Postgraduate Conference 2014

Technical report 1-14

Edited by:
Suwimol Jungjit
Konstantin Kapinchev

School of Computing
University of Kent

Canterbury
30 May 2014

Preface

It is our pleasure to introduce you to the Postgraduate Conference 2014, organised at the School of Computing, where students demonstrate their latest research and contribution to the scientific community. This annual event became a tradition at the School of Computing, and this year is no exception. The conference presents the excellence of the research in computer science conducted by the school's postgraduate students. It covers a wide range of areas, including parallel GPU computing, data mining, artificial intelligence, cloud computing, bioinformatics, and others.

We would like to express our gratitude to the authors for participating in this conference. We hope your research will be successful and rewarding.

We also would like to thank the review panel and the academic and administrative staff for their assistance in organising the conference. This event will not be possible without their help and support.

Kindest regards,

Suwimol Jungjit

Konstantin Kapinchev

Editors

Table of Contents

Conference Programme

Time:	Event:
10:30 – 12:30	Poster session (room sw101)
	Cellular-Automaton-Entropy-Based Project Scheduling in Cloud Computing, *Huankai Chen*
	Seeing Is Not Always Believing -The relationship between performance and subjective visibility in an attentional blink task, *Luise Gootjes-Dreesbach and Howard Bowman*
	Parallel Implementation of Digital Signal Processing Algorithms in Optical Coherence Tomography, *Konstantin Kapinchev and Fred Barnes*
	Dynamic Time Warping as a Method for Measuring Latency Differences of ERP Components, *Alexia Zoumpoulaki, Abdulmajeed Alsufyani, Marco Filetti, Michael Brammer and Howard Bowman*
	Presentations (room sw101)
13:30 – 13:50	A Method for Graph Drawing Utilising Patterns, *Robert Baker and Peter Rodgers*
13:50 – 14:10	Genetic Algorithms in the Study of the Genetic Code Adaptability Problem, *Lariza Laura de Oliveira, Renato Tinós and Alex A. Freitas*
14:10 – 14:30	Gene Ontology Hierarchy-Based Feature Selection, *Cen Wan and Alex A. Freitas*
14:30 – 14:50	A Machine for Higher-Order Term Rewriting, *Connor Lane Smith*
14:50 – 15:00	Brief talks from poster session
15:00 – 15:10	Closing the conference

Papers and Extended Abstracts

A Method for Graph Drawing Utilising Patterns

Robert Baker
University of Kent
Canterbury, Kent
United Kingdom
rb440@kent.ac.uk

Peter Rodgers
University of Kent
Canterbury, Kent
United Kingdom
P.J.Rodgers@kent.ac.uk

ABSTRACT

This paper describes work in progress of a new method for drawing straight line graphs. By identifying certain predefined patterns with a graph and drawing these in a consistent manner, it is hoped that a useful and aesthetically pleasing layout can be achieved. The paper will detail the patterns identified and the various combinations of connections between patterns. The algorithms for drawing each connection are then detailed. As this is a work in progress, there has been no evaluation, although future work will be identified.

1. INTRODUCTION

There are a number of existing drawing methods for node-linked graphs, such as force-directed methods [6], simulated annealing [5], and constraint-based layout [2]. This paper, however, proposes a new method that utilizes subgraph isomorphism. The proposed solution is that certain subgraphs (or patterns) are identified within a graph and these patterns are drawn in a consistent manner. There are challenges regarding the drawing order and integrating the concept of an "ideal layout" with the previously drawn set of patterns. The basic process is as follows:

1. Patterns are identified
2. Connections between each pattern are determined
3. These patterns are placed in an order for drawing
4. Each pattern is drawn

By drawing patterns in a consistent manner, it is hoped that the result will be clearer, more compact and easier to understand. Patterns have also been used for layout in editing tools [10], however this a different concept to that we propose here because the patterns encapsulate constraints and attribute evaluation rules, rather than the simple layout definitions that we use the terminology "pattern" for.

2. RELATED WORK

There has been much research in both graph drawing and subgraph isomorphism. In this chapter, related work in both fields is discussed.

2.1 Graph Drawing Techniques

Graph drawing is a well researched area of information visualization. Techniques include force-directed methods, simulated annealing, and those based on hill climbing techniques. It is important to study other research of graph drawing techniques so that the proposed method can be compared.

2.1.1 Force-directed methods

Most graph drawing techniques are based on force-directed methods. This method was developed by Eades[6], and represents nodes as charged particles which have a repulsive force against all other nodes. Edges are represented as springs charged with attractive forces between the nodes they connect, and both these forces relate to the distance between the respective nodes. These forces move their respective nodes and when they reach rest, or a certain number of iterations have been completed, a final layout is created. Graphs drawn using this technique often have aesthetically pleasing layouts, but there can be some issues such as occlusion and local optima.

Fruchterman and Reingold [7] introduced improvements by using linear calculations of the forces, as well as limiting the repulsive forces to act only on nodes nearby rather than the entire graph.

Other variations have been created, including those that: create the concept of an ideal distance between non-connected nodes [8], orientate egdes [13], introduce edge repulsion [9], cluster nodes into smaller graphs [15], or combine force-directed methods with other drawing techniques, such as Euler diagrams [1].

2.1.2 Search Based Technique

Although force-directed methods sometimes produce aesthetically pleasing graph layouts, they only model aesthetics indirectly. Another possible drawing technique is a search based method, such as simulated annealing or a hill climber.

In simulated annealing, the graph initially moves large distances which are then "cooled" slowly to only allow more minimal movements. This helps the system to avoid local optima and, when given enough time, to find the global optimal solution. Davidson and Harel [5] used this method to draw graphs and their results compared favourably with manually drawn graphs. However, simulated annealing is computationally slower than a force-directed method.

One disadvantage of simulated annealing is that nodes may move to a worse position. Therefore, a method that prevents this, called hill climbing, was developed. Stott and Rodgers [12] developed a hill climbing multi-criteria optimization technique to automatically generate metro map layouts.

2.2 Subgraph Isomorphism

Further research has also been conducted on subgraph isomorphism. Such research includes detecting subgraphs, and drawing large graphs which are based on previously defined

or drawn subgraphs.

Ullmann [14] developed a method that creates adjacency matrices for the graph and subgraphs. It uses these matrices to detect isomorphism. This algorithm is very computationally expensive to use for the detection of subgraphs within a larger graph because it runs in cubic time.

Bonnici et al [3] developed a subgraph isomorphism algorithm utilising a search based strategy, specifically to identify subgraphs in biochemical data. As with other methods, the authors use a tree based search approach to detect isomorphism. If there is a statistically significant increase in the number of subgraphs in a particular dataset compared to random graphs, then this set is of interest to the user.

Subgraph isomorphism can be used for such tasks as identifying molecular structures [11], interpreting semantic diagrams [4], amongst others.

3. DEFINITIONS OF PATTERNS

At present, we define four types of common subgraph that can be identified within many graphs. These are cliques, circles, stars & paths and are defined as follows:

3.1 Clique

Must contain at least 4 nodes and all nodes are connected to all the others within the clique. An example of cliques of various sizes is shown in Figure 1. In figures throughout this work, cliques will be highlighted in pink and nodes prefixed with a `c`.

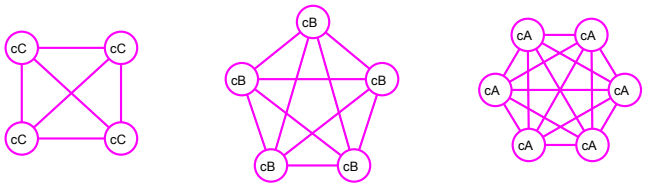


Figure 1: Example of cliques of size 4, 5 and 6

3.2 Circle

Must contain at least 4 nodes and each node must connect to exactly 2 other nodes in the pattern so that a closed path is formed. Each node may connect to any number of other nodes not in the pattern. An example of circles of various sizes is shown in Figure 2. In figures throughout this work, circles will be highlighted in blue and nodes prefixed with a `o`.

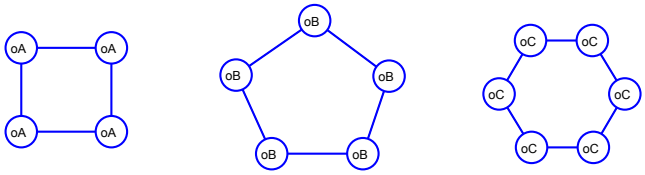


Figure 2: Example of circles of size 4, 5 and 6

3.3 Star

Must contain at least 4 nodes where one node is the central node that connects to the other nodes. The other nodes must only connect to the central node in this pattern, but can connect outside of the star. An example of stars of various sizes is shown in Figure 3. In figures throughout this work, stars will be highlighted in red and nodes prefixed with an `s`.

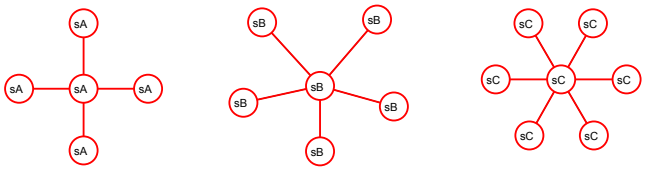


Figure 3: Example of stars of size 4, 5 and 6

3.4 Path

Must contain at least 4 nodes and the terminating nodes may connect to any other nodes outside of the pattern. Other nodes in the pattern must connect to only 2 other nodes, the previous and next node in the path. Any paths that are also circles are treated as circles only. An example of paths of various sizes is shown in Figure 4. In figures throughout this work, paths will be highlighted in green and nodes prefixed with a `p`.

