image, X_1 is moved to occur just before X_2 and this balances the relations. Note that unlike the other graphs, the third graph contains two peaks before fitness settles. In this case, instances of X and Y alternate more than they do in others, and the graph contains the greatest number of fitness peaks before the network settles. In the fourth graph, there are no peaks and only one change from Y to X before the network settles. Furthermore, the network settles slightly earlier than in the other trials. These behaviours imply that fitness peaks are caused by observation sub-sequences that alternate more, and that conflicting relations can be learned more quickly when the evidence for them alternates less.



1.	X_1	Y_1	Y_2	Y_3	X_2	Y_4	X_3	X_4	Y_5	X_5	Y_6	X_6
2.	Y_1	X_1	Y_2	Y_3	X_2	Y_4	X_3	X_4	Y_5	X_5	Y_6	X_6
3.	Y_1	Y_2	X_1	Y_3	X_2	Y_4	X_3	X_4	Y_5	X_5	Y_6	X_6
4.	Y_1	Y_2	Y_3	X_1	X_2	Y_4	X_3	X_4	Y_5	X_5	Y_6	X_6

Figure 10 – The effects of moving sub-sequence X_1, which supports evidence for relations depending on Q(x,y), further along the timeline. The resultant ordering of sub-sequences for each graph is displayed at the bottom.

This tendency of fitness to rise and fall before development is complete is similar to a phenomenon referred to as U-shaped development which has been observed in various aspects of cognitive development [2, 4, 17, 21]. One example from natural language is the way children learn past tense conjugation [21]. In early stages of natural language development children will know some regular and irregular past tense verbs and apply them correctly. However, data shows that once they realise that a large number of verbs are conjugated by the addition of 'ed' to those verbs, they over-generalise this rule to the irregular verbs as well, ignoring the irregularities they have already acquired and incorrectly conjugating them like any other. For example, upon noticing correct conjugations such as the derivation of 'reached' from 'reach' and 'heated' from 'heat', they often derive 'eated' from 'eat', 'goed' from 'go', and so forth, even though they correctly used 'ate' and 'went' before. Only once they have had more exposure to the English language and have heard regularities and irregularities frequently enough do they realise that the addition of 'ed' does not apply to all verbs. They are then able to apply regularities and irregularities correctly once again. In summary, the child's language ability gets worse before it improves. Though correction by adults may play some part in this process, it is largely credited to observations of how others use language. Errors tend to occur with verbs heard less often. Only when an irregularity is observed enough times is that irregularity able to 'block' the application of the rule to a verb stem. In SHRUTI's learning system, conflicting relations also require a sufficient number of observations before the representation of both relations can be balanced.

U-shaped development in children has been observed in a range of other cognitive tasks [2, 4, 17]. The U-shaped development observed in the SHRUTI learning model gives it another level of biological plausibility. Of course, the U-shaped development observed in SHRUTI is not caused by rules being over-generalised to irregularities but by rule pairs for which antecedents are of the same predicate but have different signs. However, the developmental process is similar in the sense that it is influenced by observations that may result in the ability of the developing structure declining before it is able to improve even further. It should also be noted that SHRUTI's U-shaped development is a result of the learning process and not of the developmental model which was implemented for these experiments. Nonetheless, the results above have been useful in determining that the genome model is able to support weights that continually gain and lose strength before they are consistently strong enough to represent relations. There was always the danger that a weight would lose enough strength that the genome would prune the connection before it was given a change to gain its strength back. This was not the case.

SHRUTI's U-shaped development will need to be taken into consideration when assessing the fitness of genomes in planned attempts to evolve them. A genome in the population which exhibits a lower fitness may have more potential for improvement than a genome with a slightly higher fitness. It may be necessary to adjust the measurement of fitness so that this potential is also taken into account, in addition to the number of questions a developed network can answer correctly. However, the challenge this idea presents is that of finding a way to quantify this potential.

5 CONCLUDING REMARKS

A scalable genome encoding of basic SHRUTI networks has been produced. A genome constructed using the presented model was successful in growing neural connections in SHRUTI networks such that those networks were able to correctly answer all their test questions correctly. Due to the rule-instructed growth, the size of the genome is independent of that of the phenotype, i.e. a network representing a logic program. The model applies the reuse of sub-structure as used by Gruau and Kitano. The four rules depicted in figure 4 share some repeated conditions, but these are only encoded once in the genome. Encoding these rules separately would have resulted in repeated encoding of these conditions, thus reducing the compactness of the genome.

The genome model proposed contributes towards two goals of neural-symbolic integration. For those interested in the practical application of neural-symbolic networks, a scalable means of representing them has been produced. With regards to developing a working model of logic representation in the human brain, this model is relevant because artificial development and neural-symbolic implementation both claim some degree of biological plausibility. The biologically plausible traits currently exhibited by the system as a whole include the indirect encoding and gradual development of the phenotype, the temporal synchrony and Hebbian learning employed by SHRUTI, and the U-shaped development observed in the change in fitness over time. However, one function that the current genome model lacks that would otherwise increase its biological plausibility further is the production of neurons. The current genome model only develops connections between neurons and not the neurons themselves. Biological development produces neurons and the genome model presented should eventually be updated to include rules for neuron production also. Cell division would be a suitable, biologically plausible means of implementing this.