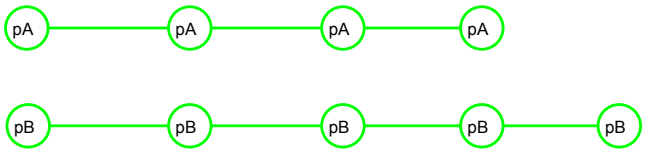


Figure 4: Example of paths of size 4 and 5

4. DRAWING ORDER

Once the patterns within a graph have been identified, the method will then determine in which order these patterns are drawn. One heuristic is to order the patterns so that closely related patterns (edge sharing and node sharing) are drawn first. This is because these patterns have less flexibility in their layout and it is sensible to draw these before the graph becomes too dense and a location cannot be found. When determining the order, several considerations need to be made:

- The type of connection to the previous drawn patterns
  - 1 Node shared
    - The pattern and the drawn set have one node in common
  - 2 Nodes shared
    - The pattern and the drawn set have two nodes in common. This will often include circles and paths connected to non-consecutive nodes in circles. Many other instances of 2 nodes sharing are actually identified as one shared edge
  - 1 Edge shared
    - The pattern and the drawn set have one edge in common. This will often include cliques and stars that share edges with other patterns.
  - 2 Edge shared

        - A star shares its central node with a circle, so two edges of the circle and a number of other edges are the "spokes" of the star.
    - Edge connection
        - The pattern and the drawn set have one edge connecting them. This edge is in neither the drawn set, nor the pattern
    - No connection
        - The pattern and the drawn set have no nodes or edges in common, nor have one edge connecting them. This does not mean the pattern is necessarily disconnected from the drawn set - it could be connected by a series of nodes and edges. This is not considered to be a close connection, so this is treated as no connection.
- The amount of connectivity to the previously drawn patterns
    - Number of nodes shared
    - Number of edges shared
- Degree of pattern
- Type of pattern

It is also necessary to discard a number of patterns that are identified. Because many graphs have an unfeasibly large number of patterns or patterns which considerably overlap others. This makes the process of drawing these patterns incredibly difficult, and therefore only a subset of patterns for drawing. As a result of this, patterns which share more than 2 edges or nodes with the drawn set are disregarded.

The patterns that have been identified are drawn in the following order:

1. Clique with 1 edge shared
2. Clique with 1 node shared
3. Star with 1 outside node shared
4. Star with 1 edge shared
5. Star with centre node shared
6. Circle with 2 edges shared
7. Circle with 2 nodes shared
8. Circle with 1 node shared
9. Path with 2 nodes shared
10. Path with 1 node shared
11. Clique with a connecting edge
12. Star with a connecting edge
13. Circle with a connecting edge
14. Path with a connecting edge
15. Clique with no connection
16. Star with no connection
17. Circle with no connection
18. Path with no connection

Once the order of drawing is established, certain cliques are manipulated. This is necessary because it helps to improve the final layout. The nodes within cliques that have a shared edge require reordering to ensure this particular edge is on the "outside" of the pattern. This helps avoid occlusion and to minimize edge crossings.

5. DRAWING EACH PATTERN

Once the drawing order has been established, the patterns can be drawn. The algorithm for drawing each pattern is different for each type of connection and, in some cases, the type of pattern. Not all connection types have yet had their drawing algorithms implemented. This means that only a subset will be described in detail.

Cliques, circles and stars can be drawn according to an "ideal layout" which is the default layout of a pattern if it had no connections. The default layout also helps to ensure consistency when drawing patterns. Slight modifications may be required to enable the ideal layout to handle connections, but these are kept at a minimum to maximise consistency. The layout of paths is flexible. This means that they do not have an ideal layout to follow, as they are usually drawn later. This enables greater scope for a closer relationship between the drawing method and the available graph space. The layout of paths has not yet been implemented.

The first pattern drawn always follows its ideal layout. Patterns are drawn according to the techniques listed below.

5.1 Share One Edge

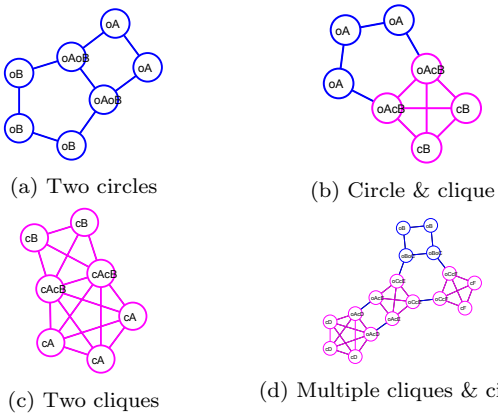
Circles and cliques share the same drawing algorithm when patterns share one edge within the drawn set. Unlike circles and cliques, stars have a unique drawing method. It is not possible for paths to share an edge with any other pattern.

5.1.1 Circle & Clique

Circles and cliques currently share the same algorithm for drawing. In this algorithm, the pattern is first drawn in its ideal layout, centred on the origin of the graph. The pattern is then scaled so all outside edges match the same length as the shared edge. This ensures the pattern does not distort the currently drawn set and that this pattern is drawn in a consistent manner (i.e. a clique of sized 4 is still drawn as a square, rather than a trapezium).

The pattern is then rotated to match the original orientation of the shared edge and moved into the correct location. After rotation, the shared edge and node have been returned to the correct position. However, the remaining pattern may be inside the drawn set. This could mean the pattern is positioned in a way that creates unnecessary edge crossings or occlusions. To overcome such issues, the pattern may be reflected. Reflection enables incorrectly placed nodes to move to a more aesthetically pleasing position.

When reflection has been completed, the pattern sits in the correct location.



(a) Two circles

(b) Circle & clique

(c) Two cliques

(d) Multiple cliques & circles

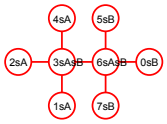
Figure 5: Examples of the Edge Sharing algorithm for Circles & Cliques

5.1.2 *Star*

Although the ideal layout of stars is different, it follows a similar drawing method to that used for cliques and circles.

The ideal layout usually arranges stars at an appropriate distance from the centre. The stars are also evenly spaced around the centre. However, the ideal layout cannot always follow this format if other patterns cause occlusions. To make provision for this, the drawing algorithm recognizes invalid areas which are used to restrict the range in which star spokes can be drawn.

Once the largest valid area has been identified, the star is drawn in its ideal layout. This layout arranges the spokes of the star consistently within this valid area, excepting shared edges. The algorithm then follows the same process used for cliques and circles (see Section 5.1.1).



(a) Two stars of size 4

(b) Two stars of size 5 & 6

Figure 6: Examples of the One Edge Sharing algorithm for Stars

5.2 Share Two Edges

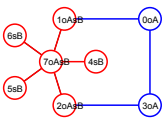
The only case where patterns share two edges is between circles and stars. This exists where a circle shares two "spokes" of a star. A decision had to be made regarding the order in which the star or circle should be drawn. One possible option was to the star around the circle. However, this would require changing the main drawing order (as stars are drawn before circles). It was decided that it was unwise to change the order, as this could have a large number of implications. It was also decided that it was not sensible to introduce an exception for this type, as that defeats the object of having a fixed drawing order. Therefore, circles are drawn after stars.

The method follows a similar process to the the One Edge Sharing drawing method (see Section 5.1.1) for Circles. Firstly, the middle node of the star is found and a virtual edge is created between the other two shared nodes. This is used to enable compatibility with the One Edge Shared method. The pattern (with the exception of the middle node) is then drawn in its ideal layout, centred on the origin of the graph. The circle is then scaled so that the virtual edge is the same length as it was before the circle was drawn. This ensures that the pattern does not distort the currently drawn set.

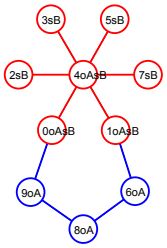
The pattern is then rotated to match the original orientation of the virtual edge and then moved into the correct location. Once this is complete, an orientation check is completed. A measure of the edge crossings and occlusion is taken. If there are no edge crossings or occlusion, then the drawing is complete. If not, the pattern is reflected along the virtual edge. Another calculation of the edge crossings and occlusion is taken. If this is lower than the previous calculation, then this layout is chosen. If not, the pattern is reflected back and the method is complete.

5.3 Connected by Edge

The algorithm for drawing a pattern connected by an edge to the drawn set utilises a search based approach. A grid



(a) A circle of size 4 and a star of size 5



(b) A circle and star of size 6

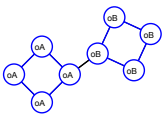
Figure 7: Examples of the Two Edge Sharing algorithm for Circles

of possible options is created, and for each of this options various combinations of rotations and reflections are tested. Numerous metrics are calculated to represent this. These include:

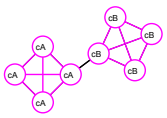
- Number of edge crossings in the graph
- Length of the connecting edge
- Distance between the pattern and the currently drawn set
- A measure of symmetry (the distance between the centres of the pattern and the drawn set)

For a combination to be considered, the separation distance must be larger than a constant value. If it is valid for consideration, the number of edge crossings are examined and if lower than the previous best, the current combination is considered to be the current best.

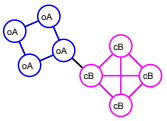
However, the values may be equal. When this happens, the length of the connecting edge is compared. If this length is less than the previous best length, this combination is ranked as the best. This process is repeated a further two times with the search space being a factor of ten smaller each time. This means that the spacing between each combination and the overall search space is ten times smaller. Once this process is complete, the pattern is drawn in the best location and combination of rotation and reflection. However, this algorithm is not yet fully implemented and is likely to be modified.



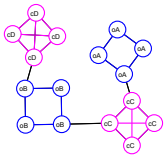
(a) Two circles



(b) Two cliques



(c) Circle & Clique



(d) Mixture of patterns

Figure 8: Example of the Edge Connected algorithm

5.4 Currently Unimplemented Connections

There are currently unimplemented drawing methods for the following types of connection:

- Patterns sharing 1 node

- Patterns sharing 2 nodes
- Patterns connected by a node
- Patterns with no connections

Nodes that do not belong to a pattern must be sensibly located. This has not yet been implemented.

6. CONCLUSION

This work proposes a new method of graph drawing. The method identifies patterns within a graph and draws these in a consistent manner. There are a number of stages to this procedure: identification of patterns, creating an order for drawing, and drawing each pattern in turn. This method has a number of real world applications and graph drawing is a heavily researched area (see Section 2).

It is hoped that this method will allow undirected straight-edge graphs to be drawn in an aesthetically pleasing and useful layout. Although not all algorithms for drawing connections have been implemented, the results from those that have are positive. They appear to show the proposed method has promise and can produce aesthetically pleasing layouts.

However, there is much work still to be completed. Firstly, the remaining drawing methods need to be implemented and tested. Several of these methods may have different algorithms for each of the patterns. Once this is complete, it will be necessary to test the system thoroughly and for modifications to be implemented for corner cases.

In order to determine the system's effectiveness, various graphs will need to be drawn from various datasets. Such datasets may include social network data, relational data, or character analysis in a novel (where characters that appear within a certain number of words of another are said to be related). It will be necessary to find, extract and process this data for use within the system.

Once various datasets have been obtained, it will then be necessary to evaluate the system. The system can be compared to previous work using a number of metrics, such as edge crossings, total area and time taken (both in a quantitative value and time complexity). It will also be necessary to compare the system to other methods by empirical study, to compare the quality of the drawn graph.

There is also scope for future research from this work. Further enhancements would involve identifying different orders and creating different patterns. It is also possible that further research could be spent investigating and developing a system where a user can define both a pattern and how this pattern should be drawn. This addition is not trivial and would require considerable extra work.

7. ACKNOWLEDGMENTS

The author acknowledges the useful discussions from Kai Xu and Rik Waller, both from Middlesex University.

8. REFERENCES

[1] R. Baker, P. Rodgers, S. Thompson, and H. Li. Multi-level Visualization of Concurrent and Distributed Computation in Erlang. In *The 19th International Conference on Distributed Multimedia Systems (DMS 2013)*, 2013.

[2] K.-F. Böhringer and F. N. Paulisch. Using Constraints To Achieve Stability In Automatic Layout. In *CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, number April, pages 43–51, 1990.

[3] V. Bonnici, R. Giugno, A. Pulvirenti, D. Shasha, and A. Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14 Suppl 7(Suppl 7):S13, Apr. 2013.

[4] H. Bunke. Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):574–582, 1982.

[5] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, Oct. 1996.

[6] P. Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984.

[7] T. M. J. Fruchterman and E. M. Reingold. Graph Drawing by Force-directed Placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[8] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(April):7–15, 1989.

[9] C. Lin and H. Yen. A new force-directed graph drawing method based on edgeâĂŞedge repulsion. *Journal of Visual Languages & Computing*, 2012.

[10] S. Maier and M. Minas. A Pattern-Based Layout Algorithm for Diagram Editors. In *Electronic Communications of the EASST Proceedings of the Workshop on the Layout of (Software) Engineering Diagrams (LED 2007)*, volume 7, 2007.

[11] D. H. Rouvray and A. T. Balaban. Chemical applications of graph theory. *Applications of Graph Theory*, 177:155–156, 1979.

[12] J. Stott and P. Rodgers. Metro map layout using multicriteria optimization. In *Eighth International Conference on Information Visualisation, 2004.*, volume 17, pages 355–362, Jan. 2004.

[13] K. Sugiyama and K. Misue. Graph drawing by the magnetic spring model. *Journal of Visual Languages and Computing*, 6(3):217–231, 1995.

[14] J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

[15] C. Walshaw. A Multilevel Algorithm for Force-Directed Graph Drawing. In *Graph Drawing*, pages 1–22, 2000.

Cellular-Automaton-Entropy-Based Project Scheduling in Cloud Computing

Huankai Chen
Future Computing Group
University of Kent
Canterbury, UK
Email: hc269@kent.ac.uk

***Abstract*— Commercial cloud offerings, such as Amazon's EC2, let users rent computing resources on demand, charging based on reserved time intervals. While this gives great flexibility to elastic project scheduling, the complexity of resource management arises when cloud consumers claim an excellent service level agreement (SLA) for multiple quality-of-service (QoS), such as deadline, cost budget and reliability. Such QoS constrains will limit a cloud system's ability to execute and deliver a project as originally planned. The entropy, as a measure of the degree of disorder in a system, is an indicator of a system's tendency to progress out of order and into a chaotic condition, and it can thus serve to measure a cloud system's reliability for project scheduling. In this paper, cellular automaton is used for modeling the complex multiple QoS constrained project scheduling system. Additionally, a method is presented to analysis the reliability of allocated cloud resources by measuring the average resource entropy (ARE). Furthermore, cellular automaton entropy based cloud resource allocation (CAE-CRA) methodology for scheduling multiple QoS constrained project is proposed to help in cloud scheduling strategies evaluation and comparison. At last, the proposed methodology is implemented under Matlab environment and verified with four basic cloud scheduling strategies, First Come First Served Algorithm (FCFS), Round Robin Algorithm (RR), Min-Min Algorithm and Max-Min Algorithm. The experimental results show that the proposed methodology can provide correctly evaluation and comparison of different cloud scheduling strategies and lead to achieve cost-efficient and reliable resource allocation solutions for scheduling projects on the cloud environment.**

***Keywords*— Resource Allocation; QoS Constrains; Scheduling; Cloud Computing; Cellular Automaton; Entropy; Cost-efficiency; Reliability; Complexity; Order; Chaos;.**

I. APPLICATION OF CA ENTROPY FOR THE RELIABILITY EVALUATION ON CLOUD SCHEDULING SYSTEMS

The theory of cellular automata was initiated by John Von Neumann in his seminal work Theory of Self-Reproducing Automata [9]. It can produce complex phenomenon by simple cell and simple rules, which has the ability to model and simulate the complex system. Since the nineteen eighties, as the evolution of computer technology and the progress of science, cellular automaton theory gets in-depth researched and is widely applied in economic, transportation, physical, chemical, artificial life and other complex systems [1] [2] [3].

A cellular automaton consists of a regular grid of cells, each in one of a finite number of states, such as Black and White. The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighbour (usually including the cell itself) is defined relative to the specified cell. An initial state (time t=0) is selected by assigning a state for each cell. A new generation is created according to some fixed rules that determine the new state of each cell in terms of the current state of the cell and the states of the cells in its neighbour.

In this work, we model the cloud scheduling system's behaviour as a cellular automaton (CA), specifically as a one-dimension CA network, and then calculate the CA entropy to measure the reliability degree of such complex system under different scheduling rules and resource allocation strategies. In this way, the collection of cells that composes the CA consists of a number of cloud resources that are rented for running the project (Each cell of CA corresponding to a cloud resource). The CA rules in our work are described as selected scheduling algorithms as follows:

- **First Come, First Served (FCFS)**
- **Round Robin (RR)**
- **Min-Min**
- **Max-Min**

Each resource gets two performance states: Low Productivity (LP) and High Productivity (HP), which are correspondingly showed as Black and White in a CA grid map. The state of a resource is determined by its performance ratio under specify scheduling rules. The performance ratio of a resource (RPR) is calculated as follow:

$$RPR = \frac{The\ completion\ time\ for\ all\ its\ assigned\ task}{The\ completion\ time\ of\ the\ project(Makespan)} \quad (1)$$

If the RPR of a resource is over 50%, then it is in High Productivity state, otherwise it is in Low Productivity state.

Reliability is one of the basic characteristics of complex system, which changes with system evolution. For cloud scheduling system, as one resource of it suffered enough power (Such power may cause by internal local activities or external force) strikes, it will fall into low productivity state or at the worst case it breaks down, this is called the resource collapse. The collapse resource will influence the productivity state of all other resources and may cause them collapse as well, which lead the scheduling system progress out of order and into a

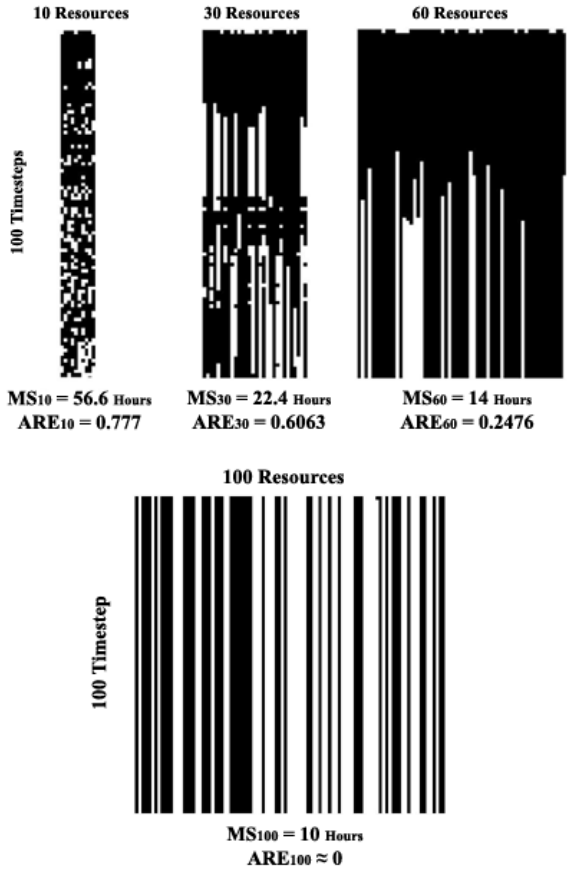


Fig. 1. Examples of Grid Pattern Generated by Cellular Automaton

disorder/chaos condition. Along with the increase in the number of collapse resources, hierarchical expansion, will eventually lead to the collapse of the whole scheduling system. Thus, the scheduling system is failed to deliver the project as original planned. We conclude that:

If a system is in order condition, is more reliable, or vice versa. The reliability can be measured by the disorder degree, thus Average Resource Entropy (ARE), of a system.