The model presented is the first step towards producing SHRUTI networks, and possibly other types of neural-symbolic network, using artificial development. The next stage is to attempt the production of these genomes using an evolutionary algorithm. The current fitness function only takes into account the number of questions a network can answer correctly. However, it should be adapted to also take into account a network's potential to improve through further development. Successful evolution of these developmental genomes would contribute towards a means of producing artificial models of cognition that takes inspiration from the way cognitive structures emerge through natural evolution.

7 REFERENCES

[1] Bader, S., Hitzler, P (2005) "Dimensions of Neural-Symbolic Integration" in *We Will Show Them: Essays in Honour of Dov Gabbay*, **1**, College Publications, pp167-194.

[2] Baylor, A.L. (2001) "A U-shaped model for the development of intuition by level of expertise", *New Ideas in Psychology*, **19**, pp237-244.

[3] Chavoya, A. (2009), "Artificial Development", *Foundations of Computational Intelligence*, **1**, Springer, 185-215.

[4] Davis, J. (1997) "Drawing's Demise: U-shaped Development in Graphic Symbolization", *Studies in Art Education*, **38**, No 3, pp132-157.

[5] Dawkins, R. (1986), "The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design", Norton, New York.
[6] Eggenberger, P. (1997) "Creation of Neural Networks Based on Developmental and Evolutionary Principles" in *Proceedings of the International Conference on Artificial Neural Networks*, Springer-Verlag, pp337-342.
[7] Eggenberger, P., Gómez, G., Pfeifer, R. (2003) "Evolving the Morphology of a Neural Network for Controlling a Foveating Retina – and its Test on a Real Robot" in *Proceedings of the Eighth International Symposium on Artificial Life*, pp243-251.
[8] Gruau, F. (1994) "Automatic Definition of Modular Neural Networks", *Adaptive Behaviour*, **3**, No 2, pp152-183.
[9] Hammer, B., Hitzler, P (2007), "Perspectives of Neural-Symbolic Integration", Springer.
[10] Hebb, D.O. (1949) *"The Organization of Behaviour: A Neuropsychological Theory"*, Wiley.
[11] Khan, G.M., Miller, J.F., Halliday, D.M. (2010) "Intelligent Agents Capable of Developing Memory of Their Environment" in Loula A., Queiroz J., editors, *Advances in Modelling Adaptive and Cognitive Systems*, UEFS, pp77-114.
[12] Kitano, H (1990) "Designing Neural Networks Using Genetic Algorithms with Graph Generation System", *Complex Systems Journal*, **4**, pp461-476.
[13] Kitano, H. (1994) "Neurogenetic Learning: An Integrated Method of Designing and Training Neural Networks Using Genetic Algorithms", *Physica D: Nonlinear Phenomena*, **75**, No 1-3, pp225-238.
[14] Kitano, H. (1995), "A Simple Model of Neurogenesis and Cell Differentiation based on Evolutionary Large-Scale Chaos", *Artificial Life*, **2**, pp79-99.
[15] Lindenmayer, A. (1968) "Mathematical Models for Cellular Interactions in Development", *Journal of Theoretical Biology*, **18**, No 3, pp280-299.
[16] Marcus, G. (2001) "Plasticity and Nativism: Towards a Resolution of an Apparent Paradox" in *Wermter S., Austin J., Willshaw D., editors, Emergent Neural Computational Architectures Based on Neuroscience, Lecture Notes in Computer Science*, **2036**, Springer Berlin / Heidelberg, pp368-382.
[17] McNeil, N.M. (2007) "U-shaped Development in Math: 7-Year-Olds Outperform 9-Year-Olds on Equivalence Problems", *Developmental Psychology*, **43**, No 3, 687-695.
[18] Miller, J.F., Thomson P. (2000) "Cartesian Genetic Programming" in Proceedings of the 3rd European Conference on Genetic Programming, **1802**, Berlin, Springer-Verlag, pp121-132.
[19] Nolfi, S., Parisi, D. (1995) "Evolving Artificial Neural Networks That Develop in Time" in Morán F., Moreno A., Merelo J., Chacón P., editors, *Advances in Artificial Life, Lecture Notes in Computer Science*, **929**, Springer Berlin / Heidelberg, pp353-367.
[20] Nolfi, S., Parisi, D. (1996) "Learning to Adapt to Changing Environments in Evolving Neural Networks", *Adaptive Behaviour*, **5**, pp75-98
[21] Pinker, S. (1999) "Kids Say the Damedest Things" in *Words and Rules: The Ingredients of Language*, pp189-210.
[22] Shastri, L., Ajjanagadde, V. (1993) "From Simple Associations to Systematic Reasoning", *Behavioral and Brain Sciences*, **16**, No 3, pp417-494.
[23] Shastri, L. (1999) "Advances in SHRUTI – A Neurally Motivated Model of Relational Knowledge Representation and Rapid Inference using Temporal Synchrony", *Applied Intelligence*, **11**, pp79-108.
[24] Siebel, N.T., Sommer G. (2007) "Evolutionary Reinforcement Learning of Artificial Neural Networks", *International Journal of Hybrid Intelligent Systems*, **4**, No 3, pp171-183
[25] Stanley, K.O., Miikkulainen R. (2002), "Efficient Evolution Of Neural Network Topologies" in *Proceedings of the Genetic and Evolutionary Computation Conference,* Morgan Kaufmann, pp1757-1762.
[26] Wendelken, C., Shastri, L. (2003) "Acquisition of Concepts and Causal Rules in SHRUTI" in *Proceedings of the Twenty Fifth Annual Conference of the Cognitive Science Society*, Boston.