To evaluate the reliability of scheduling system in CA, we decrease the computing capacity of one resource by 1% for each time step, with a total of 100 time step, which simulates a resource from full computing capacity till break down. The whole scheduling system's evolution pattern is generated and represented by CA grids. Fig. 1 shows some examples of grid pattern generated by CA for running a project consists of 100 random tasks by FCFS algorithm with different number of allocated resources.

The Average Resource Entropy in CA can be calculated by:

$$ARE = \sum_1^n(-p_{LP} \times \log p_{LP} - p_{HP} \times \log p_{HP})/n \qquad (2)$$

Where n refers to the number of resources that rented for running the project, p_{LP} and p_{HP} refers to probability of Low Productivity State and High Productivity State for a resource respectively.

II. Cellular Automaton Entropy Based Cloud Resource Allocation Methodology (CAE-CRA)

In this section, a multiple QoS constrained Cloud Resource Allocation (CAE-CRA) methodology for scheduling project on the cloud is proposed based on the CA Entropy. The proposed model can be used to achieve the optimal resource allocation strategy by considering both cost-efficiency and reliability for running project on the cloud within deadline and cost budget. The main components and control flow of CAE-CRA model are shown in Fig. 2.

The optimal resource allocation solution selected by CAE-CRA model meets the following condition:

- Meeting project's multiple QoS constrains: deadline, cost budget and reliability threshold
- With the minimum Cost-Efficiency and Reliability Rate (CERR)

Where the Cost-Efficiency and Reliability Rate (CERR) is calculated by Formula:

$$CERR = \frac{MS_n \times \sum_{i=1}^{n} cp_i + ARE_n \times (MS_{n-1} - MS_n) \times \sum_{i=1}^{n} cp_i}{Cost\ Budget} \qquad (3)$$

Where n refers to the number of rented resources to run the project, MS refers to the project's completed time, cp refers to the cost price of a resource and ARE refers to the Average Resource Entropy.

III. Conclusion

In summary, the proposed CAE-CRA methodology is capable of providing useful information and quantitative measurement for aiding the decision maker to achieve an optimal resource allocation and project scheduling solution while meeting the multiple QoS constrains.

References

[1] Toffoli, Tommaso, and Norman Margolus. Cellular automata machines: a new environment for modeling. MIT press, 1987.

[2] Wolfram, Stephen. "Universality and complexity in cellular automata." Physica D: Nonlinear Phenomena 10.1 (1984): 1-35.

[3] Langton, Chris G. "Computation at the edge of chaos: Phase transitions and emergent computation." Physica D: Nonlinear Phenomena 42.1 (1990): 12-37.

[4] LEON, O. "Local activity is the origin of complexity." International journal of bifurcation and chaos 15.11 (2005): 3435-3456.

[5] Matthews, Robert AJ. "The science of Murphy's Law." SCIENTIFIC AMERICAN-AMERICAN EDITION- 276 (1997): 88-91.

[6] Lambert, F.L.: The Second Law of Thermodynamics. http://www.secondlaw.com, 2005.

[7] Amazon EC2, http://aws.amazon.com/ec2/, 2013.

[8] Khinchin, A. Ya. Mathematical foundations of information theory. Dover Publications, 1957.

[9] Von Neumann, John, and Arthur W. Burks. "Theory of self-reproducing automata." (1966).

Seeing Is Not Always Believing - The relationship between performance and subjective visibility in an attentional blink task

Luise Gootjes-Dreesbach
School of Computing
University of Kent
Canterbury, Kent, UK
elg34@kent.ac.uk

Howard Bowman
School of Computing
University of Kent
Canterbury, Kent, UK
H.Bowman@kent.ac.uk

ABSTRACT

Verbal report is among the most widely used measures of consciousness in cognitive science. For example, it has proven useful in exploring the question whether the transition between conscious and subconscious processing is gradual or all-or-none [3]. These studies tend to use the Attentional Blink (AB) paradigm [4]. The AB occurs in Rapid Serial Visual Presentation (RSVP) when identification of two targets is required: performance on reporting the second target declines when it appears between 100-500 ms after a fully processed first target. While subjective experience generally seems to reflect performance, prior data and participants' experience suggests that the relationship between these measures might differ with varying temporal separation of the targets. In this study, we recorded identity and subjective visibility for letter targets in an AB task with digit distractors. We found that metacognitive levels changed with target separation when looking at the correlation between performance and ratings. Additionally, we analyzed the relevance of order errors for this relationship.

Keywords

Attentional Blink, Consciousness, RSVP

1. INTRODUCTION

The subjective experience of items in a RSVP stream is very unusual, and extremely rare in everyday life. The presented stimuli are distractor items that participants are instructed to ignore and usually two target items ($T1$ and $T2$) that they report at the end of each trial. The most common measure is report accuracy dependent on the lag (temporal separation) between the targets. During the AB (around a target separation of 100-500 ms/Lag 3 or 4), it is much less likely that a second target is reported, while at Lag 1, where the second target immediately follows the first, it is more likely for both targets to be reported (Lag 1 sparing), but often in the wrong order (order errors). All these variables make the AB paradigm extremely interesting for studies on psychological concepts such as attention, consciousness and perception. Adding subjective visibility ratings can add greatly to the kind of inferences that can be drawn from these studies, for example when looking at the nature of conscious perception. However, in these studies ratings tend to be collected for only one of the targets and order error trials are usually not treated separately. This study aimed to explore the relationship between ratings and accuracy at this level. As many non-AB studies have shown, ratings and performance are not necessarily highly correlated. A good example of this is the Blindsight phenomenon [2]. Blindsight has been defined as residual visual capacity in the absence of conscious perception [5]: People with damage to their striate cortex who are cortically blind can make judgments about the nature of stimuli presented in their blind spots at an above chance level, while they report to be merely guessing. Such dissociation between subjective experience and performance is indicative of unconscious knowledge in the participants [1].

The main aim of the present study is to explore if the relationship between accuracy and subjective visibility of the T2 on trials where the T1 is correctly reported ($T2 \mid T1$) is dependent on lag. Lag 1 is a particularly interesting case, reflected in the occurrence of order errors as well as Lag 1 sparing. Additionally, the effect of order errors on visibility ratings and how they might inform this relationship is of interest.

2. METHOD

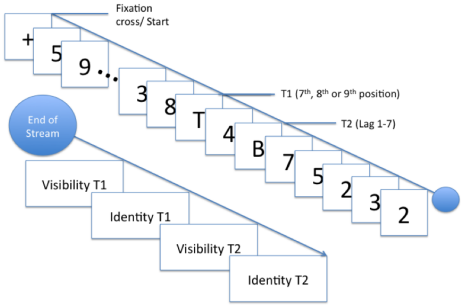


Figure 1: Letter-digit attentional blink task

We recruited 18 participants from the Jobshop website of the University of Kent to take part in a letter-digit version of the attentional blink task (Figure 1). The RSVP stream consisted of two targets ($T1$ and $T2$) drawn from a set of uppercase letters (excluding I, M, O, Q, W), while all other elements were randomly selected single digits (excluding 0 and 1). These items were presented with a stimulus-onset asynchrony (SOA) of 67 ms. At the end of each trial, participants were asked about the identity and visibility of both targets. The whole experiment consisted of 168 trials di-

vided into 4 blocks.

3. RESULTS

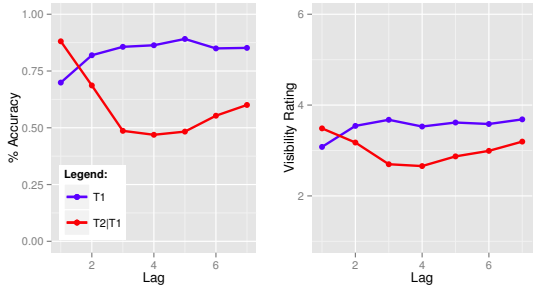


Figure 2: $T1$ and $T2$ | $T1$ accuracy and subjective visibility rating

The fast SOA resulted in high performance and order errors at Lag 1 and 2 (Figure 2). Due to this, our hypothesis predicts a disconnect between accuracy and visibility at these lags. A Lag x Measure (7x2) ANOVA on an aggregate variable of the z-scores of the $T2$ | $T1$ accuracy and visibility of each participant showed a significant interaction, p<.001. $T2$ | $T1$ visibility data was log transformed due to skew. A 5x2 ANOVA excluding Lag 1 and Lag 2 resulted in a non-significant interaction, p=.41. Due to possible non-normality, we confirmed the pattern of results with Friedman tests on the differences between the two measures.

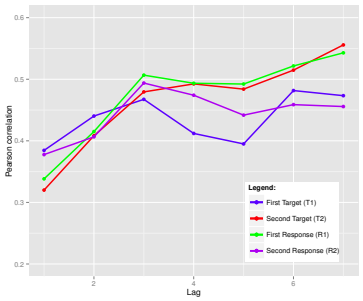


Figure 3: Accuracy/rating correlation for targets ($T1$ and $T2$) and participant's response ($R1$ and $R2$).

To further corroborate these results, the point biserial Pearson correlation between visibility and accuracy was examined (Figure 3, all p <.001). The correlation between $T1$ and $T2$ rating across all trials was fairly high, r=.33, p<.001.

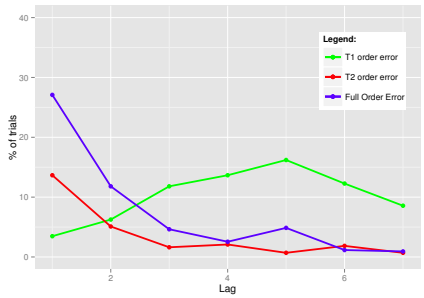


Figure 4: Percentage of order errors by lag.

Figure 4 shows the percentage of order errors at each lag. Wilcoxon tests were run to investigate the effect of order errors on visibility. Visibility was lower for $T2$ than $T1$ when both targets were identified in the correct order, p<.001. When reported in the wrong order, there was no significant difference (p=.82). Both $T1$ and $T2$ visibility was significantly lower for order errors (p <.001 for $T1$, p <.05 for $T2$).

4. CONCLUSIONS

As expected, the relationship between subjective visibility and performance seems to change with proximity of targets. The prediction that the lowest correlations between accuracy and ratings would be found at Lag 1 and 2 has been confirmed. This study does not suggest a specific effect of the attentional blink on the accuracy/visibility correlation. We found preliminary evidence that the lower correlation may be related to order error: Additionally to the Lag 1/Lag 2 full order errors, this dataset shows late $T1$ half order errors ($T1$ reported second and first response incorrect) that coincide with lower correlation. It is possible that the high correlation between $T1$ and $T2$ visibility plays an important role in this but more research is needed to explore this relationship. Currently, studies tend to conduct analyses regardless of reported target order. We have shown that proximity of targets and order errors may need to be taken into account when analysing subjective visibility in AB tasks.

5. REFERENCES

[1] Z. Dienes. Subjective measures of unconscious knowledge. *Progress in brain research*, 168:49–269, 2007.

[2] M. Overgaard. Visual experience and blindsight: a methodological review. *Experimental brain research*, 209(4):473–479, 2011.

[3] J. E. Raymond, K. L. Shapiro, and K. M. Arnell. Temporary suppression of visual processing in an rsvp task: An attentional blink? *Journal of Experimental Psychology: Human Perception and Performance*, 18(3):849, 1992.

[4] C. Sergent and S. Dehaene. Is consciousness a gradual phenomenon? evidence for an all-or-none bifurcation during the attentional blink. *Psychological Science*, 15(11):720–728, 2004.

[5] L. Weiskrantz. Blindsight revisited. *Current opinion in neurobiology*, 6(2):215–220, 1996.

Parallel Implementation of Digital Signal Processing Algorithms in Optical Coherence Tomography

Konstantin Kapinchev
School of Computing
University of Kent

Fred Barnes
School of Computing
University of Kent

ABSTRACT
Digital signal processing (DSP) is a computationally intensive task in real-time systems. In optical coherence tomography (OCT) systems, a standard sequential CPU implementation of the DSP algorithms creates a performance bottleneck and prevents the system from working in real time. A parallel implementation, on the other hand, which utilizes the architecture of the graphics processing unit (GPU), offers the solution to this problem. This study investigates DSP algorithms and how they can be parallelized in order to improve their performance and facilitate real-time operation.

General Terms
Algorithms, Performance, Languages

Keywords
Parallel Computing, General Purpose GPU Computing, Digital Signal Processing, Optical Coherence Tomography

1. INTRODUCTION
OCT relies on the ability of the electromagnetic radiation at the wavelengths of infra-red and near infra-red laser light to penetrate semitransparent materials and to reflect back visual information from different depths below the surface of the test sample. To generate an image from a particular depth, the OCT system must perform a significant amount of DSP, which involves a number of processing steps such as Fourier transforms (forward and inverse), zero padding, window function, cross correlation, and others. If the OCT needs to operate in real time, the processing must be done within short time window, after the acquiring the data and before visualizing the image [1].

The OCT system used in this research, at its current setting, takes 0.8 seconds to scan one frame and another 0.8 seconds to retract its scanning mirrors. The OCT system operates in real-time if the signal processing is completed within the period of 1.6 seconds. A LabVIEW project is used as data acquisition software. It also has the functionality to perform the necessary signal processing steps. By using LabVIEW for both data acquisition and signal processing, the OCT system can produce images from two different depths with a resolution of 200x200 points (pixels). When used in practice, two different images with resolution of 200x200 cannot give all the necessary data. In ophthalmology, for example, while scanning objects such as optic nerves and retinas, clinicians need much more information. The system needs to deliver images from more depths with better resolution. In a CPU bound LabVIEW implementation, an increase of the data will increase the processing time linearly. As a result, the OCT system will not be able to operate in real time. A parallel implementation of the DSP, on the other hand, which uses the highly parallel architecture of the GPU, can deliver higher number of images with increased resolution within the requirements for real-time operation.

2. PARALLEL IMPLEMENTATION OF SIGNAL PROCESSING IN OCT SYSTEMS
The OCT system used in this research produces images from layers below the surface of samples by applying *cross-correlation*, a processing technique largely used in interferometry. Formula (1) presents the *cross-correlation* theorem (FT - Fourier Transform, IFT - Inverse Fourier Transform, $\overline{A}$ - complex conjugate of A) [2].

$$A \cdot B = \int_{-\infty}^{\infty} \overline{A}(t) B(x+t) dt = IFT\left(FT\left(\overline{A}\right) * FT\left(B\right)\right) \quad (1)$$

During its operation, the OCT system collects a number of values (the input signal) from each scanned point (pixel). These values form the *channeled spectra*. Another signal, called *mask*, is collected at an earlier stage, before scanning the tested material. It is obtained by using highly reflective surface as a sample. This signal corresponds to a specific depth. The OCT system can use simultaneously several different masks that correspond to different depths. The cross correlation is performed between these two signals: the *channeled spectra* (A) collected form the sample and the *mask* (B) (Formula 1) [3].

The size (number of values) of the *channeled spectra* and the *mask* must be the same, usually 512 or 1024. To obtain the intensity of a single point from the image the values form the corresponding *channeled spectra* and the values from the *mask* must be multiplied and then the absolute values of all multiplication products must be summed (Formula 2). The multiplication is performed in the frequency domain, so forward and inverse Fourier transforms are applied before and after multiplying.

$$Intensity = \sum_{i=0}^{1023} \left| ChanneledSpectra_i * Mask_i \right| \quad (2)$$

The General Purpose GPU (GPGPU) computing allows the usage of a great number of GPU cores, up to 2880 in NVIDIA TESLA K40, for general computations. The multiplications and summations used in cross correlation can largely benefit from this parallel architecture and thus deliver considerable improvement in the performance. The high number of parallel threads can absorb an increase in the number of images with improved resolution, without significantly increasing the processing time. This allows the OCT system to improve its output and still operates in real time.

3. INTEGRATION

The parallel architecture of the GPU offers better performance, but the GPU does not share its memory with the other modules in the OCT system. Intensive data exchange between CPU bound modules and the GPU application must be performed. In the case of the OCT system used in this research, the exchange is between the LabVIEW project, which acts as a data acquisition software and a primary user interface, and the GPU application, which performs the DSP. These overheads are compensated by the efficiency of the GPU and as a result they do not take significant effect on the performance. The results from this implementation are presented and analyzed in [4].

4. CONCLUSION

The type of signal processing performed in OCT systems can largely benefit from the parallel acceleration delivered by the GPU architecture. A parallel implementation of the DSP can achieve simultaneous generation of multiple images with higher resolutions, while still operates in real time.

5. REFERENCES

[1] Adrian Podoleanu "Optical coherence tomography", The British Journal of Radiology, pp. 976-988, 2005.

[2] Eric Weisstein, "Cross-Correlation Theorem" from MathWorld – a Wolfram Web Resource, [http://mathworld.wolfram.com/Cross-CorrelationTheorem.html]

[3] Adrian Podoleanu, Adrian Bradu, "MS interferometry for parallel spectral domain interferometry sensing and versatile 3D optical coherence tomography", Optical Society of America, 2013

[4] Kapinchev, Barnes, Bradu, Podoleanu "Approaches to General Purpose GPU Acceleration of Digital Signal Processing in Optical Coherence Tomography Systems", IEEE International Conference on System, Man, and Cybernetics, Manchester, 2013

Genetic algorithms in the study of the genetic code adaptability problem

Lariza Laura de Oliveira
Department of Computing and Mathematics
University of São Paulo
Ribeirão Preto, Brazil
larizalaura@gmail.com

Renato Tinós
Department of Computing and Mathematics
University of São Paulo
Ribeirão Preto, Brazil
rtinos@ffclrp.usp.br

Alex A. Freitas
School of Computing
University of Kent
Canterbury, UK
A.A.Freitas@kent.ac.uk

ABSTRACT
The canonical genetic code is almost universal and is present in most organisms. Many researchers believe that the genetic code is a product of natural selection. This hypothesis is supported by its robustness against mutations, when some amino acid property is considered. The most frequently used property is the polar requirement of the amino acids. In this paper, we suggest that robustness considering polar requirement is not the only measure adapted during evolution of the genetic code and we propose to consider other properties at the same time. A Genetic Algorithm is used to generate hypothetical codes, which are evaluated by a robustness-based function. Our results suggest that the use of more properties is promising.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, and Search—*Heuristic methods*

; J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and genetics*

General Terms
Algorithms

Keywords
Genetic Algorithms, Genetic Code

1. INTRODUCTION
The genetic code is the interface between the genetic information encoded as DNA and RNA molecules and the proteins. The cell uses the messenger RNA as a template to assemble the proteins in a process called translation. The RNA sequence is read in series of three nucleotides, called codons. Each codon determines what will be the amino acid codified and these correspondence between codons and amino acids is called genetic code [10, 9].

If we consider all possible codes mapping the 64 codons into 21 amino acids, more than 1.51×10^{84} possible genetic codes would be generated [12]. So, two intriguing questions are: why only one code is used in almost all complex living organisms and why the canonical code was selected over this large number of possible codes [13], [1], [4], [6].

Many researchers argue that the genetic code is a product of natural selection, instead of a frozen accident [1]. This hypothesis is supported by its robustness against mutations when amino acids properties are considered [13]. In fact, according to Haig and Hurst [8] a very small percentage of random codes are better than the canonical code, when a function of robustness considering polar requirement is used to evaluate the codes.

In this context, Genetic Algorithms (GAs) have been used to identify regions of the genetic code space where best codes, according to a given evaluation function [11], can be found. In this paper, using a GA, we investigate other amino acid properties as hydropathy index and molecular volume and how these properties can be incorporated to the fitness function. The results suggest that polar requirement is an important property to be taken into account, but should not be the only one.

2. METHODS
The GA used here is implemented using C++ programming language. The used GA encoding was the non-restrictive presented in [11]. In this encoding, each individual of the GA's population is composed of 61 positions, each one related to one codon. Each position corresponds to one of 20 amino acids. In this sense, each GA's individual encodes a hypothetical genetic code.

The GA uses two reproduction operators: swap and mutation. The first one interchanges amino acids associated to two codons, i.e., two positions are randomly selected and their amino acids are swapped. In the mutation operator, a position is selected in the individual and its corresponding amino acid is replaced by another one, selected among the 20 possible amino acids. The position and the new amino acid are randomly selected. Tournament selection is employed to select the individuals to be reproduced.

2.1 Robustness-based evaluation function
The standard evaluation (fitness) function commonly employed in literature is the mean square (M_s) change in an amino acid property, which computes all possible changes to each base of all codons of a given code [6], [8],[11], [5], [7]. The measure M_s is defined as:

$$M_s(C) = \frac{\sum_{ij}(X(i,C) - X(j,C))^2}{\sum_{ij} N(i,j,C)} \quad (1)$$

where $X(i, C)$ is the amino acid property value for the amino acid codified by the i-th codon for the genetic code C, and $N(i, j, C)$ is the number of possible replacements between codons i and j for the code C. For example, when the robustness consider the polar requirement, $X(i, C)$ represents the polar requirement for the amino acid codified by the i-th codon for the genetic code C.

2.2 Comparison of the evaluation functions

In order to compare the canonical genetic code to the best codes obtained by the GA, we use two measures:

- Percentage of Minimization Distance (*pmd*), as described in [4];

  The *pmd* is computed as follows:

  $$pmd = 100\frac{\Delta_{mean} - \Delta_{code}}{\Delta_{mean} - \Delta_{low}} \quad (2)$$

  where Δ_{mean} is the average fitness of genetic codes randomly generated, Δ_{code} is the fitness of the canonical genetic code, and Δ_{low} is the fitness of the best code found by the GA.

  Higher values of *pmd* means that the fitness of the best generated code and the fitness of the canonical genetic code are closer when compared to the average fitness of the random codes, i.e., the *pmd* indicates how close is the fitness of the canonical code to the fitness of the best code found by the GA when compared to the average fitness of the random codes.

- Improvement, as mentioned in [11];

  The improvement, gives the percentage of the best code improvement in relation to the canonical code fitness, i.e.:

  $$Improvement = 100\frac{\Delta_{code} - \Delta_{low}}{\Delta_{code}} \quad (3)$$

  Improvement decreases as the *pmd* increases, providing a measurement of how the best code found improved the fitness.

3. RESULTS AND CONCLUSIONS

We performed experiments with multi-objective approaches, considering not only polar requirement, but also molecular volume and hydropathy index. The obtained results were submitted to BMC Bioinformatics journal and are still under review. Some previous already published results can be seen in [3, 2]. In general, our results show that polar requirement seems to be a important property to be taken into account, but cannot not be the only.

4. ACKNOWLEDGMENTS

The authors would like to thank Fapesp and CNPq for the financial support.

5. REFERENCES

[1] F. H. Crick. The origin of the genetic code. *Journal of Molecular Biology*, 38(3):367–379, 1968.

[2] L. L. de Oliveira and R. Tinós. Entropy-based evaluation function for the investigation of genetic code adaptability. In *BCB '12: Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 558–560, New York, NY, USA, 2012. ACM.

[3] L. L. de Oliveira and R. Tinós. Using base position errors in an entropy-based evaluation function for the study of genetic code adaptability. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, pages 99–111. Springer, 2014.

[4] M. Di Giulio. The extension reached by the minimization of the polarity distances during the evolution of the genetic code. *Journal of Molecular evolution*, 29(4):288–293, 1989.

[5] M. Di Giulio, M. Capobianco, and M. Medugno. On the optimization of the physicochemical distances between amino acids in the evolution of the genetic code. *Journal of Theoretical Biology*, 168(1):43–51, 1994.

[6] S. J. Freeland and L. D. Hurst. The genetic code is one in a million. *Journal of Molecular Evolution*, 47(3):238–248, 1998.

[7] N. Goldman. Further results on error minimization in the genetic code. *Journal of Molecular Evolution*, 37(6):662–664, 1993.

[8] D. Haig and L. D. Hurst. A quantitative measure of error minimization in the genetic code. *Journal of Molecular Evolution*, 33(5):412–417, 1991.

[9] A. L. Lehninger, D. L. Nelson, and M. M. Cox. *Lehninger Principles of Biochemistry*. W. H. Freeman, 4th edition, 2005.

[10] H. Lodish, A. Berk, S. L. Zipursky, C. A. Kaiser, M. Krieger, M. P. Scott, A. Bretscher, H. Ploegh, and P. Matsudaira. *Molecular Cell Biology*. W. H. Freeman, 6th edition, 2007.

[11] J. Santos and A. Monteagudo. Study of the genetic code adaptability by means of a genetic algorithm. *Journal of Theoretical Biology*, 264(3):854–865, 2010.

[12] S. Schoenauer and P. Clote. How optimal is the genetic code. In *Computer Science and Biology, Proceedings of the German Conference on Bioinformatics (GCB'97)*, pages 65–67, 1997.

[13] C. R. Woese. On the evolution of the genetic code. *Proceedings of the National Academy of Sciences of the United States of America*, 54(6):1546–1552, 1965.

A machine for higher-order term rewriting

Connor Lane Smith
School of Computing
University of Kent
Canterbury, UK
cls204@kent.ac.uk

ABSTRACT

Presented is a machine for higher-order term rewriting. The machinery has two main components: first, a reducer for the λ-calculus, a fusion of the Krivine machine and σ-calculus; and a rewriter which drives the former component so as to find the most general unifier for a higher-order pattern and perform a rewrite step. These two components then work together to compute a given term's strong normal form.

1. INTRODUCTION

Higher-order term rewriting [7] is a powerful generalisation of standard first-order term rewriting in which rewrites are performed modulo the simply-typed λ-calculus. Although there have been a number of implementations published for first- and second-order term rewriting systems, none for third- and higher-order rewrite systems have appeared (yet presumably some have been written and kept private).

Here I present a machinery for performing higher-order term rewriting, comprising two main components. The first is a reducing engine for the λ-calculus, based on a fusion of the Krivine machine [3] — an abstract machine for the weak reduction of the λ-calculus — and the σ-calculus of explicit substitutions [1], with which we achieve strong reduction. The second component is a relatively conventional rewriting engine, but instead of working with terms themselves, it drives the former component and uses its output in the process of pattern matching. Thus the rewrites of the latter component are performed, in a sense, modulo the conversions of the former.

We shall walk through the specifics of the machinery, starting with the λ-calculus with De Bruijn indices, followed by a brief description of higher-order rewriting, and then onto each of the machine's components. The subject of this paper is a work in progress, and so some ongoing developments and analyses are outside the scope of this paper.

2. LAMBDA CALCULUS

The λ-calculus is traditionally defined with named bound variables, which can be renamed whilst retaining structural equivalence. However, the correctness of naming is, I feel, overly intricate considering its being wholly syntactic — throughout the literature its definition has been fraught with mistakes. I will avoid these and similar problems by using De Bruijn indices [2] exclusively, as is done in almost all concrete implementations of the calculus.

Intuitively, a variable's De Bruijn index indicates the number of abstractions through which one must pass upwards before finding its binder. For example, $\lambda x.x(\lambda y.xy)$ is represented with De Bruijn indices as $\lambda\underline{0}(\lambda\underline{10})$. Note that the first $\underline{0}$ represents x, the second y.

Definition 1. *The set $\mathcal{S}$ of* simple types *is the closure of a fixed set $\mathcal{A}$ of* type atoms *under the function space constructor $\to$.*

Definition 2. *We'll use a number of list and list-like data types. I use ϵ for the empty list, and $\cdot$ for the* cons *operator (i.e. type $[\alpha] = \epsilon \mid \alpha \cdot [\alpha]$). I may write just x for $x \cdot \epsilon$ where there is no ambiguity.*

Definition 3. *A* basis Γ *is a list of simple types, with which we may derive a well-typed* term t*, written $\Gamma \vdash t : \tau$, by the following rules:*

$$\begin{aligned} \sigma \cdot \Gamma &\vdash \underline{0} : \sigma \\ \Gamma \vdash \underline{n} : \sigma &\implies \rho \cdot \Gamma \vdash \underline{n{+}1} : \sigma \\ \sigma \cdot \Gamma \vdash t : \tau &\implies \Gamma \vdash \lambda t : \sigma \to \tau \\ \Gamma \vdash s : \sigma \to \tau \wedge \Gamma \vdash t : \sigma &\implies \Gamma \vdash st : \tau \\ K : \tau \in \Sigma &\implies \Gamma \vdash K : \tau \end{aligned}$$

Definition 4. *A term's* variable string *is a list of booleans ("bits") indicating whether a particular variable occurs* free *in the term. This is related to, but conceptually simpler than, director strings [5]. Variable string form a Boolean algebra with* bitwise OR *for $\vee$,* bitwise AND *for $\wedge$,* bitwise NOT *for $\neg$, and infinite sequences of $\bot$ and $\top$ as 0 and 1, respectively.*

$$\begin{aligned} \phi &= \phi \cdot \bot \\ VS(\underline{0}) &= \top \\ VS(\underline{n}) = \bot \cdot \phi &\implies VS(\underline{n{+}1}) = \phi \\ VS(t) = \alpha \cdot \phi &\implies VS(\lambda t) = \phi \\ VS(s) = \phi \wedge VS(t) = \psi &\implies VS(st) = \phi \vee \psi \\ VS(K) &= \epsilon \end{aligned}$$

The set $FV(t)$ of free variables in t *is the set whose characteristic function is $\{\underline{n} \mapsto VS(t)_n\}$.*

Definition 5. *A* substitution θ *is a mapping from variables to terms of the same type, which may be lifted to a homomorphism over terms, written $\hat{\theta}(t)$.*

DEFINITION 6. *β-reduction is a relation $(\lambda s)t \to_\beta \hat{\theta}(s)$ where $\theta = \{\underline{0} \mapsto t, \underline{n+1} \mapsto \underline{n}\}$. η-reduction is a relation $\lambda(t\underline{0}) \to_\eta \hat{\theta}(t)$ where $\underline{0} \notin FV(t)$ and $\theta = \{\underline{n+1} \mapsto \underline{n}\}$.*

γ-reduction is a union relation $\to_\gamma = \to_\beta \cup \to_\eta$, closed under contexts and substitutions:

$$\begin{aligned} t \to t' &\implies \lambda t \to \lambda t' \\ t \to t' &\implies st \to st' \\ s \to s' &\implies st \to s't \\ t \to t' &\implies \hat{\theta}(t) \to \hat{\theta}(t') \end{aligned}$$

A reduction is said to be weak *if it cannot occur under an abstraction,* strong *otherwise.*

3. HIGHER-ORDER REWRITING

We deal here with Nipkow's Higher-order Rewrite Systems (HRSs) [7], or, strictly speaking, *higher-order pattern rewrite systems*, to which they are most commonly restricted. The intuition here is that rewrite steps are performed modulo the simply-typed λ-calculus.

DEFINITION 7. *A* higher-order pattern *[6] is a β-normal term in which a free variable f may only occur in the form $ft_1 \dots t_k$, where each t_i is η-equivalent to a distinct bound variable.*

DEFINITION 8. *A Higher-order Rewrite System $\mathcal{H}$ is a set of* rewrite rules *(l, r) where l is a term and r a pattern of the same atomic type, and $FV(l) \supseteq FV(r)$. Each rule $\mathcal{R}$ induces a rewrite relation $t \to_\mathcal{R} t'$ where $\hat{\theta}(l) \leftrightarrow_\gamma^* t$ and $t' \leftrightarrow_\gamma^* \hat{\theta}(r)$. $\mathcal{H}$ then induces the union $\to_\mathcal{H} = \bigcup_{\mathcal{R}\in\mathcal{H}} \to_\mathcal{R}$.*

DEFINITION 9. *A rule is* left-linear *if each variable on the left-hand side occur only once. Two rules with left-hand sides l_1 and l_2* overlap *if a subterm of l_2, t, is such that t is not a free variable and $\hat{\sigma}(l_1) \leftrightarrow_\gamma^* \hat{\theta}(t)$. An HRS is* orthogonal *if all its rules are left-linear and none overlap.*

For the remainder of this paper we will assume that all HRSs dealt with are orthogonal.

4. K-SIGMA MACHINE

The *Krivine machine*, or K-machine [3], is an abstract machine for the weak reduction of λ-terms, defined by a small set of *transition rules*, below.

$$\langle st, \sigma, stack\rangle \to \langle s, \sigma, t[\sigma] \cdot stack\rangle \quad (1)$$
$$\langle \lambda s, \sigma, t[\rho] \cdot stack\rangle \to \langle s, t[\rho] \cdot \sigma, stack\rangle \quad (2)$$
$$\langle \underline{0}, t[\rho] \cdot \sigma, stack\rangle \to \langle t, \rho, stack\rangle \quad (3)$$
$$\langle \underline{n+1}, t[\rho] \cdot \sigma, stack\rangle \to \langle \underline{n}, \sigma, stack\rangle \quad (4)$$

For strong reduction we need more sophisticated machinery for substitution. With this in mind we look to the *$\lambda\sigma$-calculus* [1], a 'substitution calculus' that renders the higher-order λ-calculus into a first-order term rewriting system. The σ-calculus can be seen as a kind of data structure for substitutions.

$$\begin{aligned} \mathrm{id} &= \{\underline{n} \mapsto \underline{n}\} \\ \uparrow &= \{\underline{n} \mapsto \underline{n+1}\} \\ t \cdot \sigma &= \{\underline{0} \mapsto t, \underline{n+1} \mapsto \sigma(\underline{n})\} \\ \rho\,;\sigma &= \sigma \circ \rho \end{aligned}$$

σ-substitutions then form a homomorphism over terms:

$$\begin{aligned} (\lambda s)t &= s[t \cdot \mathrm{id}] \\ \underline{n}[\sigma] &= \sigma(\underline{n}) \\ (st)[\sigma] &= (s[\sigma])(t[\sigma]) \\ (\lambda t)[\sigma] &= \lambda(t[\underline{0} \cdot (\sigma\,;\uparrow)]) \end{aligned}$$

Inspired by the $\lambda\sigma$-calculus, I have extended the K-machine to the *Kσ-machine*, which generalises environment stack of the K-machine's to a σ-substitution, thus unlocking strong reduction.

$$\langle \lambda t, \sigma, \mathrm{id}\rangle \to \langle t, \underline{0}[\mathrm{id}] \cdot (\sigma\,;\uparrow), \epsilon\rangle \quad (5)$$
$$\langle \underline{n}, \uparrow, stack\rangle \to \langle \underline{n+1}, \mathrm{id}, stack\rangle \quad (6)$$
$$\langle \underline{n}, (\pi\,;\rho)\,;\sigma, stack\rangle \to \langle \underline{n}, \pi\,;(\rho\,;\sigma), stack\rangle \quad (7)$$
$$\langle \underline{0}, (t[\pi] \cdot \rho)\,;\sigma, stack\rangle \to \langle t, \pi\,;\sigma, stack\rangle \quad (8)$$
$$\langle \underline{n+1}, (t[\pi] \cdot \rho)\,;\sigma, stack\rangle \to \langle \underline{n}, \rho\,;\sigma, stack\rangle \quad (9)$$
$$\langle \underline{n}, \mathrm{id}\,;\sigma, stack\rangle \to \langle \underline{n}, \sigma, stack\rangle \quad (10)$$
$$\langle \underline{n}, \uparrow\,;\sigma, stack\rangle \to \langle \underline{n+1}, \sigma, stack\rangle \quad (11)$$

If we allow ourselves to step a little further from the original K-machine, we can see that the Kσ-machine comprises two essential steps, *traversal* and *lookup*. Lookup occurs only when a variable is encountered, walking through a substitution so as to find the correct substitute term, or else the correctly adjusted De Bruijn index. In OCaml:

```
type subst = Comp of subst * subst
           | Cons of (int,thunk) either * subst
           | Id
           | Shift
 and thunk = term * subst

let lookup : int -> subst -> (int,thunk) either =
  fun i r ->
    match r with
    | Comp(p,q)            ->
        match lookup i p with
        | Left(j)      -> lookup j q
        | Right((t,p)) -> Right((t, Comp(p,r)))
    | Cons(x,_) when i = 0 -> x
    | Cons(_,p)            -> lookup (i-1) p
    | Id                   -> Left(i)
    | Shift                -> Left(i+1)
```

5. REWRITE MACHINE

The rewrite machine uses the λ-calculus machine to reduce a closure $s[\rho]$ (initially $s[\mathrm{id}]$) by driving a machine $\langle s, \rho, \epsilon\rangle$ — whilst counting the number of times the machine enters state (5), which we will call k — until it terminates. Upon termination the machine's state will be $\langle t_0, \sigma_0, t_1[\sigma_1] \cdots t_n[\sigma_n]\rangle$, where t_0 is a either constant or a variable with an empty

stack. It will have then computed the spine normal form of s: $\lambda \ldots \lambda t_0[\sigma_0] t_1[\sigma_1] \ldots t_n[\sigma_n]$, with k abstractions.

The complete normal form can thus be found spine by spine, so any higher-order unification algorithm for the $\lambda\sigma$-calculus will correctly match rewrite rules; one such unification algorithm is that of Dowek, et al. [4]. However, we need only deal with a particular subclass of the problem, namely *orthogonal higher-order pattern matching*, for we can devise a far more efficient algorithm, in which we separately (1) perform a free variable check, and then (2) construct a most general unifier, using what we know about the structure of orthogonal higher-order patterns to outpace a more general higher-order unification algorithm.

5.1 Free variable checks

We label each term with its variable string, so that we needn't traverse the term in order to determine whether it matches a free variable on the left-hand side of a rule. For this we close variable strings under σ-substitutions:

$$\begin{aligned} \mathrm{VS}(t[\sigma]) &= \mathrm{VS}(t)[\sigma] \\ \phi[\mathrm{id}] &= \phi \\ \phi[\uparrow] &= \bot \cdot \phi \\ (\top \cdot \phi)[t \cdot \sigma] &= \phi[\sigma] \vee \mathrm{VS}(t) \\ (\bot \cdot \phi)[t \cdot \sigma] &= \phi[\sigma] \\ \phi[\rho\,;\sigma] &= \phi[\rho][\sigma] \end{aligned}$$

It is worth noting that, with this approach to matching, parallel–outermost reduction is not necessarily normalising for orthogonal pattern rewrite systems: these labels denote only whether a variable is free in the term, not whether it is *needed* free. This means that a rule $F(\lambda\underline{1}) \to G$ may not at first match a term $F(\lambda(\lambda G)\underline{0})$, as although $F(\lambda\underline{1})[G \cdot \mathrm{id}] \leftrightarrow^*_\gamma F(\lambda(\lambda G)\underline{0})$, the variable bound by the outermost abstraction is still free in its unreduced subterm. This has no effect on the correctness of the system, however: normal forms are preserved.

5.2 Most general unifier

For simplicity, we will require (w.l.o.g.) that, as we traverse the term, the De Bruijn index of each free variable encountered be one less than the previous, and their arguments' indices (all being variables) be increasing. We begin with the substitution [id], and then for each free variable prepend a new substitute so as to build the unifier.

When matching a free variable instance $f u_1 \ldots u_k$ with a term t, we first perform a variable check: let $\psi = \bigvee_i^k \mathrm{VS}(u_i)$; if $\psi \wedge \mathrm{VS}(t) \neq \psi$ then matching fails. We then enumerate the tops ($\top$) of ψ such that the first is replaced with $\underline{0}$, the second with $\underline{1}$, and so on. Bottoms ($\bot$) are left as is, representing the null-term; since the variable cannot occur, it will never be a substitute, and so any term will do. We then build the term $\lambda \ldots \lambda t[\sigma]$, where there are n abstractions and σ is the substitution built as described. This value is then prepended to the unifier.

Example 1. *A rule* $\mathcal{R} = (A(\lambda\lambda\lambda\lambda B(\underline{513})), C(\lambda\underline{10}D))$, *having the variable string* $VS(\underline{1},\underline{3}) = \bot \cdot \top \cdot \bot \cdot \top \cdot \bot$, *induces the following rewrite relation:*

$$A(\lambda\lambda\lambda\lambda Bt) \to_{\mathcal{R}} C(\lambda\underline{10}D)[\lambda\lambda t[\bot \cdot \underline{0} \cdot \bot \cdot \underline{1} \cdot \bot \cdot id] \cdot id]$$

This rewriting proceeds in a leftmost–outermost fashion until a strong normal form is found, or else fails to terminate (if, for example, the HRS is not strongly normalising). Note that although a pattern's most general unifier is unique *with respect to equality*, there may exist isomorphic substitutions — or different rewrite mechanisms altogether — better suited to this approach to reduction. This is to be investigated further.

6. CONCLUSION

This approach to rewriting is very different to those machines dealing with lower-order rewriting, and should, by taking advantage of the higher-order nature of the rewrite system, open up a number of new possibilities for sharing that cannot be utilised without the higher-order machinery inherent to HRSs.

Here we have only dealt with terms in their simplest form, which is when they are modelled as immutable trees. Where this approach to evaluation really shines, however, is when the machine works on term graphs, à la Wadsworth [8]. Viewing the HRSs as being embedded in a term graph rewriting system is more faithful to the underlying representation of the terms in memory, and is where the Kσ-machine distinguishes itself from the other environment machines as far as higher-order rewriting is concerned.

7. REFERENCES

[1] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. In *17th ACM SIGPLAN–SIGACT Symp. Principles of Programming Languages*, pages 31–46, 1989.

[2] Nicolaas G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem. In *Indagationes Mathematicae*, volume 75, pages 381–392, 1972.

[3] Pierre Crégut. An abstract machine for lambda-terms normalization. In *ACM Conf. LISP and Functional Programming*, pages 333–340, 1990.

[4] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher-order unification via explicit substitutions. *Information and Computation*, 157(1):183–235, 2000.

[5] Maribel Fernández, Ian Mackie, and François-Régis Sinot. Lambda-calculus with director strings. *Applicable Algebra in Engineering, Communication and Computing*, 15(6):393–437, 2005.

[6] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.

[7] Tobias Nipkow. Higher-order critical pairs. In *6th IEEE Symp. Logic in Computer Science*, pages 342–349, 1991.

[8] Christopher P. Wadsworth. *Semantics and Pragmatics of the Lambda Calculus*. PhD thesis, University of Oxford, 1971.

Gene Ontology Hierarchy-Based Feature Selection

Cen Wan
School of Computing
University of Kent
Canterbury, United Kingdom
cw439@kent.ac.uk

Alex A. Freitas
School of Computing
University of Kent
Canterbury, United Kingdom
A.A.Freitas@kent.ac.uk

ABSTRACT
We address the classification task of data mining, where the model organism *C. elegans'* genes are classified into "pro-longevity" or "anti-longevity" genes. We adopted hierarchically organised Gene Ontology (GO) terms as features, and proposed one feature selection algorithm to alleviate the redundancy between GO's hierarchy. The computational results show that the proposed algorithm can significantly improve the predictive performance of the Naïve Bayes classifier.

General Terms
Algorithms

Keywords
Classification, Feature Selection, Hierarchy, Gene Ontology

1. INTRODUCTION
This is an extended abstract of our recent work described in [3]. We address the classification task of data mining, where the model organism *C. elegans'* genes are classified into "pro-longevity" or "anti-longevity" genes. We created a dataset integrating data from Human Ageing Genomic Resources [1] and Gene Ontology (GO) [2] database. There is a type of "is_a" relationship among GO terms, which are used as features in our dataset. That means one GO term might have one or more parent GO terms. Due to this hierarchical relationship, there is redundancy between GO terms (features). Hence, we proposed a feature selection algorithm that is able to effectively alleviate the redundancy between features, as a pre-processing step for classifying the *C. elegans'* genes into "pro-" or "anti-longevity".

2. PROPOSED METHOD
The proposed feature selection algorithm firstly evaluates the relevance of each feature based on its predictive power, then deletes features based on the hierarchical relationship among features. In more detail, when classifying a new instance, if the value of one GO term equals to "1" (i.e. the GO term is present in that instance), then we delete its ancestor GO terms whose relevance values are equal or lower than that GO term's relevance, since those ancestors are redundant. If the value of one GO term equals to "0" (i.e. the GO term is absent in that instance), then we delete its descendant GO terms whose relevance values are equal or lower than that GO term's relevance, since those descendants are redundant.

3. COMPUTATIONAL RESULTS
The classification algorithm used in this work is Naïve Bayes, which is known to be sensitive to redundant features. In our experiments, Naïve Bayes using only the features (GO terms) selected by our feature selection algorithm obtained an average accuracy rate of 68.1%, sensitivity of 57.5%, and specificity of 72.6%. As a baseline, Naïve Bayes using all original features (i.e. without feature selection) obtained average accuracy of 62.5%, sensitivity of 51.9%, and specificity of 69.2%. Hence, the proposed feature selection algorithm significantly optimizes the predictive performance of Naïve Bayes.

4. CONCLUSION
In conclusion, information on the hierarchical structure of GO terms (features) was valuable for alleviating feature redundancy and so improving the predictive performance of Naïve Bayes, in our dataset of longevity-related *C. elegans'* genes.

5. REFERENCES
[1] J. P. de Magalhaes, A. Budovsky, G. Lehmann, J. Costa, Y. Li, V. Fraifeld, and G. M. Church. The human ageing genomic resources: online databases and tools for biogerontologists. *Aging Cell*, 8(1):65–72, Feburary 2009.

[2] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.

[3] C. Wan and A. A. Freitas. Prediction of the pro-longevity or anti-longevity effect of *Caenorhabditis Elegans* genes based on bayesian classification methods. In *IEEE International Conference on Bioinformatics and Biomedicine Proceedings*, pages 373–380. IEEE, December 2013.

ERP latency contrasts using Dynamic Time Warping algorithm

Alexia Zoumpoulaki
University of Kent
az61@kent.ac.uk

Abdulmajeed Alsufyani
University of Kent
asa41@kent.ac.uk

Howard Bowman
University of Kent
H.Bowman@kent.ac.uk

Keywords
ERP, latency contrasts

Poster Presentation
Latency contrasts are central to Event Related Potential (ERP) research. For example, they are used to determine the order and length of cognitive processes or to evaluate how experimental conditions influence processing time. The most popular methods employed by researchers are peak latency, fractional peak and fractional area. However there are difficulties with these methods, which often include acute sensitivity to noise and window placements [2]. In addition, they require parameter settings that are often difficult to justify. In order to address these issues, we propose Dynamic Time Warping (DTW), an algorithm used for measuring similarity between two sequences, and more precisely the use of the warping path and its relationship to the main diagonal as a measure of latency difference. We tested the performance of DTW by comparing it to 25% - 50% fractional area, peak and 50% fractional peak. In addition to the default DTW we also tested the type IIa step pattern, which constraints the resulting warping path [3]. Data were obtained from a deception detection experiment [4] and ERPs were generated from two channels for one condition. Then the second condition was created in two ways: firstly by offsetting the first condition by 100 time points (0.05 ms), and secondly by offsetting the first condition by an amount sampled from a normal distribution with a mean of 100 time points, while also varying the amplitude. In this way, we simulated latency jitter and amplitude variability between conditions as found in real ERP experiments. Each method was applied to windows placed accordingly to relevant experiments. This was done for each one of the channels (P3a at Fz and P3b at Pz). Then noise was added at the power spectrum of human EEG [1], and the performance of each method was evaluated through permutation tests (100 p-values) for different Signal-to-Noise Ratios (SNR). In order to test the methods independently of response bias, we performed ROC (Receiver Operating Characteristic) analysis, where the false positive rate was obtained by generating the second condition without any latency difference. We also compared DTW's sensitivity to window placement against 25% and 50% fractional area, by selecting a fixed time point as the start of the bounding window and then sequentially adjusting the end of the window and calculating the proportion of windows, for which each of the methods failed to determine the correct latency difference.

Our analysis shows that the basic DTW performs at the same level as the fractional area method, which outperforms peak and fractional peak. The typeIIa DTW performs better than all the other methods for most SNRs. Although there is a slight inflation of false positives for high SNRs, the ROC analysis shows that it does not substantially affect DTW's performance. At the same time, DTW shows significantly less sensitivity to window placement than fractional area. These results indicate that DTW is a promising technique for determining latency differences, being more robust to noise and window placement, without being subject to the same number of assumptions or parameterisation as the other methods evaluated.

1. REFERENCES
[1] Generation of simulated eeg data. http://www.cs.bris.ac.uk/ rafal/phasereset. Accessed: 2014-05-05.

[2] K. Andrea, J. Miller, P. Jolicceur, and B. Brisson. Measurement of erp latency differences: A comparison of single?participant and jackknife?based scoring methods. *Psychophysiology*, 45(2):250–274, 2008.

[3] M. Cory. *A comparative study of several dynamic time warping algorithms for speech recognition*. PhD thesis, Massachusetts Institute of Technology, 1980.

[4] B. Howard, M. Filetti, D. Janssen, L. Su, A. Alsufyani, and B. Wyble. Subliminal salience search illustrated: Eeg identity and deception detection on the fringe of awareness. *PloS one*, 8(1), 2013.